Chapter

Studies in Fuzziness and Soft Computing
Publisher: Springer Berlin / Heidelberg
ISSN: 1434-9922 (Paper) 1860-0808 (Online)
Subject: Engineering

Volume 180 / 2005
Title: Foundations and Advances in Data Mining
Editors: Wesley Chu, Tsau Lin
ISBN: 3-540-25057-3
DOI: 10.1007/b104039

Chapter: pp. 221 - 274
DOI: 10.1007/11362197_9

Web Page Classification*

B. Choi[1] and **Z. Yao**[1]

(1) Computer Science, College of Engineering and Science, Louisiana Tech University, Ruston, LA 71272, USA

Abstract
This chapter describes systems that automatically classify web pages into meaningful categories. It first defines two types of web page classification: subject based and genre based classifications. It then describes the state of the art techniques and subsystems used to build automatic web page classification systems, including web page representations, dimensionality reductions, web page classifiers, and evaluation of web page classifiers. Such systems are essential tools for Web Mining and for the future of Semantic Web.

B. Choi
Email: pro@BenChoi.org

Z. Yao
Email: zya001@latech.edu

The references of this article are secured to subscribers.

Web Page Classification*

B. Choi and Z. Yao

Computer Science, College of Engineering and Science, Louisiana Tech University, Ruston, LA 71272, USA
pro@BenChoi.org, zya001@latech.edu

Abstract. This chapter describes systems that automatically classify web pages into meaningful categories. It first defines two types of web page classification: subject based and genre based classifications. It then describes the state of the art techniques and subsystems used to build automatic web page classification systems, including web page representations, dimensionality reductions, web page classifiers, and evaluation of web page classifiers. Such systems are essential tools for Web Mining and for the future of Semantic Web.

1 Introduction

1.1 Motivation

Over the past decade we have witnessed an explosive growth on the Internet, with millions of web pages on every topic easily accessible through the Web. The Internet is a powerful medium for communication between computers and for accessing online documents all over the world but it is not a tool for locating or organizing the mass of information. Tools like search engines assist users in locating information on the Internet. They perform excellently in locating but provide limited ability in organizing the web pages. Internet users are now confronted with thousands of web pages returned by a search engine using simple keyword search. Searching through those web pages is in itself becoming impossible for users. Thus it has been of more interest in tools that can help make a relevant and quick selection of information that we are seeking. This chapter presents one of the most promising approaches: web page classification, which can efficiently support diversified applications, such as Web Mining, automatic web page categorization [17], information filtering, search engine, and user profile mining. It describes the state of the art techniques and subsystems used to build automatic web page classification

*This research was supported in part by a grant from the Center for Entrepreneurship and Information Technology (CEnIT), Louisiana Tech University.

B. Choi and Z. Yao: *Web Page Classification*, StudFuzz **180**, 221–274 (2005)
www.springerlink.com

systems. It starts with a definition of web page classification and a description of two types of classifications.

1.2 A Definition of Web Page Classification

Web page classification, also known as web page categorization, may be defined as the task of determining whether a web page belongs to a category or categories. Formally, let $C = \{c_1, \ldots, c_K\}$ be a set of predefined categories, $D = \{d_1, \ldots, d_N\}$ be a set of web pages to be classified, and $A = D \times C$ be a decision matrix:

Web Pages	Categories				
	c_1	…	c_j	…	c_K
d_1	a_{11}	…	a_{1j}	…	a_{1K}
…	…	…	…	…	…
d_i	a_{i1}	…	a_{ij}	…	a_{iK}
…	…	…	…	…	…
d_N	a_{N1}	…	a_{Nj}	…	a_{NK}

where, each entry a_{ij} $(1 \leq i \leq N,\ 1 \leq j \leq K)$ represents whether web page d_i belongs to category c_j or not. Each $a_{ij} \in \{0, 1\}$ where 1 indicates web page d_i belongs to category c_j, and 0 for not belonging. A web page can belong to more than one category. The task of web page classification is to approximate the unknown assignment function $f : D \times C \rightarrow \{0, 1\}$ by means of a learned function $f' : D \times C \rightarrow \{0, 1\}$, called a classifier, a model, or a hypothesis, such that f' coincides to f as much as possible [98].

The function f' is usually obtained by machine learning over a set of training examples of web pages. Each training example is tagged with a category label. The function f' is induced during the training phase and is then used during the classification phase to assign web pages to categories.

Throughout this chapter, we use "document" to denote a text document including web pages, and use "HTML page", "web document", or "web page" to indicate a web page. We also emphasize the characteristics of web page classification that are different from traditional text classification.

1.3 Two Basic Approaches

Web page classification is in the area of machine learning, where learning is over web pages. Using machine learning techniques on text databases is referred to as text learning, which has been well studied during the last two decades [71]. Machine learning on web pages is similar to text learning since

web pages can be treated as text documents. Nevertheless, it is clear in advance that learning over web pages has new characteristics. First, web pages are semi-structured text documents that are usually written in HTML. Secondly, web pages are connected to each other forming direct graphs via *hyperlinks*. Thirdly, web pages are often short and by using only text in those web pages may be insufficient to analyze them. Finally, the sources of web pages are numerous, nonhomogeneous, distributed, and dynamically changing. In order to classify such large and heterogeneous web domain, we present in this chapter two basic classification approaches: *subject-based classification* and *genrebased classification*.

In *subject-based classification (also called topic-based classification)*, web pages are classified based on their contents or subjects. This approach defines numerous topic categories. Some examples of categories, under the "Science" domain used in Yahoo.com, are "Agriculture", "Astronomy", "Biology", "Chemistry", "Cognitive Science", "Complex Systems", and "Computer Science". The subject-based classification can be applied to build topic hierarchies of web pages, and subsequently to perform contextbased searches for web pages relating to specific subjects.

In *genre-based classification (also called style-based classification)*, web pages are classified depending on functional or genre related factors. In a broad sense, the word "genre" is used here merely as a literary substitute for "a kind of text". Some examples of web page genres are "product catalogue", "online shopping", "advertisement", "call for paper", "frequently asked questions", "home page", and "bulletin board". This approach can help users find immediate interests. Although text genre has been studied for a long history in linguistic literature, automatically text genre classification does not share much literature and fewer sophisticated research work has been done on web page genre classification. One important reason is that, up to recently, the digitized collections have been for the most part generically homogeneous, such as collections of scientific abstracts and newspaper articles. Thus the problem of genre identification could be set aside [52]. This problem does not become salient until we are confronted with heterogeneous domain like the Web. In fact, the Web is so diverse that no topic taxonomy can hope to capture all topics in sufficiently detail. It is worth noticing that the genre-based approach is not to reduce the importance of the subject-based approach, but rather to add another dimension to web page classification as a whole.

1.4 Overview of this Chapter

This chapter addresses three main topics of web page classification: web page representations, dimensionality reductions, and web page classifiers. It then concludes with an evaluation of classification methods and a summary. Section 2 describes how to encode or represent web pages for facilitating machine learning and classification processes. Web pages are usually represented by multi-dimensional vectors [93], each dimension of which encodes a feature

of the web pages. If all features of web pages are used in the representations, the number of dimensions of the vectors will usually be very high (hundreds of thousands). To reduce both time and space for computation, various methods are introduced to reduce the dimensionality as described in Sect. 3. Machine learning methods for classifying web pages are then applied to induce the classification function f' as defined in Sect. 1.2 or to induce representations for categories from the representations of training web pages. Section 4 describes various classifiers that have been proven efficient for web page classification. When a web page needed to be classified, the classifiers use the learned function to assign the web page to categories. Some classifiers compare the similarity of the representation of the web page to the representations of the categories. The category having the highest similarity is usually considered as the most appropriate category for assigning the web page. Section 5 discusses how to evaluate web page classification methods and provides experimental results for comparing various classifiers. Finally, Sect. 6 provides a summary of this chapter.

2 Web Page Representations

The first step in web page classification is to transform a web page, which typically composes of strings of characters, hyperlinks, images, and HTML tags, into a feature vector. This procedure is used to remove less important information and to extract salient features from the web pages. Apparently, the subject-based classification prefers features representing contents or subjects of web pages and these features may not represent genres of the web pages. Here we present different web page representations for the two basic classification approaches.

2.1 Representations for Subject Based Classification

Most work for subject-based classifications believes the text source (e.g. words, phases, and sentences) represents the content of a web page. In order to retrieve important textual features, web pages are first preprocessed to discard the less important data. The preprocessing consists of the following steps:

- **Removing HTML tags:** HTML tags indicate the formats of web pages. For instance, the content within <title> and </title> pair is the title of a web page; the content enclosed by <table> and </table> pair is a table. These HTML tags may indicate the importance of their enclosed content and they can thus help weight their enclosed content. The tags themselves are removed after weighting their enclosed content.
- **Removing stop words:** stop words are frequent words that carry little information, such as prepositions, pronouns, and conjunctions. They are removed by comparing the input text with a "stop list" of words.

- **Removing rare words:** low frequency words are also removed based on Luhn's idea that rare words do not contribute significantly to the content of a text [64]. This is to be done by removing words whose number of occurrences in the text are less than a predefined threshold.
- **Performing word stemming:** this is done by grouping words that have the same stem or root, such as computer, compute, and computing. The Porter stemmer is a well-known algorithm for performing this task.

After the preprocessing, we select features to represent each web page. Before going into details of web page representations, we standardize to use M to denote the dimension of a vector (or the number of features in the training corpus), N for the total number of web pages in the training collection, i for the ith vector, and j for the jth feature of a vector.

Bag-of-terms Representation

In the simplest way, a web page is represented by a vector of M weighted index words (see Fig. 1). This is often referred to as *bag-of-words* representation. The basic hypothesis behind the bag-of-words representation is that each word in a document indicates a certain concept in the document. Formally, a web page is represented by a vector d_i with words $t_1, t_2, \ldots, t_M$ as the features, each of which associates with a weight w_{ij}. That is,

$$d_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{iM}) \tag{1}$$

where M is the number of indexing words and w_{ij} $(1 \leq j \leq M)$ is the importance (or weight) of word t_j in the web page d_i.

Since a phrase usually contain more information than a single word, the bag-of-words representation can be enriched by adding new features generated

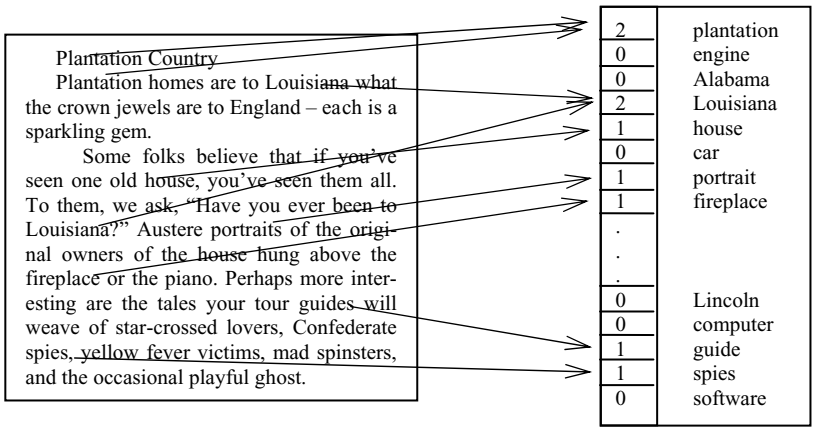

Fig. 1. Representing a web page in a vector space model. Each web page is converted into a vector of words in this case

from word sequences, also known as n-grams. For example, [71] generated features that compose up to five words (5-grams). By using n-grams we can capture some characteristic word combinations. Here we refer to the enriched bag-of-words representation as bag-of-terms, where a term can be a single word or any n-gram. The process of feature generation is performed in passes over documents, where i-grams are generated in the ith pass only from the features of length $i-1$ generated in the previous pass [71]. Hereafter, we use "word" to denote a single word and use "term" to name an n-gram. Experiments showed that terms up to 3-grams are sufficient in most classification systems.

Deciding the weights of terms in a web page is a term-weighting problem. The simplest way to define the weight w_j of term t_j in a web page is to consider the binary occurrence as following:

$$w_j = \begin{cases} 1, & \text{if term } t_j \text{ is in the web page} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Term Frequency Inverse Document Frequency

One of the most successful term weighting methods is TFIDF (Term Frequency Inverse Document Frequency) [94, 95], which is obtained by the product of the local term importance (TF) and the global term importance (IDF):

$$w_{ij} = TF(t_j, d_i) \cdot IDF(t_j) \tag{3}$$

where term frequency $TF(t_j, d_i)$ is the number of times term t_j occurs in document d_i, and document frequency $DF(t_j)$ in (4) is the number of documents in which term t_j occurs at least once.

$$DF(t_j) = \sum_{i=1}^{N} \begin{cases} 1, & \text{if } d_i \text{ contains } t_j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The inverse document frequency $IDF(t_j)$ can be calculated from the document frequency $DF(t_j)$:

$$IDF(t_j) = \log\left(\frac{N}{DF(t_j)}\right) \tag{5}$$

where N is the total number of documents. Intuitively, the inverse document frequency of a term is low if it occurs in many documents and is the highest if the term occurs in only one document. Many experiments have supported the discriminant characteristic of the inverse document frequency [48]. TFIDF term weighting expresses that a term is an important feature for a document if it occurs frequently in the document, i.e. the term frequency is high. On the other hand, a term becomes an unimportant feature for the document if the term occurs in many documents, i.e. the inverse document frequency is low

[45]. Furthermore, the weight w_{ij} of term t_j in document d_i can be normalized by document length, e.g. Euclidian vector length (L^2-norm) of the document:

$$w_{ij} = \frac{TF(t_j, d_i)IDF(t_j)}{\sqrt{\sum_{j=1}^{M}(TF(t_j, d_i)IDF(t_j))^2}} \tag{6}$$

where M is the number of features (unique terms) in all training web pages. The normalization by document length can be considered as a part of term weighting.

Reference [45] proposed a modified version of the inverse document frequency. He modified the inverse document frequency as following:

$$DF'(t_j) = \sum_{i=1}^{N} \frac{TF(t_j, d_i)}{|d_i|} \tag{7}$$

$$IDF'(t_j) = sqrt\left(\frac{N}{DF'(t_j)}\right) \tag{8}$$

where the document length $|d_i|$ is the number of words in document d_i, and N is the total number of web pages in the training collection. As we can see, Joachims defined $DF'(t_j)$, document frequency of term t_j, as the sum of relative frequencies of term t_j in each web page. He argued that $IDF'(t_j)$ can make use of the frequency information instead of just considering binary occurrence information of $DF(t_j)$ (4). It can be noticed that the interpretations of $DF(t_j)$ and $DF'(t_j)$ are similar. The more often a term t_j occurs throughout the corpus, the higher $DF(t_j)$ and $DF'(t_j)$ will be. A difference is that the square root is used instead of the logarithm to dampen the effect of document frequency. Joachims then used $|d_i|$ for normalization instead of using Euclidian length, i.e.

$$w_{ij} = \frac{TF(t_j, d_i) \cdot IDF'(t_j)}{|d_i|} \tag{9}$$

The modified TFIDF term-weighting scheme applied in a probabilistic variant of Rocchio classifier showed performance improvement of up to 40% reduction of error rate in comparing to Rocchio classifier [45].

Balanced Term-weighting

The balanced term-weighting scheme [49] is optimized for similarity computation between documents, since similarity computation is often required in classification tasks. While the TFIDF term-weighting scheme only considers the terms that occur in documents, the basic premise under the balanced term-weighting scheme is that similarity between two documents is maximized if there are high number of matches not only in occurrence of terms but also in *absence* of terms. The main procedure of this scheme consists of the following four phases [49]:

1. Local term-weighting phase: assigns term frequency to the present terms, and assigns negative weights -1 for the absent terms. Use one vector to represent the positive weights, and another vector to represent the negative weights;
2. Global term-weighting phase: applies inverse document frequency for the positive and the negative weights;
3. Normalization phase: normalizes the positive weight vector and the negative weight vector independently by L^2-norm (Euclidean distance);
4. Merging phase: merges the positive and the negative weights into one vector.

By assigning negative weights to absence terms, the balanced termweighting scheme resolves the masking-by-zero problem in the dot product operation for calculating the similarity (or distance) of two vectors. Thus the similarity of two documents is increased if high number of same terms are absent in both documents. Experiments illustrated that the balanced term-weighting scheme achieved higher average precisions than other term-weighting schemes [49].

Web Structure Representation

The bag-of-terms web page representation does not exploit the structural information on the Web. There are at least two different kinds of structural information on the Web that could be used to enhance the performance of classification:

- The structure of an HTML representation which allows to easily identify important parts of a web page, such as its title and headings;
- The structure of the Web, where web pages are linked to each other via hyperlinks.

In this section, we will exploit different web page representation methods that are unique for the Web domain.

HTML Structure

For improving web page representation, exploiting HTML structure will help us identify where the more representative terms can be found. For example, we can consider that a term enclosed within the <title> and </title> tags is generally more representative for the topic of a web page than a term enclosed within the <body> and </body> tags. For instance, several sources (elements) for web page representation are:

- BODY, the body part of a web page;
- TITLE, the title of a web page;
- H1∼H6, section headings;
- EM, emphasized content;

- URL, hyperlinks that may contain descriptor for the linked web pages. If keywords can be extracted from an URL, these keywords are relatively important;
- META, the meta-description of a web page. It is invisible to users of web browsing, but it may provide description, keywords, and date for a web page.

A Structure-oriented Weighting Technique (SWT) [89] can be used to weight the important features in web pages. The idea of SWT is to assign greater weights to terms that belong to the elements that are more suitable for representing web pages (such as terms enclosed in TITLE tags). SWT is defined by the function [89]:

$$\mathrm{SWT}_w(t_j, d_i) = \sum_{e_k} (w(e_k) \cdot TF(t_j, e_k, d_i)) \tag{10}$$

where e_k is an HTML element, $w(e_k)$ denotes the weight assigned to the element e_k, and $TF(t_i, e_k, d_j)$ denotes the number of times term t_i is present in the element e_k of HTML page d_j. Reference [89] defined the function $w(e)$ as:

$$W(e) = \begin{cases} \alpha, & \text{if } e \text{ is META or TITLE} \\ 1, & \text{otherwise} \end{cases} \tag{11}$$

where, $\alpha = 2$, $\alpha = 3$, $\alpha = 4$, and $\alpha = 6$ were tested and compared with standard $TF(t_i, d_i)$. The experimental results showed that using SWT can improve classification accuracy.

A more sophisticated SWT was employed by [82]. They not only utilized the HTML elements but also the important sentences in a web page. The important sentences are identified by using Paice's scheme [81], such as,

- Sentences with most frequently used words;
- Sentences with words that occur within title or section headings;
- Sentences using cue words, such as "greatest" and "significant", or cue phrases, such as "the main aim of", "the paper describes" and "the purpose of";
- Sentences in the beginning or last part of the document; and
- Sentences at the beginning of each paragraph.

Reference [82] defined $w(e)$ function as shown in Table 1 and applied the function in (10) to weight the terms. Obviously, their local term-weighting scheme is more mature than the simple $TF(t_i, d_i)$ method. It is then applied together with global term-weight, the inverse document frequency, and normalization factor to weight the terms.

Hyper-textual Nature of Web Pages

Unlike the typical text database, web pages maybe contain no obvious textually clues as to their subjects. For example, the home page of *Microsoft.com*

Table 1. Weights assigned to different elements in a web page

$w(e)$	e (different elements)
2	First or last paragraph
3	The important sentences identified by cue words/phrases
4	<Title>
2	<H1>
2	<EM>
3	<Strong>
4	<Meta Name = "keywords" or "descriptions">
4	URL
1	Otherwise

does not explicitly mention the fact that Microsoft sells operating systems. Some web pages are very short providing few text information and some are non-text based, such as image, video, sound, or flash format. In addition to analyzing a web page itself, a feasible way to represent a web page is to use hyperlinks that link to other related web pages. The basic assumption made by link analysis is that a link is often created because of a subjective connection between the original web page and the linked web page.

Hyperlinks of web pages can be analyzed to extract additional information about the web pages. A hyperlink in web page A that points to or cite web page B is called an *in-link* (incoming link) of web page B; meanwhile, this hyperlink is also an *out-link* (outgoing link) of web page A. A hyperlink is associated with "anchortext" describing the link. For instance, a web page creator may create a link pointing to Google.com, and define the associated anchortext to be "My favorite search engine". Anchortext, since it is chosen by people who are interested in the cited web page, may better summarize the topic of the cited web page, such as the example from Yahoo.com (Fig. 2).

By making use of the information provided by hyperlinks, a target web page can be represented by not only its full text, but all terms in the web pages linking the target page (including in-links and out-links). However, the

<A HREF="http://www.fisita.com/">International Federation of Automotive Engineering Societies (FISITA)</A> - a global forum for transport engineers promoting advances in automotive engineering technology and working to create efficient, affordable, safe, and sustainable transportation worldwide.

Fig. 2. An example of hyperlink in web pages from Yahoo.com. The words between <A HREF = ...> and </A> are *anchortext*. The anchortext and the text nearby summarize the content of the cited web page

naïve combination of the local full text and the text in the linked web pages does not help the classification. Experiments conducted by [16] over the web pages of IBM patents and Yahoo corpus showed that the combined features increased the error rate of their system by 6% over the full text in the target page alone. Reference [77] also showed similar results that the performance of their classifier decreased by 24% when adding words in the linked web pages into the target web page in a collection of online encyclopedia articles. Hyperlinks clearly contain high quality semantic clues, but they are noisy. One reason is that those linked web pages go to a diverse set of topics and worse scenarios could be seen on the Web with completely unrelated topics.

Instead of adding all terms in the linked web pages into the target web page, a more appropriate way for web page representation is to retrieve only anchortext and text nearby anchortext in its in-linked web pages (the web pages pointing to the target page). This relaxes the relationship between the in-linked web pages and the target web page. For instance, some of the in-linked web pages may have different topics, but the anchortext in the in-linked web pages describes the hyperlinks and therefore the anchortext should help represent the target web page. Nevertheless, using anchortext alone to represent the target page is less powerful than using the full text in the target page [11, 39]. In many cases, the text nearby the anchortext is more discriminant. We can retrieve terms from the anchortext, headings preceding the anchortext, and paragraphs where the anchortext occurs in the in-linked web pages to represent a target page, instead of the local full text. This representation method has been shown to improve the accuracy by 20% compared to the performance of the same system when using full text in the target web page [36]. To make it simple, a window can be used to define the neighborhood of the anchortext, e.g., up to 25 words before and after the anchortext. We define the anchortext and its neighborhood as the extended anchortext [11, 39]. It has been reported that the best web page representation is the combination of the full text in a web page and the extended anchortext in its in-linked web pages, compared to the full text, anchortext, or extended anchortext alone [11, 39].

Whereas the work cited in this subsection provides insights in exploiting information in hyperlinks for web page classification, many questions still remain unanswered. For example, even though most classification tasks over the WebKB university corpus (see Sect. 5.2) reported good performance results, it is not clear whether the performances will increase over a more diverse web page corpus. Reference [114] showed that drawing general conclusions for using hyperlinks in web page classification without examinations over multiple datasets can be seriously misleading.

Link Based Representation

We believe that although hyperlinks are noisy, there are still some linked web pages that show the subjective connections with a target web page. Some

linked web pages may share the same topic or subject as the target web page. Based on this assumption, the link based representation of web pages utilizes the neighborhood of a web page (created by its in-links and out-links) where some neighbors share a same topic. This representation method, making use of topics, differs from the methods in the above subsection where they make use of the text (e.g. anchortext) in the linked web pages.

For link based representation, the topics or class labels of the neighbors of a web page are used to represent the web page. Among all its neighbors, we should choose some of the neighbors who have the same topic as the target web page. One solution is to compute the probability of class i of some neighbors given all neighbors, $Pr(c_i|N_i)$, where c_i is the class i to which some neighbors belonged, and N_i represents the group of all the known class labels of the neighbors [16]. N_i includes the in-links I_i and the out-links O_i, $N_i = \{I_i, O_i\}$. The $Pr(c_i|N_i)$ can be computed according to Bayes theorem as following:

$$Pr(c_i|N_i) = \frac{Pr(N_i|c_i) \cdot Pr(c_i)}{Pr(N_i)} \tag{12}$$

where $Pr(N_i)$ is no difference among classes of neighbors and thus can be removed, and $Pr(c_i)$ is the proportion of training examples that belongs to the class c_i. We assume all classes in N_i are independent, so that

$$Pr(N_i|c_i) = \prod_{\delta_j \in I_i} Pr(c_j|c_i) \cdot \prod_{\delta_k \in O_i} Pr(c_k|c_i) \tag{13}$$

where δ_j is the neighbor in the in-links belonging to class c_j, and δ_k is the neighbor in the out-links belonging to class c_k [16]. The class which maximizes the $Pr(c_i|N_i)$ is selected as the class of the target web page. In order to get class labels for all neighbors, the neighbors should first be classified by employing a text based classifier. This procedure can be performed iteratively to assign the class labels to web pages and finally converge to a locally consistent assignment. This method significantly boosts classification accuracy by reducing error up to 70% from text based classifiers [16].

A similar approach using the neighborhood around a web page is to weight the categories (or classes) of its neighbors by the similarity between the categories of the neighbors and the category of the target web page [18]. The target web page and all its neighbors are first classified by a text based classifier. Then the similarities between the category of the target page and the categories of the neighbors are computed to weight each category of the neighbors. Some neighbors belonging to a same category are grouped. If the sum of similarities for a group exceeds a predefined threshold, the target web page is assigned to the category that the group belongs. This approach increases the accuracy by 35% over a local text based classifier when it was tested using Yahoo directory [18].

Word Sense Representation

The problem with the bag-of-words or bag-of-terms representation is that using word occurrence omits the fact that a word may have different word senses (or meanings) in different web pages or in the same web page. For instance, the word "*bank*" may have at least two different senses, as in the "*Bank*" of America or the "*bank*" of Mississippi river. However using a bag-of-words representation, these two "*bank*" are treated as a same feature. Rather than using a bag-of-words, using word senses to represent a web page can improve web page classification.

By using word senses to be features of a web page vector, a web page d_i is represented as:

$$d_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{iM}) \tag{14}$$

where w_{ij} is the weight of *word sense* s_j $(1 \leq j \leq M)$ in web page d_i. Although this representation has the same format as (1), which is a vector for bag-of-words representation, the indexed features in (14) are word senses rather than words (character patterns) in (1).

Using word senses to be features of a web page, and further be features of each class, is more accurate than using words as features. However the problem is how to find the correct word sense that each word refers. This is also called *word sense disambiguation (WSD)*. Although solutions to WSD are provided in literature [57, 68], semantic analysis has proved to be expensive to be implemented. No work in the literature of classification domain has used this method although initial attempt has been conducted by [19]. To reduce the complexity of semantic analysis, they use a thesaurus to relate a word to its synonyms. This approach does not address WSD; instead it expands word senses to groups of words having similar meanings. It extends the concept of term frequency (TF) to account for word senses. To determine TF of a word, the number of occurrences of its synonyms is also taken into account.

Employing word sense representation is still an open issue and it is not clear whether it will be found more successfully than the statistical approach, e.g. bag-of-terms representation which has been examined to be moderately successful.

2.2 Representations for Genre Based Classification

Web page representations for genre based classification is quite different from representations for subject based classification in that features representing the genre of a web page are different from features representing the subject of a web page. In order to retrieve important features representing the genre of a web page, here we first discuss genres of web pages.

Genre Taxonomy

Genre is necessarily a heterogeneous classification principle. While the concept of genre has a long tradition in rhetorical and literary analysis (Bakhtin

Table 2. Current contents in genre taxonomy

Sources	Genres
Widely recognized genres [117]	Business letter, memo, expense form, report, dialogue, proposal, announcement, thank you note, greeting card, face-to-face meeting system, frequently-asked-questions (FAQ), personal homepage, organizational homepage, bulletin board, hot-list, and intranet homepage
Acorn Project (Yates et al. 1999)	Official announcement, trip report, publication notice, release note, reference, lost and found system, team announcement, traditional memo, electronic memo, dialogue, solicitation, and team report
Online Process Handbook [65]	Welcome page, login page, introduction (user guide, reference), contents page, guide tour, search (search request, result), process knowledge viewer (process compass, process viewer, description, attributes list, tradeoff table, mail), discussion, and options
Web Page Types (Haas and Grams 1998)	Organizational pages (table of contents, site content pages, index pages), documentation (how-to pages, FAQ pages, descriptions of services or products), Text (article, scholarly paper, contract, bibliography, resume, biography, list, copyright), Homepage (organization homepage, personal homepages), Multimedia (non-textual documents), Tools (search tools, order forms, email or comment forms), and Database entry

1986), a number of researches in cultural, rhetorical, and design studies, have recently begun using it to refer to *a typified social action* [6, 13, 69]. Reference [78] defined genre as "a distinctive type of communicative action, characterized by a socially recognized communicative purpose and common aspects of form", taking into account three main aspects of genre: content, form, and purpose. Reference [117] analyzed genres in terms of a number of dimensions of communications, such as What, When, Where, Why, Who, and How. Genre can be analyzed in terms of purposes, contents, participants, timings, places, forms, structural devices, and linguistic elements. How to define each genre of communicative actions is beyond the scope of this chapter. Here we show some genre taxonomy in Table 2, which may help us understand genre based web page classification.

As we can see from Table 2, genre is a subtle and difficult to define notion. One challenge is that it is difficult to know whether we have considered all possible purposes and forms. Without a strong foundational theory of genre to guide us, it is also problematic to construct a classification structure that will accommodate all genres.

The problem that we can solve now is to focus on some genres in which we are most interested and to identify salient features discriminating a genre

from others. Once we have found all features for discriminating genres, we can represent a web page as a feature vector, as we have done for subject based classification. The features most useful for genre based classification can be broadly divided into two classes: *presentation features* and *lexical features* [4, 15, 27, 35, 50, 52, 56, 88, 101].

Presentation Features

There are many features that reflect the way in which a web is presented and these features are called as presentation features [27]. The presentation features can further be divided into syntactic features, text complexity features, character-level features, and layout features, as described in the following.

Syntactic Features

Syntactic features are a set of features directly encoding information about the syntactic structure of the text. Syntactic structure can be expected to be more revealing of genres than simple words counts, as argued in the work by [10]. The idea is that particular document genre will favor certain syntactic constructions. Typical syntactic features are passive count, nominalization count, and counts of the frequency of various syntactic categories (e.g. part of speech tags, such as noun, verb, and adjective) [52]. Previous work on syntactic features was based on automated parsing of the input documents. However, this is a time consuming and unreliable procedure. Whereas this set of features was used in some recent work [10, 50], others tried to avoid this set of features [52, 88].

Text Complexity Features

Text complexity features have been widely used due to their ease of computation and their discriminant ability. These features are based on text statistics, such as the average word length, the long word count, the average sentence length, the number of sentences and paragraphs, and the type-token ratio (the number of different words "type" divided by the number of word tokens). Text complexity correlates in certain ways with genre. For instance, long average word length may indicate documents containing technical terms. The complexity of certain constructs turns out to be captured quite well by these simple measures, such as the metrics for transforming all the counts into natural logarithms used by [52]. In addition to the mean measures, variation measures (e.g. the standard deviation in sentence length) can also be used [27, 52]. Furthermore, other derived measures, e.g. readability grade methods, such as Kincaid, Coleman-Liau, Wheeler-Smith Index, and Flesh Index, may be used as more condensed features. The readability measures are generated based on the above mentioned basic measures by using various transformations to obtain graded representations according to stylistic evaluation parameters.

Character Level Features

A wealth of genre information can be obtained from specific character counts, among which the most prominent features are punctuation marks [27, 88, 101]. For instances, exclamation marks seldom appear in scientific articles; high number of question marks may indicate interviews; high sentence lengths and high counts of commas or semicolon may indicate complicated sentence structures [88]. Example features include counts of quotation marks, exclamation marks, hyphens, periods, apostrophe, acronyms, slash marks, and various brackets.

Other specific characters, such as financial symbols, mathematical symbols, and copyright signs, can also be analyzed [88]. These specific characters hint special genre categories. For instance, the financial symbols, like $ and £, appear more often in the genre of online shopping, while equations frequently occur in technical reports.

Layout Features

Layout features are formed from mark-up tags that can be used to extract information from HTML documents. Mark-up tags provide information, such as the amount of images presented in a given web page, line spacing, and number of tables, equations, and links. Furthermore, mark-up tags can be used to retrieve common layouts for some specific genres like home pages and bulletin boards. Previous work exploiting mark-up tags for genre analysis includes [27, 88], and others.

Lexical Features

Lexical features (or word features) are extracted from analyzing the use of words. This is further described in terms of stop words and keywords as following.

Stop Words

Contrary to the principle of subject based classification, a lot of genre information is conveyed by stop words, such as pronouns or adverbs [4, 88, 101]. A list of stop words is added as features such as *I, you, us, mine, yours, little, large, very, mostly, which, that, and where*. The rational behind their use is that the frequency of such words is presumably not driven by content and hence might be expected to remain invariant for a given genre over different topics. For instance, pronoun frequencies can be used to predict the formality or informality of a document; the word "*should*" may appear frequently in editorials; and the word "*today*" may appear frequently in news [4].

Keywords

For some genre analyses, keywords or key phrases may provide additional cues. For instance, keywords "*resume*", "*education*", and "*experience*" are usually expected to appear in *resume* genre. Keywords are retrieved in the same way as the subject based classification, and are primarily based on how discriminant the words or phases are among different genres. Selecting keywords as features is employed in recent work of [27, 35, 56].

Focusing on the goal of genre analysis, only a small subset of the available features may necessarily be selected for web page classification. How to select the features of high discriminant abilities will be addressed in the next section.

3 Dimensionality Reductions

In subject based classification, a web page may contain hundreds of unique terms and the whole collection of web pages may contain a total of hundreds of thousands of unique terms. If all the unique terms are included as features for representing web pages, the dimension of the feature vectors may be hundreds of thousands. Similarly, the total number of the presentation features and lexical features in the genre based classification may also be too large and superfluous for the genre classification. The high dimensionality of the feature space could be problematic and expensive for computation [102]. On the other hand, it is always expected that we can perform classification in a smaller feature space in order to reduce the computational complexity. Thus, techniques for dimensionality reduction should be employed, whose tasks are to reduce the dimensionality of feature vectors and also to insure that this process will not reduce the accuracy of classification.

Dimensionality reduction is also beneficial in that it tends to reduce the problem of over fitting, i.e. the phenomenon by which a classifier is tuned to the training data, rather than generalized from necessary characteristics of the training data [97]. Classifiers that over fit the training data tend to be very good at classifying the trained data, but are remarkably worse at classifying other data [70]. Some experiments suggested that to avoid over fitting, a number of training examples, roughly proportional to the number of features, is needed [97]. This means that, after dimensionality reduction is performed, over fitting may be avoided by using a smaller number of training examples.

Numerous dimensionality reduction functions, either from information theory or from linear algebra, have been proposed and their relative merits have been experimentally evaluated. These functions can be divided, based on what kinds of features are chosen, into *feature selection* and *feature extraction* [97], as described below.

- **Dimensionality Reduction by Feature Selection:** Feature selection selects a subset of the original feature space based on some criteria. Two

broad approaches for feature selection have been presented in the literature: *the wrapper approach* and *the filter approach* [47]. *The wrapper approach* [47, 53] employs a search through the space of feature subsets. It uses an estimated accuracy for a learning algorithm as the measure of goodness for a particular feature subset. Thus the feature selection is being "wrapped around" a learning algorithm. For example, for a neural network algorithm the wrapper approach selects an initial subset of features and measures the performance of the network; then it generates an "improved set of features" and measures the performance of the network. This process is repeated until it reaches a termination condition (either a minimal value of error or a number of iterations). While some wrapper based methods have encountered some success for classification tasks, they are often prohibitively expensive to run and can break down when a very large number of features are present [54]. For *the filter approach*, feature selection is performed as a preprocessing step before applying machine learning. Thus the method of feature selection is independent to the learning algorithm. The filter algorithm does not incur the high computational cost and is commonly used in classification systems even in a very high feature space. Concerning the advantages of the filter approach over the wrapper approach and the requirement of feature reduction in a very high feature space, the wrapper approaches are omitted in this chapter while several filter approaches are to be discussed and evaluated in Sect. 3.1.

- **Dimensionality Reduction by Feature Extraction:** For feature extraction, the resulting features are not necessary a subset of the original features. They are obtained by combinations or transformations of the original feature space, as discussed in Sect. 3.2.

3.1 Feature Selection

The principle under dimensionality reduction by feature selection is that the features that are more discriminant for some categories from others should be selected. The whole process of feature selection is simplified by the assumption of feature independence. All the features are independently evaluated based on some criteria and a score is assigned to each of them. Then a predefined threshold helps select the best features.

Many criteria have been employed to indicate the discriminant abilities of features. In a broad view, feature selection criteria can be divided into two sets: One set considers only the value denoting a feature occurred in an example, such as the feature selection by using *Document Frequency*, *Mutual Information*, *Cross Entropy*, *and Odds Ratio*, as described in the following subsections. The other set considers all possible values of a feature including both the present and the absent cases, such as the feature selection by using *Information Gain* and *Chi-square Statistic*, as described in the subsequent subsections. This section also describes special methods used for feature

selection in a category hierarchy and concludes with the evaluations of the various feature selection techniques.

Document Frequency

A simple and surprising effective feature selection criterion is document frequency. Document frequency of a feature is the number of documents (or web pages) in which the feature occurs. The features whose document frequencies are less than some predetermined threshold are removed. The basic assumption is that rare features are either not informative for classification or not influential in performance [113]. An important characteristic of document frequency is that it does not require class labels of training examples.

This method has a time complexity of $O(n)$ where n is the number of training examples. Because of its simple computation and good performance, document frequency has been widely used in dimensionality reduction. Reference [113] has shown that by using document frequency as threshold, it is possible to reduce about 89% dimensionality of the feature space and results in either an improvement or no less in classification accuracy when tests were conducted over data from Reuters collection and OHSUMED collection (see Sect. 5.2). Similar good performance was also seen in [38].

Mutual Information

Mutual information [20, 113] is commonly used for word associations and can be used here for feature selection. Within a collection of web pages, the mutual information for each feature is computed and the features whose mutual information is less than some predetermined threshold are removed.

The mutual information between feature f and category (or class) c_i is defined to be

$$MI(f, c_i) = \log \frac{Pr(f \wedge c_i)}{Pr(f)Pr(c_i)} \tag{15}$$

where $Pr(f)$ is the proportion of examples containing feature f over all training examples, $Pr(c_i)$ is the proportion of examples in category c_i over all training examples, and $Pr(f \wedge c_i)$ is the joint probability of feature f and category c_i, which is equal to the proportion of examples in which feature f and category c_i both occur [113]. Equation (15) can be transformed to

$$MI(f, c_i) = \log \frac{Pr(c_i|f)}{Pr(c_i)} \quad \text{or} \quad MI(f, c_i) = \log \frac{Pr(f|c_i)}{Pr(f)} \tag{16}$$

where $Pr(c_i|f)$ is the conditional probability of category c_i given feature f, and $Pr(f|c_i)$ is the conditional probability of feature f given the category c_i. We can estimate $MI(f, c_i)$ by letting A be the number of times f and c_i co-occur, B be the number of times f occurs without c_i, C be the number

of times c_i occurs without f, and N be the total number of examples [113]. Then $MI(f, c_i)$ can be estimated from (15) as:

$$MI(f, c_i) \approx \log \frac{AN}{(A+B)(A+C)} \tag{17}$$

The average and the maximum of mutual information of feature f are computed through all categories as:

$$MI_{\text{avg}}(f) = \sum_{i=1}^{K} Pr(c_i) MI(f, c_i) \tag{18}$$

$$MI_{\max}(f) = \max_{i=1}^{K} \{MI(f, c_i)\} \tag{19}$$

where K is the total number of categories. The time complexity for computing mutual information is $O(MK)$, where M the size of feature space and K is the number of categories [113].

A weakness of mutual information is that favors rare features [113]. From (16), for features with an equal conditional probability $Pr(f|c_i)$, rare features will have a higher score than common features since rare features have small values of $Pr(f)$ at the denominator.

Cross Entropy

Cross entropy used in document classification [54, 55, 74] employs information theoretic measures [25] to determine a subset from an original feature space. It can also be used as a method for feature selection to remove redundant features. Formally, let μ and σ be two distributions over some probability space Ω. The cross entropy of μ and σ is defined as [54, 55]:

$$D(\mu, \sigma) = \sum_{x \in \Omega} \mu(x) \cdot \log \frac{\mu(x)}{\sigma(x)} \tag{20}$$

It provides us with a notion of "distance" between μ and σ [54, 55]. We can transform the above equation to be used for our feature selection as follows. For each feature f and a category set $C = \{c_1, c_2, \ldots, c_K\}$, $Pr(C|f)$ is substituted for μ and $Pr(C)$ for σ. Then, the expected cross entropy $CE(f)$ of feature f is determined as:

$$\begin{aligned} CE(f) &= Pr(f) \cdot D(Pr(C|f), Pr(C)) \\ &= Pr(f) \cdot \sum\nolimits_{i=1}^{K} Pr(c_i|f) \cdot \log \frac{Pr(c_i|f)}{Pr(c_i)} \end{aligned} \tag{21}$$

where $Pr(f)$ is a normalization component [55, 74]. The features whose $CE(f)$ are less than a certain predefined threshold are eliminated. The time complexity for computing cross entropy is $O(MK)$, which is same as that for mutual information. Cross entropy overcomes a weakness of mutual information by favoring common features instead of rare features.

Odd Ratio

Odds ratio is commonly used to indicate feature goodness in information retrieval, where the task is to rank documents according to their relevance [73, 74, 90, 92]. It is based on the idea that the distribution of features in the relevant documents is different from that in the non-relevant documents. Used for feature selection, the odds ratio of feature f and category c_i captures the difference between the distribution of feature f on its positive class c_i and the distribution of feature f on its negative class $\bar{c}_i$. It is defined as:

$$OR(f, c_i) = \log \frac{\text{odds}(f|c_i)}{\text{odds}(f|\bar{c}_i)} = \log \frac{Pr(f|c_i)(1 - Pr(f|\bar{c}_i))}{Pr(f|\bar{c}_i)(1 - Pr(f|c_i))} \tag{22}$$

where $Pr(f|c_i)$ is the conditional probability of feature f given category c_i, and $Pr(f|\bar{c}_i)$ is the conditional probability of feature f given categories $\bar{c}_i$ [72, 74]. From the definition, we can see that $OR(f, c_i)$ will have a high score if feature f appears frequently in the positive training example set c_i and infrequently in the negative training example set $\bar{c}_i$. The average and the maximum odds ratio of feature f are computed through all categories:

$$OR_{\text{avg}}(f) = \sum_{i=1}^{K} Pr(c_i) OR(f, c_i) \tag{23}$$

$$OR_{\max}(f) = \max_{i=1}^{K} \{OR(f, c_i)\} \tag{24}$$

The features whose odds ratios are less than a certain predefined threshold are removed. The time complexity for computing odds ratio is $O(MK)$, where M is the size of feature space and K is the number of categories.

Information Gain

Information gain is frequently employed as a feature goodness criterion in machine learning [70]. The information gain of a feature measures the expected reduction in entropy caused by partitioning the training examples according to the feature. Entropy characterizes the impurity of an arbitrary collection of training examples. Information gain is also called expected entropy loss [39]. More precisely, the information gain $IG(f)$ of a feature f is defined as:

$$IG(f) \equiv \text{Entropy}(D) - \sum_{v \in (f, \bar{f})} \frac{D_v}{|D|} \text{Entropy}(D_v) \tag{25}$$

where D is a collection of training examples, and D_v is a subset of D which is determined by binary feature value v [70]. For instance, D_f is a subset of D in which each example contains feature f, and $D_{\bar{f}}$ is a subset of D in which each example does not contain feature f. Entropy (D) is defined as

$$\text{Entropy}(D) \equiv -\sum_{i=1}^{K} Pr(c_i) \log Pr(c_i) \tag{26}$$

where K is the total number of classes (or categories) in the collection D, and $Pr(c_i)$ is the proportion of examples in category c_i over total training examples [70]. By substituting (26) into (25), the information gain of feature f is

$$\begin{aligned} IG(f) = &-\sum_{i=1}^{K} Pr(c_i) \log Pr(c_i) \\ &+ Pr(f) \sum_{i=1}^{K} Pr(c_i|f) \log Pr(c_i|f) \\ &+ Pr(\bar{f}) \sum_{i=1}^{K} Pr(c_i|\bar{f}) \log Pr(c_i|\bar{f}) \end{aligned} \tag{27}$$

which is equivalent to

$$\begin{aligned} IG(f) = &\, Pr(f) \cdot \sum_{i=1}^{K} Pr(c_i|f) \log \frac{Pr(c_i|f)}{Pr(c_i)} \\ &+ Pr(\bar{f}) \cdot \sum_{i=1}^{K} Pr(c_i|\bar{f}) \log \frac{Pr(c_i|\bar{f})}{Pr(c_i)} \end{aligned} \tag{28}$$

where $Pr(f)$ is the proportion of examples in which feature f is present, $Pr(\bar{f})$ is the proportion of examples in which feature f is absent, $Pr(c_i|f)$ is the conditional probability of category c_i given feature f, and $Pr(c_i|\bar{f})$ is the conditional probability of category c_i given feature f is absent.

The information gain of each feature is computed and the features whose information gain is less than a predetermined threshold are removed. The computation includes the estimation of the conditional probabilities of a category given a feature and the entropy computations. The probability estimation has a time complexity of $O(N)$ where N is the number of training examples. The entropy computation has a time complexity of $O(MK)$ where M the size of feature space and K is the number of categories [113].

It is worthy to note the difference between information gain and cross entropy. As we can see from (21) and (28), the difference between the information gain $IG(f)$ and the cross entropy $CE(f)$ of feature f is that the former makes use of the feature presence and the feature absence, i.e. $IG(f) = CE(f) + CE(\bar{f})$. The similar difference exists between information gain and mutual information. From (28), Information Gain of feature f can be proven equivalent to:

$$IG(f) = \sum_{X \in \{f, \bar{f}\}} \sum_{Y \in \{c_i\}} Pr(X \wedge Y) \log \frac{Pr(X \wedge Y)}{Pr(X)Pr(Y)}$$
$$= \sum_{i=1}^{K} Pr(f \wedge c_i) MI(f, c_i) + \sum_{i=1}^{K} Pr(\bar{f} \wedge c_i) MI(\bar{f}, c_i) \quad (29)$$

This shows that information gain is the weighted average of the mutual information $MI(f, c)$ and $MI(\bar{f}, c)$ (see (15)). Thus information gain is also called average mutual information [113].

Chi-Square Statistic

The Chi-Square (χ^2) statistic measures the lack of independence between feature f and category c_i. The feature goodness metric by χ^2 statistic is defined as [38, 113]

$$\chi^2(f, c_i) = \frac{N[Pr(f \wedge c_i)Pr(\bar{f} \wedge \bar{c_i}) - Pr(f \wedge \bar{c_i})Pr(\bar{f} \wedge c_i)]^2}{Pr(f)Pr(\bar{f})Pr(c_i)Pr(\bar{c_i})} \quad (30)$$

where N is the total number of training examples. The features f with the high values of $\chi^2(f, c_i)$ are thus the more dependent (closely related) with category c_i, which are selected for the purpose of feature selection. We can estimate the value of $\chi^2(f, c_i)$ by letting A be the number of times f and c_i co-occur, B be the number of time f occurs without c_i, C be the number of times c_i occurs without f, and D be the number of times neither c_i nor f occurs, i.e.

$$\chi^2(f, c_i) = \frac{N(AD - BC)^2}{(A + B)(C + D)(A + C)(B + D)} \quad (31)$$

The average and the maximum χ^2 statistic values of feature f over all categories are then computed as following [113]:

$$\chi^2_{\text{avg}}(f) = \sum_{i=1}^{K} Pr(c_i)\chi^2(f, c_i) \quad (32)$$
$$\chi^2_{\text{max}}(f) = \max_{i=1}^{K}\{\chi^2(f, c_i)\} \quad (33)$$

The computation of χ^2 statistic has a time complexity of $O(MK)$, where M is the size of feature space and K is the number of categories. The features with low χ^2 statistic values are removed. Using $\chi^2_{\text{max}}(f)$ for feature selection outperformed using $\chi^2_{\text{avg}}(f)$ as reported in [113] and [38].

Rare features are emphasized by χ^2 in the form of $Pr(f)$ at the denominator. This is not what we expected based on the fact that rare features are not influential in performance as in the case of Document Frequency. To avoid emphasizing rare features, a simplified χ^2 statistics was proposed in [38]:

$$\chi^2(f, c_i) = Pr(f \wedge c_i)Pr(\bar{f} \wedge \bar{c_i}) - Pr(f \wedge \bar{c_i})Pr(\bar{f} \wedge c) \quad (34)$$

It emphasizes the positive correlation between f and c_i (i.e. $Pr(f \wedge c_i)$ and $Pr(\bar{f} \wedge \bar{c_i})$) and de-emphasizes the negative correlation (i.e. $Pr(f \wedge \bar{c_i})$ and $Pr(\bar{f} \wedge c_i)$). Reference [38] showed that the simplified χ^2 outperformed the original χ^2 when feature reduction above 95% is required. However, when below 95% reduction is required, the simplified version is slight inferior to the original one.

Feature Selection in a Category Hierarchy

For a large corpus, we may have hundreds of categories and hundreds of thousands of features. Even by applying the above feature selection techniques, the computational cost for the remaining features may still poses significant limitations. Another approach is to organize categories in a hierarchy (e.g. Yahoo directory) and to divide the classification task into a set of smaller classification tasks, each of which corresponds to some splits in the classification hierarchy. The key insight is that within each of the smaller subtasks and their corresponding smaller subset of feature space, it is possible to use fewer features to differentiate among a smaller number of categories [55]. The approach requires creating a category hierarchy and dividing a classification task into smaller subtasks as described in following.

Creating a Category Hierarchy

The most representative example for web page category hierarchy is provided by Yahoo.com. The top level of the hierarchy consists of 14 main categories (see Fig. 3), such as "Business and Economy", "Computer and Internet", "Society and Culture", "Education", and "Reference". Each main category contains many sub-categories. For instance, "References" main category contains 129 sub-categories, "Education" contains 349 sub-categories, and "Computer and Internet" contains 2652 sub-categories. To represent a category, features are selected from the training web pages taken from the category. To represent the hierarchy, the features in the low level categories are added into the

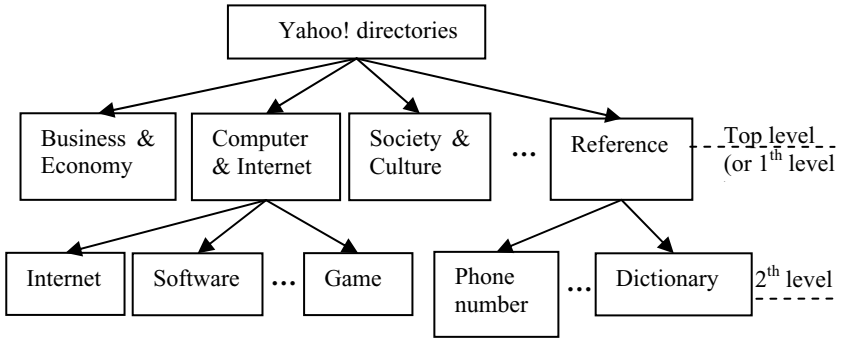

Fig. 3. Top two levels of Yahoo directory

top level categories along the paths in the category tree, which is based on the fact that the low level categories belong to its more general top level categories. When propagating the features upward from the lowest level categories among the paths in the category tree to the root, the weights of the features are reduced and normalized in proportional to the size of its category [73, 82].

Dividing Classification Task into Smaller Subtasks

It is important to note that the key here is not only the use of feature selection, but also the integration of feature selection process within the hierarchical structure. As we can see from Fig. 3, focusing on a main category at the top level, we are only interested in those features that can discriminate its main category from the other 13 main categories. In other words, when selecting features for one category, we only consider features that are discriminant from its sibling categories, which share a same parent node. Among a category and its sibling categories, this smaller feature selection task can be done by applying any feature selection criterions discussed. For instance, Odds Ratio was used to select features in a hierarchy in [73]; Cross Entropy was employed in [55]; and Information Gain was used in [32].

Specially for selecting features among categories, [82] proposed to use a *Uniqueness* criterion, which scored feature f in category c_i (or node c_i) according to the uniqueness in comparing to its sibling categories:

$$U(f, C_i) = \frac{Pr(SubT_{c_i}|T)Pr(f|c_i)}{Pr(f|parent)} \tag{35}$$

where *parent* is the parent node of node c_i, T represents the tree rooted at parent node, $SubTc_i$ is a sub-tree located underneath tree T, and $Pr(SubTc_i|T)$ is a weight factor assigned to node c_i which is equal to the proportion of the size of node c_i over the size of its parent node. The idea of the uniqueness criterion is that if a feature is unique in one node, it is the only source that can be propagated to the parent feature space. By regarding the parent category C as a whole domain $C = \{c_i, c_{\text{siblings}}\}$, the $U(f, c_i)$ can be shown equivalent to

$$U(f, c_i) = \frac{Pr(c_i|C)Pr(f|c_i)}{Pr(f|C)} = \frac{Pr(c_i)Pr(f|c_i)}{Pr(f)} = Pr(c_i|f) \tag{36}$$

From the above equation, it can be noticed that the uniqueness criterion $U(f, c_i)$ is comparable to the mutual information $MI(f, c_i)$ (16); the main difference is that the uniqueness criterion removes the factor $Pr(c_i)$ at the denominator of $MI(f, c_i)$. Since [82] only selected features whose uniqueness score is 1 for a category c_i, the $Pr(c_i|f)$ alone contains enough information to be a criterion for their feature selection.

Evaluations of Feature Selection Techniques

While many feature selection techniques have been proposed, thorough evaluations have rarely carried out for classification in a large feature space. The most impressive work on evaluating some feature selection techniques can be found in [113] and [73, 74]. To assess the effectiveness of feature selection techniques, two classifiers, a k-nearest-neighbor (kNN) classifier and a Linear Least Squares Fit mapping (LLSF) classifier (described Sect. 4), were employed in [113]. The classifiers learned over two text collections, the Reuters collection and the OHSUMED collection.

Reference [113] found Information Gain and χ^2 statistics most effective in their experiments compared to Document Frequency, Mutual Information, and Term Strength, which is omitted in this chapter. Using Information Gain with a kNN classifier on the Reuters collection not only achieved up to 98% reduction of feature space but also yielded improvements in classification accuracy. They found strong correlation between Document Frequency, Information Gain, and χ^2 statistics. This suggests that Document Frequency, the simplest method with the lowest cost in computation, can be reliably used instead of Information Gain and χ^2 statistics when the computations of the later criterions are too expensive [113]. In contrast, Term Strength was not competitive at high percentage of feature reduction. Mutual Information had relatively poor performance due to its use of feature presence only and its bias toward favoring rare features. However, the effect of favoring rare features can be compensated by first removing those rare features and Mutual Information showed no significant performance difference among other feature selection techniques in [92].

Reference [73, 74] showed that the best performing feature selection methods were Odds Ratio among eleven feature selection methods tested. They employed a Naïve Bayes classifier for learning over web pages derived from Yahoo directory. The next group of methods that achieve good results favors common features (e.g. Cross Entropy). Mutual Information differs from Cross Entropy only in favoring rare features and achieved worse results than Cross Entropy. The worst feature selection method was Information Gain, which on the other hand achieved the best performance in experiments by [113].

The differences in evaluation results reflect the differences in classification algorithms and test domain used. We can observe that Information Gain makes use of feature presence and feature absence while Odds Ratio, Cross Entropy and Mutual Information only consider information of feature presence. In experiments by [73, 74], the data collection from Yahoo directory has unbalanced class distribution and highly unbalanced feature distribution. They observed that the prior probability of a feature in a web page, $Pr(f)$, is rather small. Most of the features selected by Information Gain are features having high absent feature value $Pr(\bar{f})$. If $Pr(\bar{f})$ is much larger than $Pr(f)$, the high value of Information Gain in most cases means that the second part $Pr(\bar{f}) \sum_{i=1}^{k} Pr(c_i|\bar{f}) \log \frac{Pr(c_i|\bar{f})}{c_i}$ of the Information Gain formula (28) is high.

In other words, knowing that feature f does not occur in a web page brings useful information about the category of the web page. The problem is that a classification relied mostly on the absence of features is usually more difficult and requires larger feature space than a classification relied on feature presence [73, 74]. In contrast, Odds Ratio (22) favors features from positive examples (high $Pr(f|c_i)$). Thus, Odds Ratio outperformed all other feature selection techniques in [73, 74]. Another reason for the differences of evaluation results is that the classification algorithm used by [73, 74] is a Naïve Bayes classifier, which considers only features that occur in training examples. This means that the selected features should be the features that will probably occur in new web pages to be classified [73, 74].

The common conclusions made by [113] and [73, 74] include the followings. When choosing a feature selection method, both classification algorithm and data domain should be taken into considerations. A rather small feature subset should be used since it gives either better or as good results as large feature space. A simple frequency count of features, such as document frequency, achieves very good results. Feature selection methods favoring frequent features achieve better results than methods favoring rare features. This indicates that frequent features are informative for classification.

One limitation of using feature selection techniques as described in this subsection is the inability to consider co-occurrence of features. Two or more features individually may not be useful, but when combined may become highly effective. This limitation is addressed by using feature extraction.

3.2 Feature Extraction

Feature extraction synthesizes a set of new features from the set of original features where the number of new features is much smaller than the number of original features. The rationale for using synthetic, rather than naturally occurring, features is that the original features may not form an optimal dimension for web page representation [97]. Methods for feature extraction aim at creating artificial features that do not suffer the problems of polysemy, homonymy, and synonymy present in the original features. Several approaches have been reported and successfully tested [96, 105, 108]. In the following, we describe two approaches: latent semantic indexing and word clustering.

Latent Semantic Indexing

Latent semantic indexing (LSI) is based on the assumption that there is an underlying or latent semantic structure in the pattern of features used across the web page corpus and some statistical techniques can be used to estimate the structure [9, 26, 96, 105, 108]. LSI uses singular value decomposition (SVD), which is a technique related to eigenvector decomposition and factor analysis.

The main idea in latent semantic indexing (see also [9]) is to explicitly model the interrelationships among features by using SVD and to exploit this

to improve classification. The process begins by constructing a M features by N documents matrix called A, where each entry represents the weight of a feature in a document, i.e.,

$$A = (a_{ij}) \tag{37}$$

where, a_{ij} is the weight of feature i ($1 \leq i \leq M$) in document j ($1 \leq j \leq N$). Since not every feature normally appears in every document, the matrix A is usually sparse. The singular value decomposition (SVD) of matrix A is given by:

$$A_{M\times N} = U_{M\times R} \sum_{R\times R} V^T_{R\times N} \tag{38}$$

where R is the rank of A ($R \leq \min(M, N)$); U and V have orthogonal unitlength columns ($U^TU = I$; $V^TV = I$); and Σ is the diagonal matrix of singular values of A ($\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_R)$) which are the nonnegative square roots of the eigenvalues of AA^T. Table 3 outlines the definition of the terms.

Table 3. Interpretation of SVD components within LSI

A = matrix of M features × N documents	M = number of features
U = feature vectors	N = number of documents
Σ = singular values	R = rank of A
V = document vectors	k = number of factors (k highest singular values)

If the singular values in Σ are ordered by size, the k largest may be kept and the remaining smaller ones set to zero, i.e. $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, \ldots, \sigma_R)$, where $\sigma_i > 0$ for $1 \leq i \leq k$ and $\sigma_i = 0$ for $i > k$. The product of the resulting matrices is a matrix A_k which is an approximation to A with rank k, i.e.

$$A_k = U_k \sum_k V^T_k \tag{39}$$

where $k \ll M, \Sigma_k$ is obtained by deleting the zero rows and columns of Σ, and U_k and V_k are obtained by deleting the corresponding rows and columns of U and V (showed in Fig. 4).

The resulting A_k captures most of the underlying structure in the association of features and documents in A. The three matrices U_k, Σ_k, and V_k reflect a breakdown of the original feature-document relationships into linearly-independent vectors or factors. The use of k factors or k-largest singular triplets is equivalent to approximate the original matrix. In addition, a new document d can be represented as a vector in k-dimensional space as:

$$\tilde{d} = d^T U_k \sum\nolimits_k^{-1} \tag{40}$$

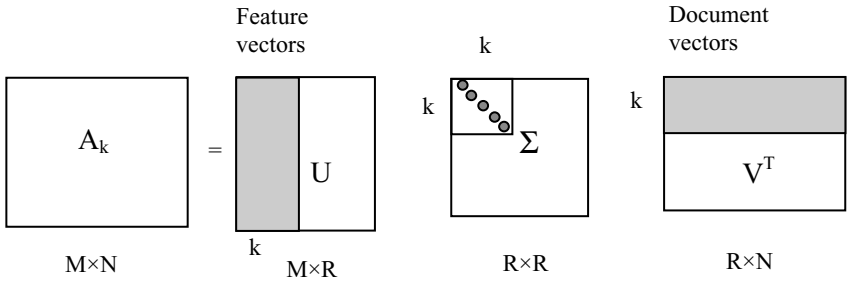

Fig. 4. Graphical interpretation of the matrix A_k

where $d^T U_k$ reflects the sum of k-dimensional feature vectors and Σ_k^{-1} weights the separate dimensions [9, 40].

It is difficult to interpret the new smaller k-dimensional space although it is assumed to work well in bringing out the latent semantic structure of feature-document matrix. An example provided in [9] may help us understand the new space: consider the words *car*, *automobile*, *driver* and *elephant*. The words *car* and *automobile* are synonyms, *driver* is a related concept and *elephant* is unrelated. The words *car* and *automobile* will occur with many of the same words, such as *motor, model, vehicle, chassis,* and *engine*. Thus, they will have similar representations in the k-dimensional space. The context for *driver* will overlap to a lesser extent, and those for *elephant* will be quite dissimilar. This relates to the fact that features which occur in similar documents will be near each other in the k-dimensional space even if these features do not co-occur in the same documents. This farther means that two documents may be similar even if they do not share same keywords.

Word Clustering

Word clustering aims at grouping words or phrases into clusters based on their semantic relatedness. The resulting clusters or their centroids are then used in place of the original groups of words or phases [97]. A word clustering method can also be interpreted as a method for converting the original representation of a document into a new representation that has a much smaller dimensionality than the original one.

One example of this approach is the work by [63], who view semantic relatedness between words in terms of their co-occurrence and co-absence within training documents. By using the criteria in the context of a hierarchical clustering algorithm, they witnessed only a marginal improvement, which may not by conclusive due to the small size of their experiments. Other works [5, 28, 100], such as distributional clustering of words, has achieved improvements over feature selection methods in terms of classification accuracy, especially at lower number of features. Additional related research could be found in [7, 23, 28, 31, 34, 44, 51, 66, 87, 99, 106, 116].

4 Web Page Classifiers

After features of training web pages have been selected to form concise representations of the web pages, various machine learning methods and classification algorithms can be applied to induce the classification function f' as defined in Sect. 1.2 or to induce representations for categories from the representations of training web pages. When a new web page is to be classified, the classifiers use the learned function to assign the web page to categories.

In what follows we discuss the state of the art classifiers in terms of web page classification. We partition classifiers appeared in literature into profile, rule learning, direct example, and parameter based classifiers, where the first three are called non-parametric approaches and the last one is called parametric approach [31, 58]. Here first the definitions and some general notations are given. A web page is usually represented by a vector $d_i = \{w_1, w_2, \ldots, w_M\}$, where each w_i is the weight of a feature of the web page and M is the size of feature space. Predefined categories are denoted by a set $C = \{c_1, c_2, \ldots, c_K\}$, where each c_i is a category label and there are K categories. Training examples consists of N web pages represented by vectors $d_1, d_2, \ldots d_N$, which are tagged with true category labels $y_1, y_2, \ldots y_N$, respectively. Let N_j be the number of training web pages for which the true category label is c_j. In general, the classification process consists of a training phase and a testing phase: during the training phase, training examples are used to train the classifiers; during the testing phase, the classifiers are applied to classify web pages. Some rule learning classifiers also consist of a validation phase for optimizing the rules.

4.1 Profile Based Classifiers

For profile based classifiers, a profile (or a representation) for each category is extracted from a set of training web pages that has been predefined as examples of the category. After training all categories, the classifiers are used to classify new web pages. When a new web page is to be classified, it is first represented in the form of a feature vector. The feature vector is compared and scored with profiles of all the categories. In general, the new web page may be assigned to more than one category by thresholding on those webpage-category scores and the thresholding methods used can influence the classification results significantly [114]. In the case where a web page has one and only one category, the new web page is assigned to the category that has the highest resulting score. Examples of classifiers using this approach are Rocchio classifier, Support Vector Machine, Neural Network classifier, and Linear Least Square Fit classifier, each of which is reviewed in the followings.

Rocchio Classifier

Rocchio algorithm is a classic algorithm for document routing and filtering in information retrieval [14, 43, 91]. Rocchio algorithm employs TFIDF feature

weighting method to create a feature vector for each document. Learning (or training) is achieved by combining feature vectors into a prototype vector $\vec{c}_j$ for each class c_j. The normalized feature vectors of the positive examples for class c_j and those of the negative examples are first summed up. Then, the prototype vector $\vec{c}_j$ is calculated as a weighted difference as:

$$\vec{c}_j = \alpha \frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|C - c_j|} \sum_{\vec{d} \in C - c_j} \frac{\vec{d}}{\|\vec{d}\|} \tag{41}$$

where α and β are parameters that adjust the relative impact of positive and negative training examples, $|c_j|$ is the number of elements in set c_j and $||\vec{d}||$ is the length of vector $\vec{d}$ [45]. The resulting set of prototype vectors, one vector for each class, represents the learned model.

Using cosine as a similarity metric and letting $\alpha = \beta = 1$, Rocchio shows that each prototype vector maximizes the mean similarity of the positive training examples minus the mean similarity of the negative training examples [45], i.e.

$$\frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \cos(\vec{c}_j, \vec{d}) - \frac{1}{|C - c_j|} \sum_{\vec{d} \in C - c_j} \cos(\vec{c}_j, \vec{d}) \tag{42}$$

After obtaining a prototype vector $\vec{c}_j$ for each of the predefined categories C, the classifier is then used to classify a new document d'. To classify document d' the cosine similarity measures of each prototype vectors $\vec{c}_j$ with $\vec{d}'$ are calculated. The document d' is assigned to the class with which $\vec{d}'$ has the highest cosine metric:

$$H(d') = \arg\max_{c_j \in C} (\cos(\vec{c}_j, \vec{d}')) = \arg\max_{c_j \in C} \frac{\vec{c}_j}{\|\vec{c}_j\|} \cdot \frac{\vec{d}'}{\|\vec{d}'\|} \tag{43}$$

where $\arg\max f(x)$ returns the argument x for which $f(x)$ is maximum and $H(d')$ is the category to which the algorithm assigns document d' [45].

Note that the cosine similarity measure is nonlinear. However, this model can be recast as linear classification by incorporating its length normalization into each of the elements of its weight vector:

$$\vec{c}_j \leftarrow \frac{\vec{c}_j}{\|\vec{c}_j\|} \quad \text{and} \quad \vec{d}' \leftarrow \frac{\vec{d}'}{\|\vec{d}'\|} \tag{44}$$

Thus $H(d')$ is transformed to be:

$$H(d') = \arg\max_{c_j \in C} \vec{c}_j \cdot \vec{d}' \tag{45}$$

Previous work using the Rocchio algorithm in text classification could be found in [22, 61, 62, 86, 109]. More interesting, [45] proposed a probabilistic analysis of the Rocchio algorithm, which he called PrTFIDF classifier and showed improvement compared to the original Rocchio algorithm.

Support Vector Machine

Support Vector Machines (SVMs) have shown to yield good performance on a wide variety of classification problems, most recently on text classification [27, 33, 46, 62, 80, 103, 112]. They are based on Structural Risk Minimization principle from computational learning theory [24, 104]. The idea of structural risk minimization is to find a hypothesis h which is defined as the decision function with maximal margin between the vectors of positive examples and the vectors of negative examples [105], see Fig. 5. It was shown that if the training set is separated without errors by h, the expectation value of the probability of committing an error on a test example is bounded by a very small number, i.e. 0.03 [24].

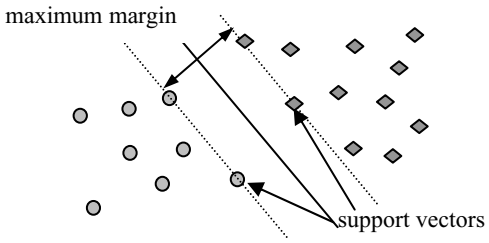

Fig. 5. Linear support vector machine. This figure shows an example of a simple two-dimensional problem that is linearly separable. The diamonds in the figure represent positive examples and the circles represent negative examples. SVM finds the hyperplane h (denoted by the *solid line*), which separates the positive and negative training examples with a maximum margin that is the distance between the two parallel *dashed lines*. The examples closest to the hyperplane are called Support Vectors (indicated in the figure with *arrows*). In other words, SVM finds h that maximizes distances to Support Vectors [104]

In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with a maximum margin (see Fig. 5). Let $D = \{(y_i, d_i)\}$ denote the training set, and $y_i \in \{+1, -1\}$ be the classification of a document vector $\vec{d}_i$, where $+1$ indicates a positive example and -1 indicates a negative example of a given category. SVM learns linear threshold functions [46] of the type:

$$h(\vec{d}) = \text{sign}(\vec{w} \cdot \vec{d} + b) = \begin{cases} +1, & \text{if} \quad \vec{w} \cdot \vec{d} + b > 0 \\ -1, & \text{otherwise} \end{cases} \tag{49}$$

where $h(\vec{d})$ represents a hypotheses given $\vec{d}$, and $\vec{w}$ represents a weight vector, while b is a scalar to be defined in (54). Finding the hyperplane having the maximum margin can be translated into the following optimization problem [104]:

$$\text{Minimizes}: \quad \|\vec{w}\| \tag{50}$$

so that:

$$\forall i : y_i[\vec{w} \cdot \vec{d}_i + b] \geq 1 \tag{51}$$

where $||\vec{w}||$ denotes the Enclidean length of a weight vector $\vec{w}$. The constraint expressed in (51) requires that all training examples are classified correctly. In order to solve the above optimization problem, Lagrange multipliers are used to translate the problem into an equivalent quadratic optimization problem [46, 104]:

$$\text{Minimize}: -\sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{d}_i \cdot \vec{d}_j \tag{52}$$

so that:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \forall i : \alpha_i \geq 0 \tag{53}$$

For this quadratic optimization problem, efficient algorithms can be used to find the global optimum [80]. The result of the optimization process is a set of coefficients α_i^* for which (52) is minimized [46]. These coefficients can be used to construct the hyperplane as follows:

$$\vec{w} \cdot \vec{d} = \left(\sum_{i=1}^{N} \alpha_i^* y_i \vec{d}_i \right) \cdot \vec{d} \quad \text{and} \quad b = \frac{1}{2}(\vec{w} \cdot \vec{d}_+ + \vec{w} \cdot \vec{d}_-) \tag{54}$$

From the above equation [46], we can see that the resulting weight vector, $\vec{w}$, is constructed as a linear combination of the training examples. Only the training vectors, for which the coefficient α_i is greater than zero, contribute the combination. These vectors are called Support Vectors, as shown in Fig. 5. To calculate b, an arbitrary support vector $\vec{d}_+$ from positive examples and one $\vec{d}_-$ from negative examples are used [46].

Once the weight vector for each of the given categories is obtained, a new document d can be classified by computing $\vec{w} \cdot \vec{d} + b$ in (49), where $\vec{w}$ is the learned weight vector of a given category, and $\vec{d}$ is the vector representing the new document. If the value is larger than 0, then the new document is assigned to this category.

Neural Network Classifier

Neural network (NNet) approaches to text classification were evaluated by many researchers, such as [27, 52, 76, 96, 105]. Reference [105] employed a perceptron approach (without a hidden layer) and a three-layered neural network (with a hidden layer), while [76] evaluated only perceptrons. Since neural networks are among the top ranking classifiers [62, 109], Perceptrons and Least Mean Square rules will be briefly described in terms of web page classification.

Perceptrons

For web page classification, a perceptron is used for a given category. It takes a feature vector representing a web page as input, calculates a linear combination of the features of the input vector, then outputs a $+1$ if the result is greater than a threshold (that is automatically learned during the training process) or -1 otherwise, which indicates whether the web page belongs to the category or not, respectively. For illustration (see also [70]), we write a perceptron function as

$$o(\vec{d}) = \text{sgn}(\vec{w} \cdot \vec{d} + b) = \text{sgn}\left(\sum_{i=1}^{M} w_i d_i + b\right) \tag{55}$$

where $\vec{w} = (w_1, w_2, \ldots, w_M)$ is an M-dimensional weight vector and $\vec{d} = (f_1, f_2, \ldots, f_M)$ is an M-dimensional input vector representing a web page, in which each element f_i is the ith feature value, b is a threshold, and the function sgn is defined as

$$\text{sgn}(y) = \begin{cases} +1, & \text{if} \quad y > 0 \\ -1, & \text{otherwise} \end{cases} \tag{56}$$

To simplify notation, we transform the vectors $\vec{w}$ and $\vec{d}$ to be $M + 1$ dimensional vectors by adding $w_0 = b$ and $f_0 = 1$. This allows us to rewrite the above (55) and (56) as

$$\text{sgn}(\vec{w} \cdot \vec{d}) = \begin{cases} +1, & \text{if} \quad \vec{w} \cdot \vec{d} > 0 \\ -1, & \text{otherwise} \end{cases} \tag{57}$$

We can view the perceptron as a hyperplane decision surface in an $M+1$-dimensional space [70]. The perceptron outputs $+1$'s for all positive examples on one side of the hyperplane and outputs -1's for negative examples on the other side. The equation for this decision hyperplane is $\vec{w} \cdot \vec{d} = 0$. Sets of web pages that can be separated by the hyperplane are called linearly separable. We can also notice that the hyperplane produced by the perceptron does not require the maximum margin between the vectors of the two classes which is required by the SVM.

When training a perceptron, the weights $w_1, w_2, \ldots, w_M$ for the perceptron are adjusted based on the training examples. The learning process begins with a setup of random weights, then iteratively applies the perceptron to each training example, and modifies the weights whenever the perceptron misclassifies an example. The weights are modified at each step according to a training rule, which revises the weights w_i in associated with the inputs f_i according to the following learning rule:

$$w_i \leftarrow w_i + \eta(y_i - o_i) f_i \tag{58}$$

where η is the learning rate, y_i is the actual class label (+1 or −1), o_i is the output generated by the perceptron [70]. This learning process iterates through the training examples as many times as needed until the perceptron classifies most training examples correctly as a result of converging toward a set of weights. The learned weight vector is then used to classify new web pages.

Least Mean Square Rule

Least Mean Square (LMS) training rule, also known as Widrow-Hoff rule, was employed and showed good performance for text classification [61, 109]. Although the perceptron rule finds a successful weight vector when the training examples are linearly separable, it may fail to converge if the examples are not linearly separable. LMS training rule is designed to overcome this difficulty.

LMS training rule is best understood by considering the task of training a perceptron without the threshold (see also [70]); that is, a linear unit (without threshold) for which the output o is given by

$$o(\vec{d}) = \vec{w} \cdot \vec{d} \tag{59}$$

The basic principle under LMS rule is to minimize the error function $E_i(\vec{w})$ defined for each individual training example d_i:

$$E_i(\vec{w}) = \frac{1}{2}(y_i - o_i)^2 \tag{60}$$

The negative of the gradient of E with respect to the weight vector $\vec{w}$ gives the direction of steepest decrease [70]. Thus the weight update rule for incremental gradient descent is:

$$\Delta\vec{w}_i = -\eta\frac{\partial E}{\partial w_i} = \eta(y_i - o_i) \cdot f_i \tag{61}$$

It can be seen that the expression of LMS rule appears to be identical with the perceptron weight update rule (58). These two training rules are different in terms of that for LMS rule the output o refers to the linear unit output $o(\vec{d}) = \vec{w} \cdot \vec{d}$ whereas for perceptron rule the output o refers to the threshold output $o(\vec{d}) = \text{sgn}(\vec{w} \cdot \vec{d})$. Similar to a perceptron, the learned weight vectors can then be used to classify new web pages.

Linear Least Squares Fit Classifier

Linear Least Squares Fit (LLSF) is a successful classifier for document classification [62, 108, 109, 111, 112, 119]. It is associated with a linear regression model [110, 111]. The training data are represented using two matrices D and C, where D is a document matrix and C is a category matrix. An element of matrix D is the weight of a feature in a corresponding document, and an

element of matrix C is the weight ($+1$ or -1) of a category in a corresponding document, where $+1$ indicates belonging to the category and -1 for not belonging to the category. The LLSF problem is defined as finding a matrix W that minimizes the squared error of the Frobenius matrix norm of $WD - C$ [108]; in other words, the objective is to find W that minimizes the squared error in the mapping from training documents to their categories. The solution of W is then used to transform an arbitrary document, represented by a feature vector $\vec{d}$, to a category vector $\vec{c}$ by computing $W\vec{d} = \vec{c}$ [110]. The elements of vector $\vec{c}$ are interpreted as the relevance scores of categories with respect to document $\vec{d}$. By sorting the scores, the most relevant categories are obtained, which are the output of the LLSF mapping [110].

A conventional method for solving a LLSF problem employs a Singular Value Decomposition (SVD) of the input matrix D as a part of the computation [40, 110]. Compute an SVD of D, yielding $D = USV^T$ (see Sect. 3.2). Then compute the mapping function $W = CVS^{-1}U^T$ [110]. Because of the high time complexity of computing SVD, dimensionality of the document vectors must be reduced before LLSF is employed.

It is worth noting that the linear classifiers (such as Perceptrons, LMS rule, LLSF, and SVM) do not explicitly construct feature combinations but use the context implicitly [110, 111]. For instance, the classification function in LLSF is sensitive to weighted linear combinations of features that co-occur in training examples [109]. This is a fundamental difference from the classification methods based on the assumption of feature independence, such as Naïve Bayes classifier (see Sect. 4.4).

4.2 Rule Learning Based Classifiers

The one of the most expressive and human readable representations for learned hypotheses is sets of if-then rules. The most important property of rule induction algorithms is that they allow the interrelationships of features to influence the outcome of the classification, whereas some other classification schemes, e.g. Naïve Bayes, assume the features as independent components. Hence, rule learning based classifiers are context sensitive classifiers [3, 22].

In general, for rule learning based classifiers, the training web pages for a category are used to induce a set of rules for describing the category. A web page to be classified is used to match the conditions of the rules. The matched rules predict the class for the web page based on the consequents of the rules. In this section we discuss three successful representatives of the rule learning based classifiers: Disjunctive Normal Form (DNF) rule, Association rule, and Decision tree. Each of them uses a different rule induction algorithm but they are theoretically equivalent because each learned model is a disjunction of conjunction rules.

Disjunctive Normal Form Rule

An example of a classifier using Disjunctive Normal Form (DNF) rules is RIPPER [21, 22], which performs quite well for document classification [109]. DNF classifiers can be interpreted as learning a disjunction of contexts, each of which defines a conjunction of simple features. For instance, the context of a feature f_1 in document d_i is a conjunction of the form:

$$f_1 \text{ and } f_2 \text{ and } f_3 \ldots \text{and } f_k \tag{62}$$

where $f_j \in d_i$ for $j = 1$ to k. This means that the context of feature f_1 consists of a number of other feature $f_2, f_3, \ldots f_k$ that must co-occur with f_1. These features may occur in any order and in any location in the document. The classifier constructed by RIPPER is a set of rules that can be interpreted as a disjunction of conjunctions. For instance,

Document d_i belongs to category "*Louisiana*" IF AND ONLY IF ("*Louisiana*" appears in d_i AND "*Cajun*" appears in d_i) OR ("*New Orleans*" appears in d_i AND "*French Quarter*" appeared in d_i)

The classification of new documents is based on the learned rules that test for the simultaneous presence or absence of features. The rule learning process consists of two main stages. The first stage is a greedy process that constructs an initial rule set. The second stage is an optimization process that attempts to further improve the compactness and accuracy of the rule set. These two main stages are briefly discussed as follows.

The first stage is based on a variant of the rule learning algorithm called incremental reduced error pruning (IREP) [37]. To construct a rule (see also [22]), the uncovered examples are randomly partitioned into two subsets, a growing set containing two-third of the examples and a pruning set containing the remaining one-third. The algorithm will first grow a rule by repeatedly adding conditions and then prune the rule. In the procedure of growing a rule, at each step i, a single condition is added to the rule r_i, producing a more specialized rule r_{i+1}. The added condition is the one that yields the largest information gain [85] for r_{i+1} relative to r_i, which is defined as

$$\text{Gain}(r_{i+1}, r_i) = T_{i+1}^{+} \cdot \left(\log_2 \frac{T_{i+1}^{+}}{T_{i+1}^{+} + T_{i+1}^{-}} - \log_2 \frac{T^{+}}{T_i^{+} + T_i^{-}} \right) \tag{63}$$

where T_i^{+} is the number of positive examples and T_i^{-} is the number of negative examples in the growing set covered by rule r_i [22]. The addition of new conditions continues until the rule covers no negative examples in the growing set or until no condition results in a positive information gain.

After growing a rule, as described in [22], the rule is then pruned or simplified. At each step, the algorithm considers deleting a condition from a rule. It chooses a condition for deletion that maximizes the function

$$f(r_i) = \frac{U_{i+1}^{+} - U_{i+1}^{-}}{U_{i+1}^{+} + U_{i+1}^{-}} \tag{64}$$

where U_{i+1}^{+} is the number of positive examples and U_{i+1}^{-} is the number negative examples in the pruning set covered by the pruned rule [85]. After pruning conditions, the rule is added into the initial rule set and the examples covered by the condition are removed.

The second stage by RIPPER is an optimization procedure in which it optimizes each rule in the current rule set in order to avoid the over fitting problem. For each rule r, two additional rules are constructed: a revised rule r' and a new replacement rule r''. The revised rule r' is constructed by greedily adding literals to r. The resulting r' is then pruned to minimize error of the current rule set. The new replacement rule r'' is constructed from an empty rule by growing. After growing, r'' is also pruned to minimize error of the current rule set. The final step is to select r, r' or r''. This selection is based on minimum description length principle. The rule that has the smallest description length and has no less descriptive power than the other rules is selected [22].

After the optimization, the current rule set may not cover all positive examples. For uncovered positive examples, additional rules are constructed and added to cover them. The optimization step is then repeated, occasionally resulting in further improvements in the current rule set. Previous experiments show that two rounds of the optimization are usually sufficient [22].

Association Rule

An example of a classifier that uses association rules is proposed by [118], who applied association rule mining in building a document categorization system. Association rules mining [2, 42] is a data mining task aiming at discovering relationships between items in a dataset. This approach has the advantage of fast training and has the performance comparable to most well known document classifiers.

For document classification, as described in [118], a set of rules can be used to encode each category. A rule is represented in "$F => c$" formal, where F is a conjunction (such as $f_1 \wedge f_2 \wedge f_3$) of features extracted from a set of training documents, which are taken from the same category c. Once a set of rules is discovered for each category, the rule set can be used to classifier new documents.

The process for discovering a set of rules for category c begins by counting the number of occurrences (or frequency) of each feature in the set D of training documents for the category. It then selects features that have concurrent frequency larger than a threshold called support. The selected features are paired to form 2-item features. Then, the process counts frequency of each of the 2-item features, and selects the 2-item features that have concurrent

frequency larger the support threshold. The resulting 2-item features are combined with 1-item features to form 3-item features, and so on. This process is repeated until k-item features are selected, where k is a predefined number. The set of k-item features are transformed to a set of rules, each of which is in form of $(f_1 \wedge f_2 \wedge \ldots f_k) \Rightarrow c$. Only some of the rules are used for classification. These rules are selected based on a confidence criterion. The confident of a rule $(f_1 \wedge f_2 \wedge \ldots f_k) \Rightarrow c$ can be defined here as the proportion of documents which belong to category c over those documents each of which contains features f_1 and f_2 and $\ldots$ f_k. A threshold of 70% for the confidence criterion was used in [118].

Decision Tree

Classifiers using decision tree are considered as rule learning based classifiers since a decision tree can be converted into a disjunction of conjunction rules (same as DNF). Decision trees have been employed for document classifications [27, 35, 60, 75, 108]. However, this approach contains no special mechanism to handle the large feature sets encountered in document classification and probably accounts for its relatively poor performance in experiments by [60, 75, 108].

To classify a document d using a decision tree, the document vector $\vec{d}$ is matched against the decision tree to determine to which category the document d belongs [1, 12, 70, 83, 84]. The decision tree is constructed from training documents. A popular approach is CART algorithm [12].

CART approach, as described in [1], constructs a binary decision tree (e.g. Fig. 6) from a set of training documents that are represented as feature

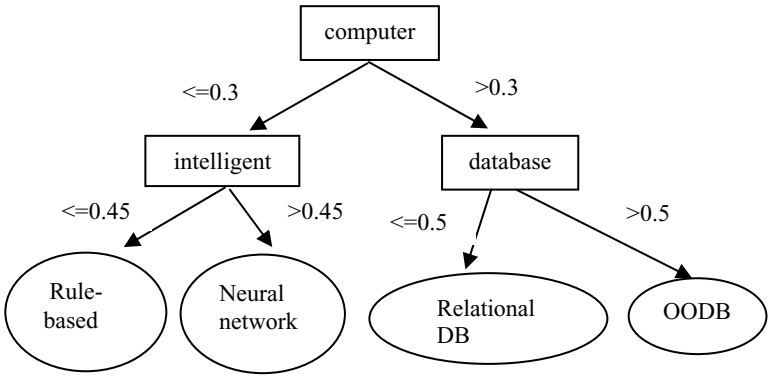

Fig. 6. A decision tree. Each node, except the leaf nodes, in the tree represents a feature, each branch descending from the node corresponds to one of the possible feature values (e.g. the TFIDF value of a term), and each leaf node is a class label. A new test document is classified by traversing it through the tree to the appropriate leaf node, and then returned the category associated with the leaf node

vectors. At each step, a feature is selected from the set of feature vectors and is used to split the set of feature vectors into two subsets. A measure called *diversity* is used to determine which feature to select. The best selection is done by maximizing:

$$diversity(before\_split) - [diversity(left\_child) + diversity(right\_child) \quad (65)$$

One of the commonly used *diversity* measures is entropy (see (26)) and another one is Gini Entropy (*GE*) which is used in CART. The Gini Entropy of a node t is defined as

$$GE(t) = 1 - \sum_{j=1}^{K} Pr(c_j|t) \quad (66)$$

where K is number of categories, and $Pr(c_j|t)$ is the probability of a training example being in class c_j that falls into node t. $Pr(c_j|t)$ can be estimated by

$$Pr(c_j|t) = \frac{N_j(t)}{N(t)} \quad (67)$$

where $N_j(t)$ is the number of training examples of class c_j at node t and $N(t)$ is the total number of training examples at node t.

To select a feature for a node (e.g. Fig. 6), each feature in all training feature vectors is evaluated using (65) and (66). The feature resulting in the maximum value in (65) is selected and used to split the set of training feature vectors. This process is repeated until no training feature vectors can be partitioned any further. Each training feature vector can then be used to traverse the resulting binary tree from root to a leaf node. The resulting leaf node is assigned a category label of the training feature vector.

After building the initial binary tree using the above algorithm, the resulting tree usually overfits the training documents and is not effective for classifying new documents. Thus, the initial tree is pruned to remove the branches that provide the least additional predictive power per leaf. The result of each pruning is a new tree. The final task is to select a tree that will best classify new documents. For this purpose, a new set of labeled documents are used as the validation set of documents. Each of the candidate trees is used to classify the validation set. The tree that has the highest accuracy is selected as the final tree.

Another well known decision tree algorithm is C4.5 [84]. It differs from CART in that it produces tree with varying numbers of branches per node while CART produces a binary tree. It also uses a different approach to prune the tree by converting the learned tree into an equivalent set of rules, which are the result of creating one rule for each path from the root to a leaf node. Each feature along the path becomes a condition while the category label at the leaf node becomes the consequent. For instance, the leftmost path of the tree in Fig. 6 is translated into the rule: *IF (computer <=0.3) and (intelligent*

<=0.45)THEN class label is "Rule-based". Next, each such rule is pruned by removing any condition, whose removal does not worsen the estimated accuracy (see also [70]).

4.3 Direct Example Based Classifiers

For a direct example based classifier, a web page to be classified is used as a query directly against a set of examples that identify categories. The web page is assigned to the category whose set of examples has the highest similarity with the web page. These classifiers are called lazy learning systems. K-nearest-neighbors classifier is a representative.

K-Nearest-Neighbors Classifier

In contrast to "eager learning" classifiers (e.g. Rocchio classifier) that have an explicit training phase before classify new documents, K-nearest-neighbors (KNN) [30, 70] is a lazy learning method that delays the learning process until a new document must be classified. KNN classifier has been successfully applied for document classification [62, 67, 107, 109, 111, 112].

KNN classifier compares a new document directly with the given training documents. It uses cosine metric to compute the similarity between two document vectors. It ranks, in a descend order, the training documents based on their similarities with the new document. The top k training documents are k-nearest neighbors of the new document and the k-nearest neighbors are used to predict the categories of the new document. Reference [109] showed that the performance of kNN is relatively stable for a large range of k, and $k = 30$, 45 or 65 were tested in their experiments.

The similarity score of each neighbor document is used as a weight for the associated category. To classify a new document, the likelihood score of a category can be calculated as [112]

$$y(\vec{d}', c_j) = \sum_{\vec{d}_i \in KNN} \text{sim}(\vec{d}', \vec{d}_i) y(\vec{d}_i, c_j) - b_j \tag{68}$$

where $\vec{d}_i$ is one of k-nearest neighbors of the new document $\vec{d}'$, $y(\vec{d}_i, c_j) \in \{0, 1\}$, is the classification for the neighbor $\vec{d}_i$ with respect to category c_j ($y = 1$ for yes; and $y = 0$ for no), and $\text{sim}(\vec{d}', \vec{d}_i)$ is the similarity (e.g. cosine similarity) between the new document d' and its neighbor d_i, and b_j is the category specific threshold. The category specific threshold b_j is automatically learned by using a validation set of documents. That is, KNN algorithm learns the optimal threshold b_j for category c_j in that it yields the best performance on the validation documents [112]. The new document is assigned to those categories having likelihood scores larger than a predefined threshold.

4.4 Parameter Based Classifiers

For parameter based classifiers, training examples are used to estimate parameters of a probability distribution. Probabilistic Naïve Bayes classifier is an example.

Naïve Bayes Classifiers

The naïve Bayes classifier, as described in [45, 70], is constructed by using training examples to estimate the probability of each category given a new document d', which is written as $P(c_j|d')$:

$$Pr(c_j|d') = \frac{Pr(c_j) \cdot Pr(d'|c_j)}{Pr(d')} \tag{69}$$

The denominator in the above equation does not differ between categories and can be left out. The naïve Bayes classifier makes an assumption of feature independence in order to estimate $P(d'|c_j)$ as

$$Pr(c_j|d') = Pr(c_j) \prod_{f_i \in d'} Pr(f_i|c_j) \tag{70}$$

where $Pr(c_j)$ is the proportion of training examples in category c_j, and f_i is a feature (or term) found in document d'. An estimate for $Pr(f_i|c_j)$ is given by [1, 70]:

$$\tilde{Pr}(f_i|c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^{M} N_{kj}} \tag{71}$$

where N_{ij} is the number of times feature f_i occurring within documents from category c_j, and M is the total number of features in the training set. The category with the maximum value of $P(c_j|d')$ is the desired category for document d'. The work of the Naïve Bayes classifier applied in text classification could be found in [46, 58, 59, 63, 73, 74, 109]. Because the fact that the assumption of feature independence is general not true for documents, the Naïve Bayes classifier showed relatively worse performance in [109] and [63].

5 Evaluation of Web Page Classifiers

This section describes how to evaluate web page classifiers and reports experimental results. While the above section concerns more about the training phase for building a classifier, this section discusses how to evaluate a classifier in the testing phase. The testing phase performs on the testing examples, which are a part of all available examples. The other part of available examples consists of training examples used in the training phase and/or validation examples used for optimizing the model generated from the training phase. Criteria for performance measures are first described in the following. Results of experiments are then provided.

5.1 Performance Measures

The experimental evaluation of classifiers tries to evaluate the effectiveness of the learned model, i.e. its capability of making the right classification decision. The most frequently used measures of classification effectiveness are presented as follows.

A classifier can be evaluated in terms of precision, recall, accuracy, and error. Precision may be viewed as the degree of soundness of the classifier, while recall may be viewed as its degree of completeness. The precision and recall can be estimated in terms of the contingency table for category c_i on a given test set (see Table 4) [1, 97].

Table 4. The contingency table for category c_i

a: the number of testing examples correctly assigned to this category
b: the number of testing examples incorrectly assigned to this category
c: the number of testing examples incorrectly rejected from this category
d: the number of testing examples correctly rejected from this category

From the quantities in Table 4, precision and recall for a category are defined as:

$$\text{precision} = \frac{a}{a+b} \tag{72}$$

$$\text{recall} = \frac{a}{a+c} \tag{73}$$

Precision has similar meaning as classification accuracy. But they are difference in that precision considers only examples assigned to the category, while accuracy considers both assigned and rejected cases. Accuracy and error for a category are defined as:

$$\text{accuracy} = \frac{a+d}{a+b+c+d} \tag{74}$$

$$\text{error} = \frac{b+c}{a+b+c+d} \tag{75}$$

The above definitions are applicable for each category. To obtain measures relating to all categories, two methods may be adopted: microaveraging and macro-averaging [1, 97, 109]:

- **Micro-averaging**: the performance measures are obtained by globally summing over all individual decisions, i.e.

$$\text{precision} = \frac{\sum_{i=1}^{K} a_i}{\sum_{i=1}^{K} (a_i + b_i)} \tag{76}$$

where K is the number of categories, a_i is the number of testing examples correctly assigned to category i, and b_i is the number of testing examples incorrectly assigned to category i.

- **Macro-averaging**: the performance measures are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories, i.e.

$$\text{precision} = \frac{\sum_{i=1}^{K} \text{precision}_i}{K} \tag{77}$$

Recall, accuracy, and error for all categories can be computed similarly. It is important to recognize that these two methods may give quite different results, especially if the categories are unevenly populated.

Precision or recall may be misleading when considered alone since they are interdependent. Thus, a combined measure is considered [1, 97]. The effectiveness of a classifier is expressed as a function F_α in terms of both precision and recall as follow:

$$F_\alpha = \frac{1}{\alpha \cdot \frac{1}{\text{precision}} + (1-\alpha) \cdot \frac{1}{\text{recall}}} \tag{78}$$

where $0 \leq \alpha \leq 1$. A value of $\alpha = 0.5$ is usually used, which attributes equal importance to precision and recall and is usually referred to as F_1 measure.

5.2 Experimental Settings and Results

A large number and diversity of document classifiers have been proposed. Comparing the effectiveness of these classifiers has shown to be difficult. Many of the classifiers were tested with different data sets and under different experimental settings. However, only the evaluations of different classifiers under same experimental setting and using same data set are comparable. The following first outlines commonly used data sets and then reports attempts to compare classifiers under same experimental settings.

Data Sets

For testing classifiers, standard text collections can be found in public domain. Typical examples include [109, 119]:

- The *Reuters*-21578 corpus: The documents are newswire stories covering the period between 1987 and 1991.
- The *OHSUMED* corpus: The documents are title and abstract from medical journals.
- The *Yahoo* corpus: It is provided by Yahoo.com consisting of a directory of manually organized web pages.
- The 20 *Newsgroups* collection: The documents are messages posted to Usenet newsgroups, and the categories are the newsgroups themselves.

- The *Hoovers* corpora of company Web pages: It contains detailed information about a large number of companies and is a reliable source of corporate information.
- The *WebKB* collection: The documents are web pages that were assembled for training an intelligent web crawler.
- The *Brown* text document collection: This dataset is used for genrebased classification.

Experimental Comparison of Classifiers

An evaluation of fourteen classifiers for the subject-based classification was reported in [109]. Here we cite the experimental results in [109] to show the performance difference among classifiers (see Table 5). The results indicate that kNN classifier has the best performance. Among the top classifiers are LLSF, NNet, and WH. The next group is the rule induction algorithms, such as SWAP-1, RIPPER and CHARADE, showing a similar performance. Rocchio and Naïve Bayes had relatively worse performance.

Reference [112] re-examined five document classifiers, SVM, kNN, NNet, LLSF and Naïve Bayes and focused on the robustness of these classifiers in dealing with a skewed category distribution. The experimental results showed that SVM, kNN, and LLSF significantly outperformed NNet and Naïve Bayes when the number of positive training examples per category is less than ten.

Reference [112] evaluated eight classifiers, including SVM, linear regression (LLSF), logistic regression (LR), NNet, Rocchio, Prototypes, kNN, and Naïve Bayes. They used a loss function to analyze the optimization criterion of each classifier. The reason for using a loss function is that the optimization of a classifier is not only driven by the training set error but also driven by the model complexity. The loss function of a classifier L_c is defined as *the training set loss + the complexity penalty*. Balancing between the two criteria has been referred as the regularization of a classifier. The degree of regularization is controlled by a parameter in the classifier. They showed that regularization made significant improvement on the performances of LLSF and NNet, and also showed that the performances of regularized LLSF, NNet, LR and SVM were quite competitive. kNN was among the top classifiers that include SVM, regularized LR, NNet and LLSF. Rocchio was second. Naïve Bayes and Prototype were the last.

However, no such experiments have been conducted for the genre-based classifiers. Most work for genre-based classification employed neural network, decision tree, or rule-based learning methods [27, 35]. No such thorough evaluation of these classifiers in terms of genre-based classification has been reported in the literature.

Table 5. Test results for comparing fourteen classifiers [109]

	Reuters Apte BrkEvn	Reuters PARC BrkEvn	OHSUMED Full Range $F(\beta = 1)$	OHSUMED HD Big $F(\beta = 1)$	Reuters Lewis BrkEvn	Reuters CONS. BrkEvn
kNN(N)	0.85*	0.82*	0.51*	0.56	0.69	–
LLSF(L)	0.85*	0.81* (−1%)	–	–	–	–
NNets(N)	–	0.82*	–	–	–	–
WH	–	–	–	0.59* (+5%)	–	–
EG(L)	–	–	–	0.54 (−4%)	– –	– –
RIPPER(N)	0.80 (−6%)	–	–	–	0.72	–
DTree (N)	[0.79]	–	–	–	0.67	–
SWAP–1 (N)	0.79 (−7%)	–	–	–	–	–
CHARADE (N)	0.78 (−8%)	–	–	–	–	–
EXPERTS (N)	0.76 (−11%)	–	–	–	0.75*	–
Rocchio (L)	0.75 (−12%)	–	–	0.46 (−18%)	0.66	–
NaiveBayes (L)	0.71 (−16%)	–	–	–	0.65	–
CONSTRUE	–	–	–	–	–	0.90*
WORD	0.29 (−66%)	0.25 (−69%)	0.27 (−47%)	0.44 (−21%)	0.15	–

L a linear method, *N* a non-linear model, *the local optimal on a fixed collection, *(x%)* the performance improvement relative to kNN, [*x*] a F_1 measure. kNN *k*-nearestneighbor algorithm, *WH* Widrow-Hoff learning algorithm (also known as Least Mean Square), *EG* Exponential Gradient algorithm, *DTree* decision tree learning algorithm, *SWAP-1* an rule-based learning algorithm, *CHARADE* an expert system consisting of manually developed categorization rules, *EXPERTS* Sleeping Experts, *CONSTRUE* a rule learning system, *WORD* a non-learning algorithm as a baseline of the comparison.

6 Summary

Web page classification is the assignment of web pages to one or more predefined categories based on their subjects or genres. A general inductive process automatically builds a model by learning over a set of previously classified web pages. This model is then used to classify new web pages.

A typical web page classification process consists of the following steps: extracting salient features from training web pages, creating a feature vector

for each web page, reducing dimensionality of the feature vectors, creating a classifier by learning over the training feature vectors, and then classifying new web pages using the classifier.

Web pages can be classified in terms of subjects or genres. For subject-based classification, the salient features of web pages are the text contents, such as words, phases, and sentences. For genre-based classification, the salient features are the genre attributes, such as presentation characteristics and functional words.

Dimensionality reduction methods, including feature selections and extractions, are used to reduce the feature space of the training web pages. Feature selection techniques are used to select subset from the original feature space based on criterions, such as Information Gain, Cross Entropy, and Odds Ratio. Feature extraction techniques, such as Latent Semantic Indexing and Word Clustering, are used to transform the original large feature space into a smaller feature space having possibly new set of features.

Numerous classifiers proposed and used for machine learning can be applied for web page classification. Various classifiers that have been applied and to some extent proven efficient for web page classification are described in this chapter.

References

1. Aas K, Eikvil L (1999) Text categorization: a survey. Report NR 941, Norwegian Computing Center 259, 262, 263, 264
2. Agrawal R, Srikant R (1994) Fast algorithm for mining association rules. In: Proceedings of 1994 Int. Conf. Very Large Data Bases. Santiago, Chile, pp. 487–499 258
3. Apte C, Damerau F, Weiss S (1994) Towards language independent automated learning of text categorization models. In: Proceedings of SIGIR '94: 17th ACM international conference on research and development in information retrieval. Dublin, Ireland, pp. 24–30 256
4. Argamon S, Koppel M, Avneri G (1998) Routing documents according to style. In: First International Workshop on Innovative Information Systems. Pisa, Italy 235, 236
5. Baker LD, McCallum A (1998) Distributional clustering of words for text classification. In: Proceedings of SIGIR '98: 21st ACM international conference on research and development in information retrieval. Melbourne, Australia, pp. 96–103 249
6. Bazerman C (1998) Shaping written knowledge: the genre and activity of the experimental article in science. The University of Wisconsin Press, Madison, WI, USA 234
7. Berkhin P (2000) Survey of clustering data mining techniques. http://www.accrue.com/products/researchpapers.html 249
8. Berry MW (1992) Large-scale sparse singular value computations. The International J. of Supercomputer Applications 6: 13–49

9. Berry MW, Dumais ST, O'Brien GW (1995) Using linear algebra for intelligent information retrieval. SIAM Review 37: 573–595 247, 249
10. Biber, D (1995) Dimensions of register variation: a cross-linguistic comparison. Cambridge University Press, Cambridge, England 235
11. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: COLT: Proceedings of the workshop on computational learning theory. Morgan Kaufmann Publishers Inc., San Francisco, CA 231
12. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Belmont, CA: Wadsworth 259
13. Brown JS (1994) Borderline issues: social and material aspects of design. Human Computer Interactions 9: 3–36 234
14. Buckley C, Salton G, Allan J (1994) The effect of adding relevance information in a relevance feedback environment. In: Proceedings of SIGIR '94: 17th ACM international conference on research and development in information retrieval. Dublin, Ireland, pp. 292–300 250
15. Burrows J (1992) Not unless you ask nicely: the interpretative nexus between analysis and information. Literary and Linguistic Computing 7: 91–109 235
16. Chakrabati S, Dom BE, Indyk P (1998) Enhanced hypertext categorization using hyperlinks. In: Proceedings of SIGMOD-98: ACM International conference on management of data. Seattle, WA, USA, pp. 307–318 232
17. Choi B (2001) Making sense of search results by automatic web-page classification. In: WebNet 2001. Orlando, Florida, USA, pp. 184–186 221
18. Choi B, Guo Q (2003) Applying semantic links for classifying web pages. Lecture Notes in Artificial Intelligence 2718: 148–153 232
19. Choi B, Li B (2002) Abstracting keywords from hypertext documents. In: International conference on information and knowledge engineering. Las Vegas, Nevada, USA, pp. 137–142 233
20. Church KW, Hanks P (1989) Word association norms, mutual information and lexicography. In: Proceedings of ACL 27. Vancouver, Canada, pp. 76–83 239
21. Cohen WW (1995) Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning. Lake Tahoe, CA, USA, pp 115–123
22. Cohen WW, Singer Y (1996) Context-sensitive learning methods for text categorization. In: Proceedings of SIGIR '96: 19th ACM international conference on research and development in information systems. Zurich, CH, pp. 307–315 251, 256, 257, 258
23. Cormen TH, Leiserson CE and Rivest RL (1990) Introduction to algorithms. Cambridge, MA: MIT Press 249
24. Cortes C, Vapnik V (1995) Support vector networks. Machine learning 20: 273–297 252
25. Cover TM, Thomas JA (1991) Elements of information theory. Wiley, New York 240
26. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J. Amer. Soc. Inf. Sci. 41: 391–407 247
27. Dewdney N, VanEss-Dykema C, MacMillan R (2001) The form is the substance: classification of genres in text. In: ACL workshop on human language technology and knowledge management. Toulouse, France 235, 236, 237, 252, 253, 259, 265
28. Dillon IS, Mallela S, Kumar R (2002) Enhanced word clustering for hierarchical text classification. In: Proceedings of the 8th ACM SIGKDD. Edmonton, Canada, pp. 191–200 249

29. Dongarra JJ, Moler CB, Bunch JR, Stewart GW (1979) LINPACK users' guide. Philadelphia, PA: SIAM
30. Duda RO, Hart PE (1973) Pattern Classification and Scene Analysis. John Wiley and Sons, New York 261
31. Duda RO, Hart PE, Stork DG (2001). Pattern classification, 2nd edn. Wiley, New York 249, 250
32. Dumais S, Chen H (2000) Hierarchical classification of web content. In: Proceedings of the 23rd ACM international conference on research and development in information retrieval (ACM SIGIR). Athens, Greece, pp 256–263 245
33. Dumais S, Platt J, Hecherman D, Sahami M (1998) Inductive learning algorithm and representations for text categorization. In: Proceedings of CIKM-98: 7th ACM international conference on information and knowledge management. Bethesda, MD, USA, pp. 148–155 252
34. Everitt BS, Landua S, Leese M (2001) Cluster Analysis. Arnold, London, UK 249
35. Finn A, Kushmerick N (2003) Learning to classify documents according to genre. In: IJCAI-2003 workshop on computational approaches to text style and synthesis. Acapulco, Mexico 235, 237, 259, 265
36. Furnkranz J (1999) Exploiting structural information for text classification on the WWW. In: Proceedings of IDA 99: 3rd Symposium on intelligent data analysis. Amsterdam, NL, pp. 487–497 231
37. Furnkranz J, Widmer G (1994) Incremental reduced error pruning. In: Proceedings of the 11th annual conference on machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA
38. Galavotti L, Sebastiani F, Simi M (2000) Experiments on the use of feature selection and negative evidence in automated text categorization. In: Proceedings of ECDL-00: 4th European conference on research and advanced technology for digital libraries. Lisbon, PT, pp. 59–68 239, 243, 244
39. Glover EJ, Tsioutsiouliklis K, Lawrence S, Pennock DM, Flake GW (2002) Using web structure for classifying and describing web pages. In: Proceedings of international world wide web conference. Honolulu, Hawaii, pp. 562–569 231, 241
40. Golub B, Loan CV (1989) Matrix computations (2nd Ed). Johns-Hopkins, Baltimore 249, 256
41. Haas SW, Grams ES (1998) Page and link classifications: connecting diverse resources. In: Proceedings of the 3rd ACM conference on digital libraries. Pittsburgh, PA, USA, pp. 99–107
42. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: ACM-SIGMOD international conference on management of data. Dallas, TX, USA, pp. 1–12 258
43. Ittner DJ, Lewis DD, Ahn DD (1995) Text categorization of low quality images. In: Symposium on document analysis and information retrieval. Las Vegas, NV, USA, pp. 301–315 250
44. Jain AK, Murty M N, Flynn PJ (1999) Data clustering: a review. ACM computing surveys 31: 255–323 249
45. Joachims T (1997) A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: International conference on machine learning (ICML). Nashville, TN, USA, pp. 143–151 227, 251, 262
46. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: European conference on machine learning. Berlin, German, pp. 137–142 252, 253, 262

47. John G, Kohavi R, Pfleger K (1994) Irrelevant features and the subset selection problem. In: Machine learning: proceedings of the 11th international conference. Morgan Kaufman Publishers Inc., San Francisco, CA, pp. 121–129 238
48. Jones KS (1972) A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28: 11–20 226
49. Jung Y, Park H, Du D (2000) An effective term-weighting scheme for information retrieval. Computer science technical report TR008. Department of computer science, University of Minnesota, Minneapolis, Minnesota 227, 228
50. Karlgren J, Cutting D (1994) Recognizing text genres with simple matrices using discriminant analysis. In: Proceeding of the 15th international conference on computational linguistics. Kyoto, Japan, pp. 1071–1075 235
51. Kaufman L and Rousseeuw PJ (1990) Finding groups in data. New York: Wiley 249
52. Kessler B, Nunberg G, Schutze H (1997) Automatic detection of text genre. In: Proceeding of 35th annual meeting of the association for computational linguistics. Madrid, Spain, pp. 32–38 223, 235, 253
53. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artificial Intelligence Journal 97(1–2): 273–324 238
54. Koller D, Sahami M (1996) Toward optimal feature selection. In: Lorenza Saitta (eds) Machine learning: proceedings of the 13th international conference. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 294–292 238, 240
55. Koller D, Sahami M (1997) Hierarchically classifying documents using very few words. In: Proc. of the 14th international conference on machine learning (ICML97). Nashville, TN, USA, pp. 170–178 240, 244, 245
56. Lee Y, Myaeng SH (2002) Text genre classification with genre-revealing and subject-revealing features. In: Proceedings of SIGIR '02: 25th ACM international conference on research and development in information retrieval. Tampere, Finland, pp. 145–150 235, 237
57. Lesk ME (1986) Automatic word sense disambiguation: how to tell a pine cone from an ice cream cone. In: DeBuys V (eds) Proceedings of SIGDOC-86: 5th ACM international conference on systems documentation. New York, US, pp. 24–26 233
58. Lewis DD (1992) An evaluation of phrasal and clustered representation on a text categorization task. In: Proceedings of SIGIR-92: 15th ACM international conference on research and development in information retrieval. Kobenhavn, DK, pp. 37–50 250, 262
59. Lewis DD (1998) Naïve (Bayes) at forty: the independence assumption in information retrieval. In: Proceedings of ECML-98: 10th European conference on machine learning. Chemnitz, DE, pp. 4–15 262
60. Lewis DD, Ringuette M (1994) Comparison of two learning algorithms for text categorization. In: Proceedings of the 3rd annual symposium on document analysis and information retrieval (SDAIR '94). Las Vegas, Nevada, USA, pp. 81–93 259
61. Lewis DD, Schapire RE, Callan JP, Papka R (1996) Training algorithms for linear text classifiers. In: Proceedings of SIGIR-96: 19th ACM international conference on research and development in information retrieval. Zurich, CH, pp. 298–306 251, 255
62. Li F, Yang Y (2003) A loss function analysis for classification methods in text categorization. In: Proceeding of the twentieth international conference on machine learning (ICML 2003). Washington, DC, USA, pp. 472–479 251, 252, 253, 255, 261

63. Li YH, Jain AK (1998) Classification of text documents. The Computer Journal 41: 537–546 249, 262
64. Luhn HP (1958) The automatic creation of literature abstracts. IBM Journal of Research and Development 2: 159–165 225
65. Malone TW et al. (1999) Tools for inventing organizations: toward a handbook of organizational process. Management Science 45: 425–443 234
66. Manning and Schutze (2001) Foundations of statistical natural language processing. The MIT Press, Cambridge Massachusetts, London England 249
67. Masand B, Linoff G, Waltz D (1992) Classifying news stories using memory based reasoning. In: 15th Ann. Int. ACM SIGIR conference on research and development in information retrieval (SIGIR '92). Kobenhavn, DK, pp. 59–64 261
68. Mihalcea R, Moldovan D (2001) A highly accurate bootstrapping algorithm for word sense disambiguation. International Journal on Artificial Intelligence Tools 10: 5–21. 233
69. Miller CR (1984) Genre as social action. Quarterly Journal of Speech 70: 151–167 234
70. Mitchell T (1997) Machine Learning. McGraw Hill, New York 237, 241, 242, 254, 255, 259, 261, 262
71. Mladenic D (1998) Feature subset selection in text-learning. In: Proceeding of the 10th European conference on machine learning (ECML98). Chemnitz, Germany, pp. 95–100 222, 226
72. Mladenic D (1998) Machine learning on non-homogeneous, distributed text data. PhD thesis, University of Ljubljana, Slovenia 241
73. Mladenic D, Grobelnik M (1998) Feature selection for classification based on text hierarchy. In: Working notes of learning from text and the web: conference on automated learning and discovery. Carnegie Mellon University, Pittsburgh 241, 245, 246, 247, 262
74. Mladenic D, Grobelnik M (1999) Feature selection for unbalanced class distribution and Naïve Bayes. In: 16th International conference on machine learning. Bled, Slovenia, pp. 258–267 240, 241, 246, 247, 262
75. Moulinier I (1997) Is learning bias an issue on the text categorization problem? Technical report, LAFORIA-LIP6, University Paris, VI 259
76. Ng HT, Goh WB, Low KL (1997) Feature selection, perceptron learning, and a usability case study for text categorization. In: 20th Ann. Int. ACM SIGIR conference on research and development in information retrieval (SIGIR '97). Philadelphia, PA, USA, pp. 67–73 253
77. Oh H, Myaeng SH, Lee M (2000) A practical hypertext categorization method using links and incrementally available class information. In: Proceedings of SIGIR-00: 23rd ACM international conference on research and development in information retrieval. Athens, GR, pp. 264–271 231
78. Orlikowski WJ, Yates J (1994) Genre repertoire: examining the structuring of communicative practices in organizations. Administrative Science Quarterly 39: 541–574 234
79. Orlikowski WJ, Yates J (1998) Genre systems as communicative norms for structuring interaction in groupware. Working paper #205, MIT Center for Coordination Science
80. Osuna E, Freund R, Girosi F (1997) Support vector machines: training and applications. A. I. memo No. 1602, A. I. Lab, Massachusetts Institute of Technology 252, 253
81. Paice CD (1990) Constructing literature abstracts by computer: techniques and prospects. Information Processing and Management 26: 171–186 229

82. Peng X, Choi B (2002) Automatic web page classification in a dynamic and hierarchical way. In: IEEE international conference on data mining. pp. 386–393 229, 245
83. Quinlan JR (1986) Induction of decision trees. Machine Learning 1: 81–106 259
84. Quinlan JR (1993) C4.5: programming for machine learning. Morgan Kaumann publishers, San Francisco, CA 259, 260
85. Quinlan JR (1995) MDL and categorical theories (continued). In: Proceedings of the 12th international conference on machine learning. Lake Tahoe, CA, pp. 464–470 257, 258
86. Ragas H, Koster CH (1998) Four text classification algorithms compared on a dutch corpus. In: Proceedings of SIGIR-98: 21st ACM international conference on research and development in information retrieval. Melbourne, AU, pp. 369–370 251
87. Rasmussen E (1992) Clustering algorithms. In: Willian B. Frakes and Ricardo Baeza-Yates (eds.), information retrieval. Englewood Cliffs, NJ: Prentice Hall, pp. 419–442 249
88. Rauber A, Muller-Kogler A (2001) Integrating automatic genre analysis into digital libraries. In: 1st ACM-IEEE joint conference on digital libraries. Roanoke, VA, USA, pp. 1–10 235, 236
89. Riboni D (2002) Feature selection for web page classification. In: 2002 proceedings of the workshop of first EurAsian conference on advances in information and communication technology. Shiraz, Iran 229
90. Rijsbergen V, Harper CJ, Porter DJ (1981) The selection of good search terms. Information Processing and Management 17: 77–91 241
91. Rocchio J (1971) Relevance feedback in information retrieval. In: Salton (ed) The SMART retrieval system: experiments in automatic document processing. Prentice-Hall, New Jersey, pp. 313–323 250
92. Ruiz ME, Srinivasan P (2002) Hierarchical text categorization using neural networks. Information Retrieval 5: 87–118 241, 246
93. Salton G (1989) Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley, New York 223
94. Salton G (1991) Developments in automatic text retrieval. Science 253: 974–979 226
95. Salton G, Buckley C (1988) Term weighting approaches in automatic text retrieval. Information Processing and Management 24: 513–523 226
96. Schutze H, Hull DA, Pedersen JO (1995) A comparison of classifiers and document representations for the routing problem. In: 18th Ann. Int. ACM SIGIR conference on research and development in information retrieval. Seattle, WA, USA, pp. 229–237 247, 253
97. Sebastiani F (1999) A tutorial on automated text categorization. In: Proceedings of ASAI-99: 1st Argentinean symposium on artificial intelligence. Buenos Aires, AR, pp. 7–35 237, 247, 249, 263, 264
98. Sebastiani F (2002) Machine learning in automated text categorization. ACM Computing Surveys 34: 1–47 222
99. Silverstein C and Pedersen JO (1997) Almost-constant-time clustering of arbitrary corpus subsets. In: Proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR '97). Philadelphia, PA, pp. 60–66 249

100. Slonim N, Tishby N (2001) The power of word clusters for text classification. In: Proc. 23rd European colloquium on information retrieval research (ECIR). Darmstadt, DE. 249
101. Stamatatos E, Fakotakis N, Kokkinakis G (2000) Text genre detection using common word frequencies. In: Proceeding of the 18th international conference on computational linguistics (COLING2000). Luxembourg, pp. 808–814 235, 236
102. Strehl A (2002) Relationship-based clustering and cluster ensembles for high-dimensional data mining. Dissertation, the University of Texas at Austin 237
103. Sun A, Lim E, Ng W (2002) Web classification using support vector machine. Proceedings of the 4th international workshop on Web information and data management. McLean, Virginia, USA, pp. 96–99 252
104. Vapnik V (1995) The nature of statistical learning theory. Springer, New York 252, 253
105. Wiener E, Pederson JO, Weigend AS (1995) A neural network approach to topic spotting. In: Proceedings of the 4th annual symposium on document analysis and information retrieval (SDAIR '95). Las Vegas, US, pp. 317–332 247, 252, 253
106. Willett P (1988) Recent trends in hierarchic document clustering: a critical review. Information Processing & Management 24: 577–597 249
107. Yang Y (1994) Expert network: effective and efficient learning for human decision in text categorization and retrieval. In: Proc. of the 7th annual international ACM-SIGIR conference on research and development in information retrieval. Dublin, Ireland, pp. 13–22 261
108. Yang Y (1995) Noise reduction in a statistical approach to text categorization. In: Proceedings of the 18th Ann. Int. ACM SIGIR conference on research and development in information retrieval (SIGIR '95). Seattle, WA, USA, pp. 256–263 247, 255, 256, 259
109. Yang Y (1999) An evaluation of statistical approaches to text categorization. Information Retrieval 1: 69–90 251, 253, 255, 256, 257, 261, 262, 263, 264, 265, 266
110. Yang Y, Chute CG (1992) A linear least squares fit mapping method for information retrieval from natural language texts. In: Proceedings of the 14th international conference on computational linguistics (COLING 92). Nantes, France, pp. 447–453 255, 256
111. Yang Y, Chute CG (1994) An example-based mapping method for text categorization and retrieval. ACM Transaction and Information System 12: 253–277 255, 256, 261
112. Yang Y, Liu X (1999) A re-examination of text categorization methods. In: Proceedings of SIGIR-99: 22nd ACM international conference on research and development in information retrieval. Berkeley, CA, US, pp. 42–49 252, 255, 261, 265
113. Yang Y, Pederson JO (1997) A comparative study on feature selection in text categorization. In: Proc. of the 14th international conference on machine learning (ICML97). Morgan Kaufmann Publishers Inc, San Francisco, USA, pp. 412–420 239, 240, 243, 246, 247
114. Yang Y (2001) A study on thresholding strategies for text categorization. In: Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2001). New Orleans, LA, USA, pp. 137–145 231, 250
115. Yang Y, Slattery S, Ghani R (2001) A study of approaches to hypertext categorization. Journal of Intelligent Information Systems 18(2–3): 219–241
116. Yao Z, Choi B (2003) Bidirectional hierarchical clustering for web mining. In: Proc. of the 2003 IEEE/WIC international conference on web intelligence. Halifax, Canada, pp. 620–624 249

117. Yoshioka T, Herman G (1999) Genre taxonomy: a knowledge repository of communicative actions. Working paper #209, MIT Center for Coordination Science 234
118. Zaiane OR, Antonie ML (2002) Classifying text documents by associating terms with text categories. In: Proceedings of the 13th Australasian conference on database technologies, Melbourne, Australia, pp. 215–222 258, 259
119. Zhang T, Oles FJ (2001) Text categorization based on regularized linear classification methods. Information Retrieval 4: 5–31 255, 264