Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Dunja Mladenić

Strojno učenje na nehomogenih, distribuiranih tekstovnih podatkih

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Ivan Bratko

Komentor: prof. dr. Tom M. Mitchell

Ljubljana, 1998

University of Ljubljana

Faculty of Computer and Information Science

Dunja Mladenić

Machine Learning on non-homogeneous, distributed text data

DOCTORAL DISSERTATION

Advisors: Prof. Ivan Bratko, Prof. Tom M. Mitchell

Ljubljana, 1998

Abstract

This dissertation proposes new machine learning methods where the corresponding learning problem is characterized by a high number of features, unbalanced class distribution and asymmetric misclassification costs. The input is given as a set of text documents or their Web addresses (URLs). The induced target concept is appropriate for the classification of new documents including shortened documents describing individual hyperlinks. The proposed methods are based on several new solutions.

Proposed is a new, enriched document representation that extends the bag-of-words representation by adding word sequences and document topic categories. Features that represent word sequences are generated using a new efficient procedure. Features giving topic categories are obtained from background knowledge constructed using the new machine learning method for learning from class hierarchies. When learning from class hierarchy, a high number of class values, examples and features, are handled by (1) dividing a problem into subproblems based on the hierarchical structure of class values and examples, (2) by applying feature subset selection and (3) by pruning unpromising class values during classification.

Several new feature scoring measures are proposed as a result of comparison and analysis of different feature scoring measures used in feature subset selection on text data. The new measures are appropriate for text domains with several tens or hundreds of thousands of features, can handle unbalanced class distribution and asymmetric misclassification costs.

Developed methods are suitable for the classification of documents including shortened documents. We build descriptions of hyperlinks, and treat these as shortened documents. Since each hyperlink on the Web is pointing to some document, the classification of hyperlinks (corresponding shortened documents) could be potentially improved by using this information. We give the results of preliminary experiments for learning in domains with mutually dependent class attributes. Training examples are used for learning 'a next state function on the Web', where document content (class attributes) is predicted from the hyperlink (feature-vector) that points to the document. Document content we are predicting is represented as a feature-vector each feature being one of the mutually dependent class attributes.

The proposed methods and solutions are implemented and experimentally evaluated on real-world data collected from the Web in three independent projects. It is shown that document classification, categorization and prediction using the proposed methods perform well on large, real-world domains.

The experimental findings further indicate that the developed methods can efficiently be used to support analysis of large amount of text data, automatic document categorization and abstraction, document content prediction based on the hyperlink content, classification of shortened documents, development of user customized text-based systems, and user customized Web browsing. As such, the proposed machine learning methods contribute to machine learning and to related fields of text-learning, data mining, intelligent data analysis, information retrieval, intelligent user interfaces, and intelligent agents. Within machine learning, this thesis contributes an approach to learning on large, distributed text data, learning on hypertext, and learning from class hierarchies. Within computer science, it contributes to better design of Web browsers and software assistants for people using the Web.

Keywords

machine learning

inductive concept learning

text-learning

information retrieval

feature selection

document representation

learning on class hierarchy

classification in topic categories

Web browsing

hyperlink

hypertext

software assistant

Acknowledgements

My gratitude goes first to my advisors, Prof. Ivan Bratko and Prof. Tom Mitchell for their generous support and comments to my research work. I would like to express my thanks to Assist. Prof. Nada Lavrač, my advisor at J. Stefan Institute for her support, help and numerous advises.
A great part of my work described in this thesis was done at the School of Computer Science, Carnegie Mellon University and I am grateful to all the nice people that I met during my stay and work there, especially to my host and research advisor Prof. Tom Mitchell and the members of his Learning lab. It was him who pointed me out the challenges of using machine learning methods on text data and the Web. Home environment during my stay there was almost as important for my thesis work as my work environment and I would like to express many thanks to Andrej Bauer, Fay Chang, Michael Harkavy, Robert Murray and Christopher Stone for being great housemades. I am also grateful to all the dancers from the Pittsburgh contact improvisation group for many nice and relaxing moments that contributed to my work.
Many thanks go to the members of the Department for Intelligent Systems at the Jožef Stefan Institute in Ljubljana for accepting and supporting me for the last six years and providing a pleasant working environment. I also thank the members of the Artificial Intelligence Laboratory at Faculty of Computer and Information Science in Ljubljana, whose close cooperation with our department was helpful for my research work.
Thanks to the Slovenian Ministry of Science and Technology for my PhD grant.
I would like to thank Irena Nančovska and Anka Korošec for being great friends for many years. Their support was one of the important 'lights on my path' toward finishing this thesis. Foremost, I would like to thank my parents, Višnja and Zvonko Mladenić, for endless love they have always provided me. Special thanks goes to Marko Grobelnik for his encouragement, fruitful cooperation and help in my research work. His generous help, supportive love and ability to remind me on my own words, goals and 'reasons why am I doing this' was of great importance for my thesis work as well as for my research and personal life for the last ten years. Finally, my silent thanks and admiration goes to all life and beings.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the growing availability of information sources, especially large, non-homogeneous, distributed sources like the World Wide Web, there is also a growing interest in tools that can help in making a good and quick selection of information we are interested in. Recent work that arises at the intersection of Information Retrieval and Machine Learning offers some novel solutions to this problem, as well as work in Intelligent Agents and Intelligent User Interfaces. Connected to this is intensive development of methods for using machine learning techniques on text databases referred to as *text-learning*, that combine research in machine learning with information retrieval. We will take look at some of this work to motivate our research on the usage of machine learning techniques on large sources of text data.

1.1.1 Web browsing

Among the systems that are designed to help users finding required information the most popular are systems that help user in **browsing the Web**. Many of them are based on the content analysis (text analysis) and these are the most relevant for our work. For example, Armstrong et al. [7] developed WebWatcher, a system that assists a user in locating information on the World Wide Web taking keywords from the user, suggesting hyperlinks and receiving evaluation. Balabanović and Shoham [9, 10] developed "a system which learns to browse the Internet on behalf of a user". It searches the World Wide Web taking bounded amount of time, selects the best pages and receives an evaluation from the user. The evaluation is used to update the search and selection heuristics. Goldman et al. [32] developed Musag, a system that takes keywords from the user and searches the Web for relevant documents. The system generates a kind of user-independent thesaurus that relates concepts that are semantically similar to each other. Generated thesaurus is used in document retrieval to extend a set of given keywords. Lieberman [67] developed Letizia, "a user interface agent that assists a user browsing the World Wide Web". The system doesn't require any keywords or rating from the user, it infers the user interests from the browsing behavior. The users of the Web browsers usually perform depth-first search. Letizia tries to help with using the time while user is reading a Web document to perform a breath-first search from the current document. Potentially interesting hyperlinks are suggested to the user in a separate NETSCAPE window. Pazzani et al. [83, 84] collect ratings of the explored Web pages from the user and learn a user profile from them. Pages are separated according to their topic and a separate profile is learned for each topic.

There are also some other systems designed to help user in Web browsing that are not based on document content. For example, Krulwich [56] developed Lifestyle finder, a system for user profiles generation based on the usage of demographic data. Questionary is used to get users characteristics and group similar users in terms of demographic data. A set of the 15 highly scored Web documents is suggested to each user. Users evaluation of suggested documents is used to evaluate system performance. Menczer [72] developed "adaptive intelligent methods to automate on-line information search and discovery in the Web" (like Web robots or crawlers) based on population of intelligent agents. He used genetic algorithms on the population of agents, rewarding agent with energy for each relevant document found on the Web and charging them energy for the usage of network resources incurred by transferring documents. Rucker and Marcos [92] developed Siteseer, a "Web-page recommendation system that uses an individual's bookmarks and the organization of bookmarks within folders for predicting and recommending relevant pages". The system measures the degree of overlap (such as common URLs) between the bookmark files of different users and group users according to that similarity. When making recommendation, priority is given to URLs from more similar folders and URLs that appear in several bookmark files of similar users. Terveen et al. [98] propose PHOAKS, a system that automatically recognizes and redistributes recommendations of the Web resources mined from the Usenet news messages. The system assumes the roles of provider and recommendation recipient are specialized and different. It reuses recommendations from existing online conversations. The special attention is given to the problem of distinguishing recommended Web pages (URLs) from advertised or announced pages. The system includes categorization rules that implement a strategy to distinguish the different purposes for which the Web resources are mentioned.

1.1.2 Finding interesting news

Web browsing is just one of the problems where user is faced with the information overload. **Usenet news** filtering or recommendation is the other problem occurring in almost every-days life of many computer users. Lang [62] developed a system for electronic news filtering that uses text-learning to generate models of user interests. The system uses World Wide Web interface to enable user access the news in a usual way and to enable the system to collect user's ratings as feedback. NewsWeeder additionally assigns predicted rating to each article and generates a personalized list of the top articles (eg., 50 articles predicted as the most interesting) found among all articles. Konstan et al. [51] developed GroupLense, "a collaborative filtering system for Usenet news". The system has a two-part database to store ratings that users have given to messages and correlations between pairs of users based on their ratings. Since each user reads a small percentage of the total number of news messages, it is difficult to find other users with whom to correlate and an enormous number of ratings is needed to cover all of the messages. This problem of ratings sparsity is common for the collaborative approaches and different systems address it in different ways. GroupLense partitions the set of news messages into clusters that are commonly read together improving the local density of ratings.

Reading **newspapers** is also an every-day task for many users consisting of a lot of filtering to find articles we are interested in. Today, many newspapers are available on the Internet. Kamba et al. [43] developed Antagonomy, a system that composes personalized newspaper on the Web. The system monitors user operations on the articles and reflects them in the user profile. The layout of the composed newspaper is based on the scores given to the articles that reflects the degree of article matching the user profile. For example, more articles with the higher score are placed on the top of the newspaper.

1.1.3 Answering questions

A common problem people are faced to is trying to find **answers to different questions**. Many answers are hard to find, taking a lifetime. Still there are many frequently asked questions that can be more or less easily answered by someone else. Some of that questions are so called *frequently asked questions* - FAQ that are stored together with their answers in a database. Burke et al. [15, 16], Hammond et al. [35] developed a system that uses a "natural language question-based interface to access distributed text information sources" and helps the user to find answers to her/his question in a databases such as FAQ files. Questions from relevant FAQ files are matched against user's question and five best matching questions are returned together with their answers. Some of our questions could potentially be answered by an expert, if we are lucky to find one. Krulwich and Burkey [55] proposed "The ContactFinder agent" that reads and responds to bulletin board messages; assists users by referring them to other people who can help them and; categorizes messages and extracts their topic areas. The system operates in the two phases. In the first phase, it searches bulletin board looking for the contact persons and their topics area and storing found information in the database. In the second phase, the system searches new messages looking for questions, extracting the topic area of the found question and finding the contact person for that topic area in the database. To extract the topics area, ContacFinder uses some heuristics, for example, extracts semantically significant phrases (eg., fully capitalized words, short phrases: 1 to 5 words, words in a different format from surrounding text). Kautz et al. [45, 46] developed Referral Web, "an interactive system for reconstructing, visualizing, and searching the social networks on the World Wide Web". The motivation is, similar as for already mentioned content-based system Contact Finder, to help in searching for an expert on a given topic. A social network is modeled by a graph, where nodes represent individuals, and edges between nodes indicate that a direct relationship between the individuals has been detected. The network model is constructed incrementally with new users, searching for the co-occurrence of names in close proximity in any documents publicly available on the Web.

1.1.4 Different activities on computer

There are also systems that try to help computer users in different activities such as **using the Internet**, **browsing software libraries** or **meeting scheduling**. Etizioni and Weld [26] offer an integrated interface to the Internet combining UNIX shell and the World Wide Web to interact with Internet resources. Their agent accepts high-level user goals and dynamically synthesizes the appropriate sequence of Internet commands to satisfy those goals. Gams and Hribovšek [31] designed an intelligent-personal-agent interface for the VAX/VMS operating system that is based on a prior knowledge about the operating system and the specific knowledge about a user environment and interaction with computer. Gams and Grobelnik [30] developed an employment agent available through the Internet that enables user browsing data and ordering e-mails when an interesting information appears. Holte and Drummond [38], Drummond et al. [23] designed a system that assists browsing of software libraries, taking keywords from the user and using a rule-based system with forward chaining inference, assuming that the library consists of one type of item and the user goal is a single item. LaMacchia [61] proposed Internet Fish, a class of resource-discovery tools designed to help users extract useful information from the Internet. The system includes a natural language interface that currently permits only limited, structured interaction. The system also includes help in browsing the Web by usage of the existing search engines and user's rating of the documents. Mitchell et al. [74] proposed a system connected to the user's electronic calendar, that generates sets of rules capturing the user's scheduling preferences and some other information about individual attendees of meetings. It uses these rules to provide advice to the user for new, unscheduled meetings.

1.1.5 Entertainment recommendation

Besides help offered by computer systems in every-day activities such as informing ourselves by following the news, trying to find answers to different questions or scheduling meeting in our calendar, there are systems that try to help by **entertainment recommendation** (eg. music, books, movies). Maes [69] described "interface agents" that learn from the user as well as from other agents. As examples of such agents they developed agents for: electronic mail handling, meeting scheduling, electronic news filtering and entertainment recommendation. Some of these agents use the context of documents and adopt Information Retrieval approaches (eg. news filtering). Others rely on correlation between different users performing "social learning" (eg. entertainment recommendation). They developed a music recommendation system Ringo, that recommends music that was scored high by users with similar music tastes. The system trays to overcome the problem of sparse coverage of ratings with building models of virtual users interested in a very narrow range of music. The same group developed the Firefly system for music, movie and books recommendation [36]. Firefly requires from the user to begin with rating several predefined items, to ensure the possibility of comparing any two user (recall that two users can be compared only if they rated exactly the same items).

1.1.6 Thesis motivation

One of the known characteristics of this time is the information overload people are faced with. There is a frequent impression that simply a day has too few hours and there are so many things waiting that 'had to be finished till yesterday'. Of course it is an illusion but it is so realistic that any help in handling at least some simpler tasks would be appreciated.

Recently there is a rapidly growing importance of the Internet not only in research community but also broader in many every-day activities. The number of systems that aim to help users in different activities show that, as we have seen from the previous Sections. The growing interest and number of ideas where computers could be used is also pressing the research community to develop new methods and extend the existing approaches. Development of machine learning techniques for learning on large, non-homogeneous, distributed text data is just one of the raising problems. One of the available information sources is the World Wide Web and it is currently growing quickly, attracting many users with different interests. Since interaction with the World Wide Web (WWW) is by means of computer, one can use computers to "observe" and record user actions, to communicate with users and use all collected information to help the users find their way in the WWW. This thesis concentrates on machine learning techniques that can be applied on collected information to help the user. The work is in great deal inspired by WebWatcher: "a Learning Apprentice for the World Wide Web" [7, 39] and some other work related to learning apprentice and learning from text [40, 54, 62, 74]. The idea of a learning apprentice is to automatically customize to individual users, using each user interaction as a training example.

1.2 Related approaches

This thesis is in the area of machine learning where learning is over text documents that are mostly written in HTML-format. Even though the learning problem definition is a part of research work in the thesis, it is clear in advance that it will be characterized by the high number of features (several tens of thousands). Additionally, documents are connected to each other in directed graphs via HTML-*hyperlinks*. The source of data is non-homogeneous, distributed and dynamically changing, there is potentially infinite number of training examples and user interests can be changing over time. The problem is not only space intensive but also time demanding, since we have to do classification while user is waiting to get the requested document.

Most of the work in learning on Web documents defines machine learning problem similar to information filtering, asking users to rank given documents and generating the model of user interests based on the users ranking (Armstrong et al. [7], Balabanović et al. [9], Pazzani et al. [83, 84]). The proposed thesis is inspired by the idea of a user customized system, that is present whenever the user is browsing the Web but does not demand any additionally efforts from the user (no ranking or search keywords required).

There is quite some work in the area of learning over text documents that is not necessarily related to the Web. We will take a look at some of these work through the prism of three questions that we find important for the research described in this thesis: in Section 1.2.1 what representation is used for documents, in Section 1.2.2 how is the high number of features dealt with and in Section 1.2.3 which learning algorithm is used. Table 1.1 summarizes them over some related papers in order to give an idea about the current trends.

1.2.1 Document representation

The frequently used document representation in Information Retrieval and text-learning is the so called $TFIDF$-vector representation. It is a *bag-of-words representation*: all words from the document are taken and no ordering of words or any structure of text is used (see Figure 1.1). Since most of our documents are in HTML format, there is a well defined underlying structure that could be used. There is also additional information in plain text, for example structure of sentences, position of words or neighboring words. The question is how much we can gain considering additional information in learning (and what information to consider) and what is the price we have to pay for it? There

Research paper reference	Document Representation	Feature Selection	Learning algorithm
Apté et al. [6]	bag-of-words (freq)	stop list+ frequency	Decision Rules
Armstrong et al. [7]	bag-of-words	informativity	TFIDF Winnow, WordStat
Balabanović et al. [9]	bag-of-words (freq)	stop list+stemming+ keep 10 best words	TFIDF
Bartell et al. [11]	bag-of-words (freq)	latent semantic indexing using SVD	—
Berry et al. [12] Foltz and Dumais [28]	bag-of-words(freq)	latent semantic indexing using SVD	TFIDF
Cohen [21]	bag-of-words	infrequent words pruned	Decision Rules ILP
Joachims [40]	bag-of-words (freq)	infrequent words+ informativity	TFIDF, PrTFIDF, Naive Bayes
Lam et al. [60]	bag-of-words (freq)	mutual info.	Bayesian Network
Lewis et al. [66]	bag-of-words	log likelihood ratio	logistic regression with Naive Bayes
Maes [69]	bag-of-words+ header info.	mail/news header + selecting keywords	Memory-Based reasoning
Pazzani et al. [83, 84]	bag-of-words	stop list+ informativity	TFIDF, Naive Bayes, Nearest Neighbor, Neural Network, Decision Trees
Sorensen and Mc Elligott [97, 25]	n-gram graph (only bigrams)	weighting graph edges	connectionist combined with Genetic Algorithms
Yang [100]	bag-of-words	stop list	adapted k-Nearest Neighbor

Table 1.1: Document representation, feature selection and learning algorithms used in some related work. Bag-of-words is used on Boolean features unless notified else: (freq) — frequency of word in a document is used instead of Boolean information saying if word occurred in a document.

is currently no well studied comparison or directions that we are aware of. There is some evidence in Information Retrieval research, that for long documents, considering information additional to bag-of-words is usually not worth efforts. But our documents are mostly HTML-documents on the Web and they are not especially long! Additionally we learn on the shortened documents each describing a hyperlink and usually containing 5-30 words.

Many current systems that learn on text use the bag-of-words representation using either Boolean features indicating if specific word occurred in document (eg. [7, 21, 66, 83, 84, 100]) or frequency of word in a given document (eg. [6, 9, 11, 12, 40, 60, 100]). There is also some work that uses additional information such as word position [21] or word tuples called n-grams [97].

We studied the bag-of-words representation enriched with domain dependent background knowledge. More about our approach to document representation is given in Section 3.1.

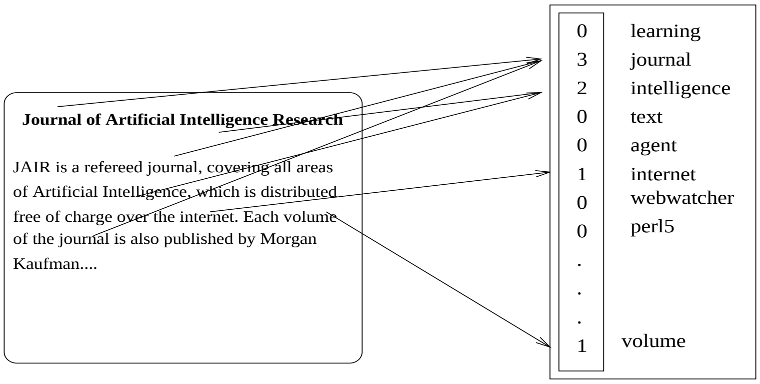

Figure 1.1: Example of bag-of-words representation using frequency vector.

1.2.2 Feature selection

In the simple document representation, when we ignore all the additional information and use only words in the described bag-of-words approach, we still end up with several tens of thousands of different words that occur in documents. Not only is using all these words time-consuming but also many of them are not really important for our learning task and their usage can degrade the system performance.

One of the frequently used approaches to reduce the number of different words is to use the "stop-list" containing common English words (eg. a, the, with) or pruning the infrequent and/or very frequent words (eg. [21, 40]). Connected to the particular language is also word stemming that reduces number of different words using a language-specific stemming algorithm. Many approaches use language independent approach and introduce some sort of word scoring measure in order to select only the best words (eg. [6, 7, 9, 40, 60, 66, 83, 84]) or reduce dimensionality using latent semantic indexing with singular value decomposition (eg. [11, 12, 28]). There are machine learning techniques developed specially for feature selection (eg. [18, 96, 42]) but most of them are not appropriate for the situations with several tens of thousands of features.

More about feature subset selection approaches including word scoring measures and our experimental results is given in Chapter 4.

1.2.3 Document classification

One of the well-established techniques for document classification in Information Retrieval is to represent each document as a $TFIDF$-vector in the space of words that appeared in training documents [93], sum all interesting document vectors and use the resulting vector as a model for classification (based on Rocchio relevance feedback method [90]). Each component of a document vector $d^{(i)}$ is calculated as the product of $TF(W_i, Doc)$ (Term Frequency — number of times word W_i occurred in document Doc) and $IDF(W_i) = log\frac{D}{DF(W_i)}$ (Inverse document Frequency), where D is the number of documents and document frequency $DF(W_i)$ is the number of documents word W_i occurred in at least once.

$$d^{(i)} = TF(W_i, Doc)IDF(W_i)$$

The exact formula used in different approaches may slightly vary (some factors are added, normalization performed [93]) but the idea remains the same. A new document is then represented as a vector in the same vector space as the generated model and the distance between them is measured (usually defined as a cosine of angle between vectors) in order to classify the document. This technique has already been used in Machine Learning experiments on World Wide Web data (eg., [7, 9, 12, 40, 83, 84]).

There are also some other techniques for model generation that have been used in text-learning. Armstrong et al. [7] used a statistical approach they called WordStat. In experiments by Pazzani et al. [83, 84] a Naive Bayesian classifier is used on Boolean vectors. They also used Nearest Neighbor and symbolic learning using Decision Trees. A variant of k-Nearest Neighbor was also used by Yang [100]. Joachims [40] introduced Probabilistic-$TFIDF$ that takes into account document representation. He also used Naive Bayesian classifier as described in [40, 73]. Apté et al. [6] used Decision Rule and observed that in the case of different topics being categories, it is better to select features for each given topic (using stop-list and frequency scoring measure) than for all topics at once. We use this finding in our document categorization experiments described in Chapter 5. Cohen [21] used Decision Rules and the Inductive Logic Programming systems FOIL and FLIPPER. Lewis and Gale [66] used a combination of a Naive Bayesian classifier and logistic regression. Maes [69] used Memory-Based reasoning, McElligot and Sorensen [25, 97] used a connectionist approach combined with Genetic Algorithms, Lam et al. [60] used Bayesian Network Induction.

In this thesis a naive Bayesian classifier and k-Nearest Neighbor algorithm are used as described in Section 3.2.

1.3 Addressed research problems and overview

Addressed research problems are connected to the usage of machine learning techniques on text data with sparse coverage of example space. Applying machine learning to our problem domain pointed out several research problems. Chapter 2 describes experimental methods and materials we used. Section 2.1 gives our problem definition. Text documents (and HTML-hyperlinks represented as shortened documents) are used here as examples. Section 3.1 gives examples representation. It is based on the commonly used bag-of-words representation that we enrich here by adding word sequences. Learning algorithms that we use on feature-vectors are described in Section 3.2. In order to validate and compare different approaches we need some appropriate model quality estimate. The specific of our problem is highly unbalanced class probability distribution and asymmetric misclassification costs. By asymmetric misclassification costs we mean that one of the class values is the target class value for which we want to get predictions and we prefer false positive over false negative. Please notice that there is no cost matrix given. It is our understanding of problem and our goal that indicated asymmetric misclassification costs. Section 2.3 gives a description of model quality estimates and suggests several appropriate estimates that are used in our experiments.

One of the important issues for our problem is feature subset selection. We are dealing with several tens of thousands of features, many of them being irrelevant or redundant or misleading. It is shown in an experimental comparison that in our domains, including all features gives worse results than using a carefully chosen feature subset. Overview of different approaches to feature subset selection is given in Section 4.1. The feature subset selection commonly used in learning on text data requires a feature scoring measure. In Section 4.2 we give description of some existing and propose several new feature scoring measures. An experimental comparison of the described feature scoring measures is given in Section 4.3.

Document representation is extended with the usage of word sequences as described in Section 3.1.1, so that we generate new features for machine learning representing word sequences instead of having only features representing single words as commonly used in text-learning. Document topic categories obtained from the constructed background knowledge can be also used as new features. Dynamic construction of background knowledge from the existing Web hierarchy is described in Chapter 5. It requires document categorization based on a large text hierarchy that needed development of a new machine learning approach for learning on class hierarchy. This new approach includes learning on large text data based on the methods developed in this thesis.

Specific to our problem is that classification is on shortened HTML-documents that describe hyperlinks. Since each hyperlink is assigned a document, we set the hypothesis that some of the document content can be predicted from the corresponding hyperlink. This requires learning of mutually dependent class attributes, where each class attribute corresponds to a feature, eg., predicting the occurrence of a word. Preliminary experiments on learning mutually dependent class attributes are given in Chapter 6. Evaluation of the proposed methods on real data was performed using a developed prototype system called Personal WebWatcher [77] for user customized Web browsing and developed a machine learning system called LEARNING MACHINE [33]. Conclusions and discussion is given in Chapter 7.

1.4 Contributions of the thesis

Contributions of this thesis are the following:

- Design of elements of machine learning techniques for learning on large sources of unstructured data (text) with sparse coverage of example space, such as HTML-documents on WWW. This includes:
  - The development of a machine learning method for learning from class hierarchies. The method handles a high number of class values, examples and features, by dividing a problem into subproblems based on the hierarchical structure of class values and examples, by applying feature subset selection and by pruning unpromising class values during the classification. Pruning unpromising class values during the classification is based on the minimal number of highly scored fetures shared by the subproblem classifier and the testing example. This pruning improves classification results while pruning 85%-95% of class values.
  - An extension of document representation that extends the bag-of-words representation by using word sequences generated in an efficient way. Using word sequences of length up to 2 instead of using only single words is more important for the classification results than using longer word sequences.
  - Comparison and analysis of different feature scoring measures for feature subset selection on text data with several tens of thousands of features and unbalanced

class distribution; and several extensions of existing feature scoring measures which are appropriate for text domains with asymmetric misclassification costs. By asymmetric misclassification costs we mean that one of the class values is the target class value for which we want to get predictions and we prefer false positive over false negative. The best results over domains and different number of selected features are achieved when features are scored using Odds ratio and only 50-100 best features are used (using 0.2% to 5% of all domain features).

- Evaluation of the proposed guidlines on real-world data collected from the Web:

  1. WWW-documents collected from residential users in Carnegie Mellon University (CMU) HomeNet project,
  2. domains generated from the Yahoo hierarchy of WWW-documents.

Chapter 2

Experimental methods and materials

2.1 Problem definition

This thesis deals with three text-learning problems connected to: helping user browsing the Web, document categorization and document content prediction. For all the three problems we use data collected from the Web and use shortened documents as training examples. We build descriptions of hyperlinks, and treat these as shortened documents. Description of the first problem including justification for using the shortened documents is given in this Section. The second problem is described in Section 5.1 while the third problem is described in Section 6.1. Experimental results are given separately for each of the three problems and new solutions in the corresponding Chapters together with the description of the problem or the used approach (Chapters 2, 3, 4, 5, 6).

Helping the user browsing the Web is performed here by highlighting interesting hyperlinks on the requested Web documents. We assume that the interesting hyperlinks are the hyperlinks that are highly probable to be clicked by the user. Our problem is defined as predicting clicked hyperlinks from the set of Web documents visited by the user. All hyperlinks that are present on the visited documents are used for constructing machine learning examples. Each hyperlink includes some underlined text pointing to a document (text, image,...). We consider two approaches that differ in the information included in training examples: (1) include information presented to the user, that is a part of the text from the document that contains a hyperlink and (2) include information that was not presented to the user, that is the content of the document pointed to by a hyperlink.

(1) Generating examples from the information presented to the user on the requested document means using machine learning to model the function $User_{HL} : HyperLink \rightarrow \{pos, neg\}$, where $HyperLink$ (HL) is defined to include the actual hyperlink from HTML-document and some of the its context (titles, neighboring words).

(2) Retrieving documents pointed to by hyperlinks on a requested document means that the model will be induced from documents $User_{DOC} : Document \rightarrow \{pos, neg\}$. The reason we can afford thinking about retrieving documents is that the classification model is induced "off-line", meaning that the induction is not so time critical as classification. The time critical part is classification, since usually the user is sitting behind the Web browser waiting for the requested document. Before showing the document to the user, we

are modifying it by highlighting some of the included hyperlinks according to the results of the classification process in which the document is used.

In both cases, classification is performed on *HyperLinks* that in the second solution can be viewed as a kind of HTML-documents. In (1) the model induced from *HyperLinks* is used for classification while in (2) the model induced from the *Documents* pointed to by hyperlinks is used. In this thesis we use the first approach that requires less time to collect data and results in domain containing less features (smaller vocabulary is used).

2.1.1 What information to include in examples?

As already pointed out training examples for predicting clicked hyperlinks can include different information. The available input data contain the set of Web documents visited by the user. Additionally, it is possible to get other data available on the Web, for instance, documents connected to the given visited documents.

We give experimental comparison of the two described approaches (induce $User_{HL}$ and induce $User_{DOC}$) on the data collected for two users participating in the HomeNet project [37]. For each user we have about 3000 examples and more than 90% examples have the majority class value. More detailed data characteristics are given in Table 2.1.

We report average results of 10 repetitions of model generation and testing. In each repetition 30 % of randomly selected examples were hold-out for the testing (see Section 2.3). Classification is performed using a naive Bayesian classifier as described in Section 3.2.1 using the selected feature subset, where each feature represents one word from the training documents. Feature subset is selected as described in Section 4.1.3 using Odds ratio to score the features. Additionally we show results of using Information gain instead of Odds ratio. Results in Figures 2.1 and 2.2 show the influence of the vector size (vocabulary size expressed as the number of selected features) to Classification accuracy (see Equation 2.1) and Information score (see Equation 2.8). Since we have domains with unbalanced class distribution (over 90% of examples have the majority class value), we found Information score (right hand-side graphs) more useful than Classification accuracy (left hand-side graphs). Reported classification accuracy is on our domains the same or lower than the accuracy of default strategy (predicting negative class value). On the other hand, Information score measures the amount of information gained by using the induced model instead of default strategy. Information score is negative if the predicted probability of the correct class value is lower than the prior probability of the same value (returned by default strategy).

Our hypothesis was that the first model described in (1), that is based only on information presented to the user (`NBayesHL` lines on graphs) is worse than the second model described in (2), generated using information in document pointed to by a hyperlink (`NBayesDoc` lines on graphs). The idea was that by using whole documents instead of only hyperlinks that are pointing to the documents should give better results. This hypothesis is not confirmed by our experiments. In contrary, the first model achieved comparable or better results than the second model. For both domains, the maximum information score is positive and it is achieved using the first model and about 50 best features selected according to Odds ratio.

At the same time, the first model requires less time and space, since there is no need to collect documents that are pointed to by the hyperlinks. This also means that the model is based on the shortened documents (hyperlink descriptions) and consequently uses smaller

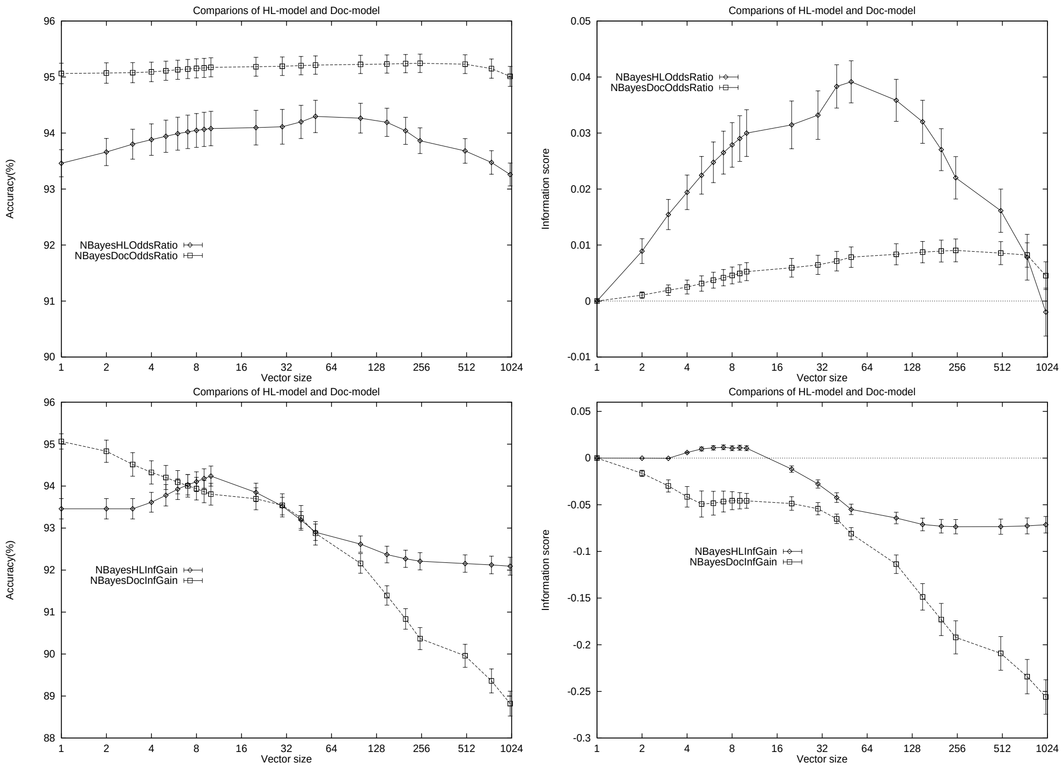

Figure 2.1: Comparison of classification accuracy and information score of HL-model (induced from the hyperlink description) and Doc-model (induced from the content of documents pointed to by hyperlinks) on data for the HomeNet user150101. The problem is defined as predicting clicked hyperlinks from the set of documents visited by the user. Given are results for two feature scoring measures: Odds ratio (upper graphs) and Information gain (lower graphs).

number of features. For instance, the first row in Table 2.1 gives characteristics of data used for the first model and the second row gives characteristics of data used for the second model. Both data sets are formed from the set of Web documents visited by the same user. The number of unique words (last column) is here the same as the number of features, since we define a feature for each word. For the first model we have 3 435 unique words while for the second model we have 42 012 unique words.

Our explanation for the success of the first model described in (1) is that most of the hyperlink descriptions HL as defined in Section 3.1.1 contain carefully selected information. In this way most training examples HL contain human selected information characteristic for the document the hyperlink points to. The other factor could be that predicting clicked hyperlink means predicting action of the user that does not see documents pointed to by the hyperlinks (that are used by the second model) but only the hyperlinks themselves (used by the first model). In the rest of this thesis, we are learning on shortened documents describing hyperlinks (as used here by the first model). The relation between the data used in the two models and potential prediction of document content from hyperlink is addressed in Chapter 6.

Experimental comparison of feature scoring measures given in Section 4.3 shows in our domains the advantage of Odds ratio, the measure known in information retrieval, over Information gain, the well known measure in machine learning. The experiments presented here confirm this finding. Namely, the usage of Odds ratio results in lower sensitivity to changes in vector size and equal or better performance than the usage of Information gain (compare the lower and the upper graphs in Figures 2.1 and 2.2).

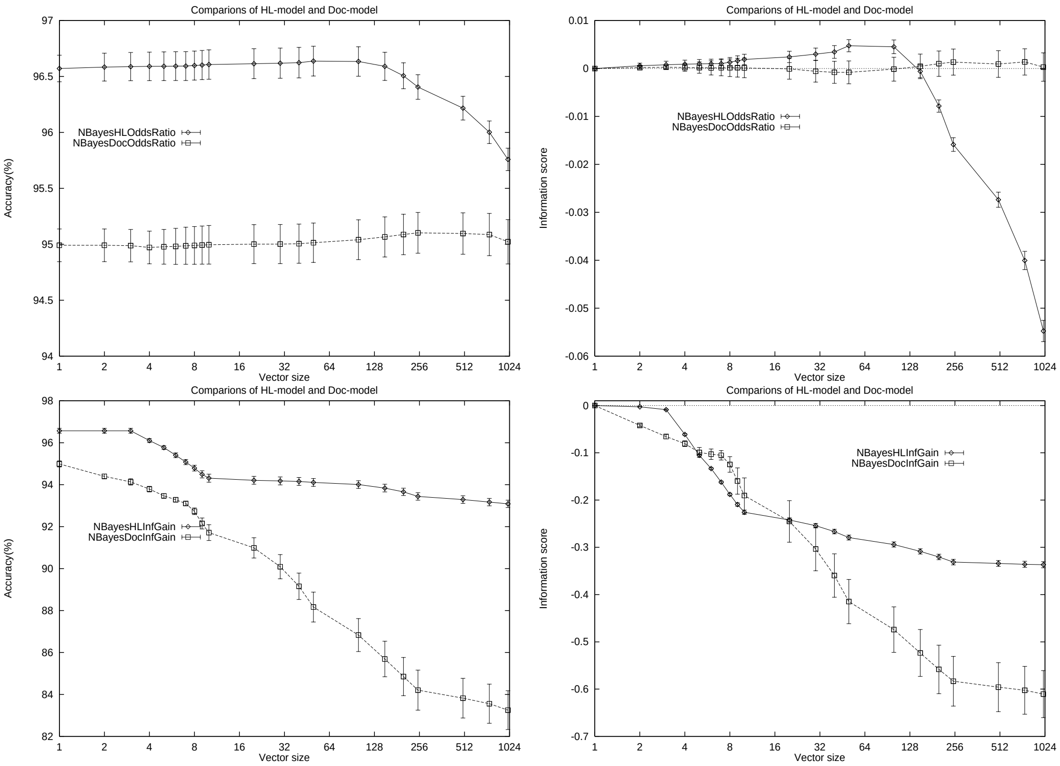

Figure 2.2: Comparison of classification accuracy and information score of HL-model (induced from the hyperlink description) and Doc-model (induced from the content of documents pointed to by hyperlinks) on data for the HomeNet user150211. The problem is defined as predicting clicked hyperlinks from the set of documents visited by the user. Given are results for two feature scoring measures: Odds ratio (upper graphs) and Information gain (lower graphs).

2.2 Description of data sets

Evaluation of the methods proposed in the thesis is performed on the real data collected from the Web:

1. WWW-documents collected from residential users in Carnegie Mellon University HomeNet project,

2. domains generated from the Yahoo hierarchy of WWW-documents.

2.2.1 Domains generated from the HomeNet project data

The goal of the HomeNet project [37] is "to increase our knowledge about the use and impact of residential electronic services". Project includes about 50 Pittsburgh families that were given computers, connection to Internet and a starting Web page with links to information sources tailored to the family members' identified interests. The data used in this thesis represent only a part of the data collected in the project from May 1996 to May 1997. What we use here is a computer generated record of Web sites visits separated for each user.

The machine learning problem is here defined as predicting clicked hyperlinks on new Web documents requested by the user. Training examples are obtained from the set of Web documents visited by the user. From the data characteristics given in Table 2.1 it can be seen that we have problems with highly unbalanced class distribution, where the probability of positive class value is in most cases < 0.1. The difference in the number of examples and the probability of positive class value when using HL or Doc for the same user is due to the fact that we are using all the hyperlinks but only some (text/html) of the corresponding documents. Since some of the hyperlinks point to non-text documents (eg., post script, image), the number of examples for the first model (HL data) is higher then the number of examples for the second model (Doc data). The problem is also characterized by a large number of features formed from the several tens of thousands of unique words that occurred in positive and negative documents. It can be also seen that the data collected for the generation of the first model (as described in Section 2.1) have much smaller number of unique words (3 435 and 4 392) than the data used for the generation of the second model (42 012 and 68 470). This indicates that the data for learning the first model by using shortened documents (HL) probably contains less information or the similar information represented in a more compact way. Experimental comparison of the two models given in Section 2.1.1 shows that in the case of shortened documents we are dealing with better quality information and achieving comparable or better results than using the second, much larger model.

UserId and data source	probability of pos. class value	number of examples	number of unique words
usr150101			
HL	0.067	2 903	3 435
Doc	0.052	2 351	42 012
usr150211			
HL	0.036	3 654	4 392
Doc	0.051	3 528	68 470

Table 2.1: Data characteristics for document (Doc) and hyperlink (HL) data for two HomeNet users. For each user the set of visited Web documents is used to form the set of machine learning examples. The problem is to predict clicked hyperlink based on the previous user actions.

2.2.2 Domains generated from the Yahoo hierarchy of WWW-documents

Domains we generated from the Yahoo hierarchy of WWW-documents represent five out of the fourteen top level Yahoo categories: *'Arts and Humanities'*, *'Entertainment'*, *'Computers and Internet'*, *'Education'*, *'References'*. The problem is defined as predicting document category from the documents already categorized into a text hierarchy. The data was obtained from the publicly accessible Yahoo Web site. Table 2.2 gives domain characteristics including information about the number of nodes in the domain hierarchy, the number of features showing also how many features represent different length word sequences, the number of examples (documents), the average distance between nodes measured as the average number of connections between any two nodes in the hierarchy and the average number of features in positive documents. The average number of features in positive documents is calculated as the average over the defined domain subproblems. A binary subproblem is defined for each node (subcategory) in the hierarchy as described in Section 5.2. Features are generated not only for single words (referred here as 1-grams) but also for word sequences (n-grams) as described in Section 3.1.1. The longest word sequences we are using contain 5 consecutive words (referred to as 5-grams).

Yahoo category	# of sub-ctgs. (nodes)	# of features (1-grams+...+5-grams)	# of examples (actual docs)	Avg. node distance	Avg. pos. features
'Entertainment'	8,081	30,998 (15,144+11,211+ 2,970+1,059+505)	79,011	7.56	60
'Arts and Humanities'	3,085	11,473 (7,380+3,538+ 463+75+17)	27,765	6.59	65
'Computers and Internet'	2,652	7,631 (5,049+2,276+ 261+38+7)	23,105	6.77	55
'Education'	349	3,198 (1,919+1,061+ 184+28+6)	5,406	4.54	100
'References'	129	928 (701+196+ 28+3+0)	1,995	4.29	45

Table 2.2: Domain characteristics for five domains formed from the five top Yahoo categories. The problem is document categorization based on the hierarchy of categories and the corresponding documents. We give for each domain (from left to right) its name, the number of included subcategories, the number of features, the number of examples (actual Web documents the category is based on), the average distance between two nodes in the hierarchy (length of the shortest path between the two nodes), the average number of features in positive examples.

More specifically, the following domains are used in our experiments. *'Entertainment'* having 8,081 nodes with an average distance 7.56 between them, 30,998 features consisting of 15,144 1-grams, 11,211 2-grams, 2,970 3-grams, 1,059 4-grams, 505 5-grams and 79,011 actual Web documents. *'Arts and Humanities'* having 3,085 nodes with an average distance 6.59 between them, 11,473 (7,380+3,538+463+75+17) features and 27,765 actual Web documents. *'Computers and Internet'* having 2,652 nodes with an average distance 6.77 between them, 7,631 (5,049+2,276+261+38+7) features and 23,105 actual Web documents. *'Education'* having 349 nodes with an average distance 4.54 between them, 3,198 (1,919+1,061+184+28+6) features and 5,406 actual Web documents. *'References'* having 129 nodes with an average distance 4.29 between them, 928 (701+196+28+3+0) features and 1,995 actual Web documents. We test our approach on domains that are parts of the Yahoo hierarchy, since they are hierarchies themselves with similar characteristics but smaller.

From the data characteristics given in Table 2.2 it can be seen that the number of unique words (1-grams) varies from 701 for domain *'References'* to 15,144 for domain *'Entertainment'*. The set of negative examples is the same for all subproblems in one domain. The main reason for this decision is efficiency supported by the fact that each subproblem can be seen as differentiation of its positive examples from the mass of examples. The set of positive examples changes for each of the subproblems. For most of the subproblems the probability of positive class value is < 0.1, meaning that we have domains with unbalanced class distribution.

2.3 Model quality estimation

In machine learning the result of learning is some knowledge model. We will use M to refer to this model. One important part of machine learning is evaluation of generated models, that is usually performed on independent set of examples (testing set) in repeated testing experiments. The commonly used methods are hold-out and cross-validation. In both methods, data set is divided into two parts: training set and testing set. The first is used to generate the model and the second is used to estimate the quality of the generated model. This loop is usually repeated (eg., 5, 10 or 25 times) and an average quality is reported. In hold-out experiments random subset of predefined size is taken out from the data to be used as the testing set while the rest of the data is used as the training set. In cross-validation experiments the data is divided into as many subsets as we want to have repetitions of model generation-testing loop. In each repetition, one of the subsets is taken as the testing set and the union of all other subsets is used as the training set.

When classifying examples into discrete-valued class, most of the quality estimates can be calculated out of the contingency table generated from the classification results. Each value in the contingency table is a number of examples with a certain combination of actual and predicted class value. As an example for formula description, we will use binary-valued class and the contingency Table 2.3. In this table, a is the number of examples having class value 'positive' that are also classified as positive (true positive), b is the number of examples having class value 'negative' that are classified as positive (false positive), c is the number of examples having class value 'positive' that are classified as negative (false negative), d is the number of examples having class value 'negative' that are also classified as negative (true negative).

2.3.1 Classification accuracy

In machine learning commonly used model quality estimate is the probability of correct classification usually referred to as $Classification\ accuracy$. Classification accuracy of model M can be estimated with the following formula:

$$Accuracy(M) = \sum_{ex} P(ex) \times Accuracy(M, ex) = P(\widehat{C}(ex) = C(ex)) \qquad (2.1)$$

	Positive examples	Negative examples
Examples predicted positive	a (true positive)	b (false positive)
Examples predicted negative	c (false negative)	d (true negative)

Table 2.3: Contingency table for binary-valued class containing number of examples in each of the four possible combinations for actual and predicted class value.

$$Accuracy(M, ex) = \begin{cases} 1; & \widehat{C}(ex) = C(ex) \\ 0; & otherwise \end{cases}$$

Where summation goes over examples (usually testing examples), $P(ex)$ is the probability of example ex that is usually equal for all examples ($\frac{1}{n}$, where n is the number of examples), $C(ex)$ is the actual class value of example ex, $\widehat{C}(ex)$ is the class value of example ex predicted by model M.

For the binary-valued class problem using the notation from Table 2.3, Classification accuracy of model M is estimated as $Accuracy(M) = \frac{a+d}{a+b+c+d}$. Where a, b, c, d are the elements of the example contingency Table 2.3. Generalization of Classification accuracy for any number of class values, can be estimated from a contingency table dividing the sum of diagonal elements (number of correctly classified examples) by the sum of all table elements: $Accuracy(M) = \frac{\sum_i x_{ii}}{\sum_i \sum_j x_{ij}}$. Where x_{ij} is element in the i-th row and the j-th column of a contingency table (eg., b in Table 2.3 would be referred to as x_{12}).

2.3.2 Precision and Recall

Precision and Recall are quality estimates for binary-valued class problems that are commonly used in Information retrieval. In ROC (receiver operating characteristics) analysis as used in [59] sensitivity and specificity are used, where sensitivity equals Recall as defined here while specificity equals Recall calculated on the non-target class value. The common problem in Information retrieval with two possible class values is finding relevant documents (class value: relevant, irrelevant). One of the class values (relevant document) is the target class value for which Precision and Recall should be maximized. An ideal system would have Precision and Recall of 1. Perfect Precision can be achieved by a system that classifies no example into the target class value, while perfect Recall can be achieved by a system that classifies all examples into the target class value. Thus, using just one of the estimates does not provide interesting results. Usually an average precision is given measuring Precision on different levels of Recall or a combination of Precision and Recall is used known as F-measure. In our evaluation we use $F - measure$. Evaluation measures for text-classification systems are discussed in more details in [65].

Precision is defined as the proportion of correctly classified examples in the set of all examples assigned to the target class value. Precision of model M for the target class value $targetC$ can be estimated using the following formula:

$$Precision(M, targetC) = P(targetC|\widehat{targetC}) \tag{2.2}$$

Where $targetC$ denotes that the actual class value is the target class value, $\widehat{targetC}$ denotes that the predicted class value is the target class value.

For the binary-valued class problem using the notation from Table 2.3, Precision of model M can be estimated as $Precision(M) = \frac{a}{a+b}$. In the case of singularity $a+b=0$ there are no examples classified to the target class value. Thus there are also no errors in predictions and by definition $Precision(M) = 1$.

Precision has similar meaning as Classification accuracy, the difference is that Precision is Classification accuracy estimated only from examples that are predicted to have the target class value ($\widehat{targetC}$).

$$\begin{aligned}
Accuracy(M) &= \sum_i P(\widehat{C_i}) \times Precision(M, C_i) \\
&= \sum_i P(\widehat{C_i}) \times P(C_i|\widehat{C_i}) \\
&= \sum_i P(\widehat{C_i}) \times \frac{P(C_i, \widehat{C_i})}{P(\widehat{C_i})} \\
&= \sum_i P(C_i, \widehat{C_i}) \\
&= P(C(ex) = \widehat{C}(ex)); equals(2.1)
\end{aligned}$$

Where $P(\widehat{C_i})$ is the probability that the predicted class value is C_i, $P(C_i|\widehat{C_i})$ is the conditional probability that the actual class value is C_i given that the predicted class value is C_i.

Recall is defined as the proportion of correctly classified examples out of all the examples having the target class value. Recall of model M for the target class value $targetC$ can be estimated using the following formula:

$$Recall(M, targetC) = P(\widehat{targetC}|targetC) \tag{2.3}$$

For the binary-valued class problem using the notation from Table 2.3, Recall of model M can be estimated as $Recall(M) = \frac{a}{a+c}$. In the case of singularity $a+c=0$ there are no examples with the target class value and by definition $Recall(M) = 1$. Notice that $Recall(M, targetC) = Precision(M, targetC) \times \frac{P(\widehat{targetC})}{P(targetC)}$

Classification accuracy can be expressed as a combination of Precision and Recall on the two class problem using only the target class value as follows:

$$\begin{aligned}
Accuracy(M) &= \sum_i P(\widehat{C_i}) \times Precision(M, C_i) \\
&= P(\widehat{targetC}) \times Precision(M, targetC) + P(\widehat{nontargetC}) \times \\
&\quad Precision(M, nontargetC) \\
&= P(\widehat{targetC}) \times P(targetC|\widehat{targetC}) + P(\widehat{nontargetC}) \times \\
&\quad P(nontargetC|\widehat{nontargetC}); using(2.2) \\
&= P(targetC, \widehat{targetC}) + P(\widehat{nontargetC}) \times (1 - P(targetC|\widehat{nontargetC})) \\
&= P(targetC, \widehat{targetC}) + P(\widehat{nontargetC}) \times (1 - \frac{P(targetC, \widehat{nontargetC})}{P(\widehat{nontargetC})})
\end{aligned}$$

$$
\begin{aligned}
&= P(targetC, \widehat{targetC}) + P(\widehat{nontargetC}) \times \\
&\quad \frac{P(\widehat{nontargetC}) - (P(targetC) - P(targetC, \widehat{targetC}))}{P(\widehat{nontargetC})} \\
&= P(targetC, \widehat{targetC}) + (1 - P(\widehat{targetC}) - P(targetC) + P(targetC, \widehat{targetC})) \\
&= 2 \times P(targetC, \widehat{targetC}) + 1 - P(\widehat{targetC}) - P(targetC) \\
&= P(\widehat{targetC}) \times Precision(M, targetC) + P(targetC) \times Recall(M, targetC) + \\
&\quad 1 - P(\widehat{targetC}) - P(targetC); using (2.2), (2.3)
\end{aligned}
$$

Where all the probabilities are estimated using model M or training data used to generate model M. It can be seen that Classification accuracy does not take into account possible asymmetry in misclassification costs. In the above formula for Classification accuracy, Precision and Recall are weighted by the predicted and actual probability respectively. F-measure given in Equation (2.4) in contrary includes a parameter that sets the relative importance of Precision and Recall. We show in Section 2.4 that this is an important issue for our problems.

$F - measure$ is a combination of Precision and Recall commonly used in information retrieval, where the relative importance of each is expressed with the value of parameter β.

$$F_\beta(M, targetC) = \frac{(1+\beta^2) Precision(M, targetC) \times Recall(M, targetC)}{\beta^2 Precision(M, targetC) + Recall(M, targetC)} \tag{2.4}$$

F_0 is the same as Precision, F_∞ is the same as Recall, β can have values between 0 and ∞. Usually results are given for β values of 0.5 (we care more about Precision then about Recall), 1.0 (we don't have preferences regarding Precision or Recall) and 2.0 (we care more about Recall then about Precision). In our experiments we use F_2, since in our domains we care more about Recall than about Precision. Given that in F_β there is a trade off between Recall and Precision, we selected a value of β that would favor higher Recall.

For the binary-valued class problem using the notation from Table 2.3, F-measure for model M can be expressed as $F_\beta(M) = \frac{(1+\beta^2)a}{(1+\beta^2)a+b+\beta^2 c}$. In the case of singularity ($a+b=0$ and $a+c=0$) the value of $F_\beta = 1$ and in the case of ($a+b=0$ and $c+d>0$) the value of $F_\beta = 0$.

For the binary-valued class problem $Square\ error\ in\ counting$ (SEC) [65] can be used to estimate error of model M. It is a square difference between the number of examples predicted to have the target class value and the actual number of examples with the target class value.

$$SEC(M) = (N_{\widehat{targetC}} - N_{targetC})^2 \tag{2.5}$$

Where $N_{\widehat{targetC}}$ is the number of examples predicted to have the target class value, $N_{targetC}$ is the actual number of examples with the target class value. Using the notation from Table 2.3, Square error in counting can be expressed as $(b-c)^2$.

2.3.3 Geometric mean of Recall and Precision

Geometric mean of recall is proposed in [57] under the name 'geometric mean of accuracy' as a model quality measure for domains with unbalanced training set (unbalanced class distribution). We decided to change the name since in our opinion, this measure relates to Recall using geometric mean of Recall for different class values (see Section 2.3.2 for the definition of Recall).

$$GeoRecall(M) = \sqrt[Cls]{\Pi_i^{Cls} Recall(M, C_i)} \quad (2.6)$$

Where Cls is the number of class values, $Recall(M, C_i)$ is the Recall of model M on the target class C_i.

For the binary-valued class problem using the notation from Table 2.3, Geometric mean of recall for model M can be estimated as $GeoRecall(M) = \sqrt{\frac{a}{a+c} \times \frac{d}{b+d}}$.

Geometric mean of recall is based on Recall estimated for each of the class values. In similar way *Geometric mean of precision* that is based on Precision can be defined:

$$GeoPrecision(M) = \sqrt[Cls]{\Pi_i^{Cls} Precision(M, C_i)} \quad (2.7)$$

2.3.4 Information score

Information score is proposed in [50] as a model quality measure that excludes the influence of prior probabilities and takes into account probabilistic answers. Namely, the known problem with Classification accuracy is that it is influenced by prior probabilities. For instance, majority class value having prior probability of 0.9 means that classification accuracy of 90% can be achieved trivially with a model classifying all examples into the majority class value.

Information score measures the average amount of information about the actual class value given by the model. The idea is in measuring the difference in the belief that an example has its actual class value before and after using the generated model.

$$InfScore(M) = \sum_{ex \in TestSet} \frac{1}{n} InfScore(M, ex) \quad (2.8)$$

Where n is the number of testing examples, $InfScore(M, ex)$ is information score of an answer given by the model M for example ex.

Let $C(ex)$ be the actual class value of example ex, $\widehat{P}(C(ex))$ be the predicted probability of the actual class value and $P_a(C(ex))$ be the prior probability of the actual class value. In the definition of information score of an answer, two cases are considered. First when the predicted (posterior) probability of the actual class value returned by the model $\widehat{P}(C(ex))$ is greater than the prior probability of the same class value $P_a(C(ex))$, thus changing the probability in the right direction. Second case is when the predicted probability of the actual class value is lower than its prior probability, thus changing the probability in the wrong direction as if the example would not have its actual class value (thus in the formula $1 - P(C(ex))$ is used instead of $P(C(ex))$).

$$InfScore(M, ex) = \begin{cases} \log \frac{\widehat{P}(C(ex))}{P_a(C(ex))}; & \widehat{P}(C(ex)) > P_a(C(ex)) \\ -\log \frac{1-\widehat{P}(C(ex))}{1-P_a(C(ex))}; & \widehat{P}(C(ex)) \leq P_a(C(ex)) \end{cases} \quad (2.9)$$

2.4 Experimental setting

Our experiments are performed using our system Personal WebWatcher [77] and our recently developed machine learning system LEARNING MACHINE [33] that supports usage of different machine learning techniques on large data sets with especially designed modules for learning on text and collecting data from the Web. We report average results and standard error for 5 (in some cases 10) repetitions of testing on randomly selected independent data set, ie., using so called hold-out testing approach. Data descriptions given in Section 2.2 shows that we have unbalanced class distribution. In such situation Classification accuracy that is commonly used in machine learning experiments can give misleading results. For instance, in the problem of distinguishing between interesting and uninteresting Web documents, usually more than 95% of documents are uninteresting for a particular user. Consider domains given in Table 2.1. For the first domain the probability of the majority class value (negative) is 0.896 meaning that 89.6% Classification accuracy can be achieved by classifying all examples as negative. For such domains some more appropriate measure should be used eg., F-measure (2.4), Geometric mean of recall (2.6) or Information score (2.8).

Our problems are additionally characterized by the asymmetric misclassification costs (the unequal importance of the class values). For example, when predicting clicked hyperlink, it is better to highlight a few more than a few less hyperlinks (rather have false positive than false negative). In other words, the set of highlighted hyperlinks on a Web document is usually small and this helps user in filtering out the classification error (ignoring incorrectly highlighted hyperlinks). In the language of the described model quality measures, we care more about Recall than about Precision. In general, the result of the induced models can be further refined by the user by filtering out the best among the suggested hyperlinks (document categories, keywords assigned to a document). In this way the model can be used not only for prediction or categorization but also for filtering out a small, promising portion of all possibilities. Now that we decide about the appropriate model quality estimate, we should be aware of its characteristics. Our goal is to maximize the model quality that is estimated using F_2-measure. This means that the goal is to get good Recall and Precision on the target class value, that is in our case the minority class value. For the feature subset selection this means that the emphasis is on the features characteristic for the target class value. This is an important difference from the problems where we maximize Classification accuracy that is averaged over all class values. The consequences of this are evident from the experimental comparison of feature scoring measures used for feature subset selection and given in Section 4.3.

2.4.1 Illustration of different model quality estimates

In order to get an idea about characteristics of the described model quality measures, we give illustration on artificial models. Actual class values for twenty artificial examples are given in Table 2.4 together with the predicted class values that are result of classification obtained by seven artificial models. Here the target class value is value 1. The actual class values are set to approximately reflect the class probability distribution in our domains (90 % examples have class value 'negative'). The used seven artificial models are selected to reflect different characteristic model behavior. The model named 'zero' returns class value 0 for all examples, while the model named 'one' returns class value 1 (the target class value) for all examples. The model named 'oneish' assigned to most examples class value 1. The model named 'mix0.5' correctly classified a half of the examples from each class value, while 'mix' made few errors for both class values. The model named 'good0' correctly classified all but one example, that is incorrectly assigned value 0 (false negative). The model named 'good1' correctly classified all but one example, that is incorrectly assigned value 1 (false positive). Among these artificial models, the last model ('good1') would be the most appropriate for our problem. For the problem of predicting clicked hyperlinks described in Section 2.1, it can be interpreted as highlighting all it should and just a bit more than that.

Model	Class value for each example																			
actual	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
zero	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
one	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
oneish	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0
mix0.5	1	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0
mix	1	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
good0	1	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
good1	1	1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.4: Actual class values of 20 artificial examples and results of classification returned by characteristic artificial models. The incorrect class value predicted by artificial model are shown in boldface numbers.

Model	Accuracy	Recall	Precision	Geo Recall	Geo Precision	$F_{0.5}$	F_2	SEC
zero	**0.9**	0	**1**	$\sqrt{0 \times 1}$=0	$\sqrt{1 \times 0.9}$=**0.95**	0	0	4
one	0.1	**1**	0.1	$\sqrt{1 \times 0}$=0	$\sqrt{0.1 \times 1}$=0.32	0.12	0.5	324
oneish	0.6	**1**	0.2	$\sqrt{1 \times 0.56}$=0.75	$\sqrt{0.2 \times 1}$=0.45	0.24	0.55	64
mix0.5	0.5	0.5	0.11	$\sqrt{0.5 \times 0.56}$=0.53	$\sqrt{0.11 \times 0.91}$=0.32	0.13	0.29	81
mix	0.85	0.5	0.34	$\sqrt{0.5 \times 0.89}$=0.67	$\sqrt{0.33 \times 0.94}$=0.56	0.36	0.45	**1**
good0	**0.95**	0.5	**1**	$\sqrt{0.5 \times 1}$=0.71	$\sqrt{1 \times 0.95}$=**0.97**	**0.84**	0.56	**1**
good1	**0.95**	**1**	0.67	$\sqrt{1 \times 0.94}$=**0.97**	$\sqrt{0.67 \times 1}$=0.82	0.72	**0.91**	**1**

Table 2.5: Illustration of model quality measures on the results of seven artificial models. The best model quality for each measure is written in boldface numbers. Each of the artificial models is selected to reflect some characteristic behavior. The idea is to identify model characteristics favored by different model quality measures.

Table 2.5 contains model quality estimated on the classification results from Table 2.4 by eight of the previously described model quality measures. In our intuition models 'zero' and 'one' are both useless, while 'good0' and 'good1' return the best prediction, with 'good1' being more appropriate for our problems. The estimated quality of models given in Table 2.5 show some specifics of each model quality measure. Classification accuracy is correctly high for the two best model ('good0', 'good1') but it is also quite high for the model that predicts majority class value ('zero' model). The highest Recall is achieved by three models, even though model 'one' predicts minority class value and among the other two we prefer the second one (model 'good1'). In similar way the highest Precision is assigned to 'good0' as well as to 'zero' (notice that by definition of Precision, no classification to the target class value results with the highest Precision). Geometric mean of recall is the highest for 'good1', the most appropriate among the tested models for our problems. Geometric mean of precision is the highest for 'zero' (by definition) and 'good0'. $F_{0.5}$-measure is a combination of Recall and Precision, where we care more about Precision. The highest quality on our examples according to $F_{0.5}$-measure is assigned to model 'good0'. F_2-measure reflects that we care more about Recall than about Precision. On our examples the highest quality according to F_2-measure is assigned to model 'good1'. Square error of count SEC is the only measure used here that should be minimized. It gives the best results for the last three models 'mix', 'good0', 'good1' and the very bad results for 'one'. From all the compared measures only Geometric mean of recall and F_2-measure demonstrated behavior appropriate for our problem domain. In our experiments described in this thesis we included F_2-measure that is known from Information Retrieval and Information score that is known from Machine Learning.

Chapter 3

Document representation and learning algorithms

3.1 Representation of HTML-documents/hyperlinks

In order to apply machine learning methods on the problems of learning to predict the clicked hyperlink, document categorization and learning to predict document content we need to consider the representation of HTML-hyperlinks and HTML-documents. It is hard to capture the information offered to the user by showing her/him some Web-page. One of the very simple, frequently used but unfortunately rather limited approaches is the bag-of-words text document representation. In this representation, each text document is represented as a vector of numbers, each number corresponds to the occurrence of a particular word in the document (see Figure 1.1).

When classifying hyperlinks, different approaches can be used to decide which part of text to include in the hyperlink representation and how to represent it. Our prototype system named Personal WebWatcher described in Section 7.2 uses an approach similar to that of WebWatcher [7]. In WebWatcher the bag-of-words representation is used, where considered text consists of underlined words, words in the sentence containing the hyperlink, words in all the headings above the hyperlink and words given as search keywords by the user. Namely, WebWatcher is designed as a 'tour-guide' to help user searching information related to the given search keywords by highlighting hyperlinks that point to potentially relevant Web documents. Some later versions of the WebWatcher system consider also information from documents that are reachable from the hyperlink [41].

3.1.1 Enriched document representation

We represent each hyperlink as a shortened document, including underlined words, words in the window around the hyperlink and words in the headings above the hyperlink. In this way, both documents and hyperlinks are represented in the same way. As a basic representation we use bag-of-words approach that represents each document as a word-vector. We enriched the bag-of-words representation by adding new features generated from word sequences and document topic categories. The following two Sections describe generation of new features. In order to reduce the number of features we use feature subset selection as described in Chapter 4.

Using word sequences

We use word sequences, also known as n-grams, to form the new features. We generate features that represent up to five words (1-grams, 2-grams, ... 5-grams) occurring in a document as a sequence (eg. 'machine learning', 'world wide web'). We interleaved this feature generation with the feature selection using "stop-list" with common English words [29] and removal of infrequent features. First we remove words contained in the "stop-list" (eg. a, the, for, each, at, will, be) enabling some features that represent a word sequence to actually capture longer sequences. For instance, 'Word for Windows' that is represented as a 2-gram 'Word Windows' or 'winners will be posted at the end of each two-week period' that is represented as a 5-gram 'winners posted end two-week period'. We additionally reduce the high number of features by pruning infrequent features as suggested in [21, 40, 101]. Infrequent features are here defined as features (word sequences) that occur less than 4 times. By using word sequences we can capture some characteristic word combinations and also increase the number of features. For example, in the whole Yahoo hierarchy (without a top category *'Regional'*) that we use for automatic document categorization as described in Chapter 5 the number of features is increased from 317,522 features representing single words (before removing infrequent features) to 1,410,303 features (before removing infrequent features) representing word sequences containing 1 or 2 words.

The process of feature generation is performed in passes over documents, where *i-grams* are generated in the *i*-th pass only from the candidate features of length *i-1* generated in the previous pass. This process is similar to the large *k-itemset* generation used in association rules algorithm [1]. In Figure 3.1 we give Algorithm for the generation of new features. In the first pass all single words not contained in the "stop-list" and having sufficient frequency (here we check for term frequency > 3) are taken `LargeNGramSet`. Word sequences of size 2 up to `MaxNGramSize` (here we take `MaxNGramSize` $= 5$) are generated using several pruning criteria. In each pass, all documents are checked one by one using a window `NGramQueue` to get a sequence of words. Before the next word is added to the window (a window is moved one word to the right) the word is checked for being a proper word (not a number or a special symbol), for not being in the "stop-list" `StopWordSet` and for being included in the current set of word sequences `LargeNGramSet`. The window is reset when the next word is not a proper word or is not included in the set of word sequences generated so far.

Variables used in Algorithm in Figure 3.1:

Input: MinNGramOcc - minimal occurrences of N-Gram to become large N-Gram
MaxNGramSize - maximal size of N-Grams
StopWordSet - set of stop words (language dependent)
DocVec - vector of documents (DocVec[1], ..., DocVec[|DocVec|])
SymVec - vector of document lexical symbols (SymVec[1], ..., SymVec[|SymVec|])

Temporary: Sym - lexical symbol in document (possible value types: word, number, symbol)
CandNGramMap - mapping from candidate N-Gram to its occurrence counter
NGramQueue - queue (window) of last NGramSize words (excluding StopWordSet)

Output: LargeNGramSet - set of large N-Grams (occurring $\geq$ MinNGramOcc times)

Algorithm (C like pseudo-code):

```
LargeNGramSet = all single words in DocVec, not in StopWordSet and
occurring ≥ MinNGramOcc times;
for NGramSize = 2 to MaxNGramSize do {
 CandNGramMap=[];
 for SymVec = DocVec[1] to DocVec[|DocVec|] do {
  NGramQueue=[];
  for Sym = SymVec[1] to SymVec[|SymVec|] do {
   if (TypeOf(Sym)==word){
    if (Sym not in StopWordSet){
     if Sym in LargeNGramSet then {
      if (|NGramQueue|+1==NGramSize){
       if (Concatenated(NGramQueue) in LargeNGramSet){
        NGramQueue.Push(Sym);
        CandNGramMap[Concatenated(NGramQueue)]++;
        NGramQueue.Pop();
           } else {NGramQueue.Push(Sym); NGramQueue.Pop();}
      } else {NGramQueue.Push(Sym);}
     } else {NGramQueue=[];}
    } /* if (Sym not in StopWordSet) */
   } else {NGramQueue=[];}
  }; /* for Sym */
 }; /* for SymVec */
 LargeNGramSet += {NGram:CandNGramMap[NGram] ≥ MinNGramOcc};
};
return LargeNGramSet;
```

Figure 3.1: Algorithm for the generation of features representing word sequences (n-grams). Please notice that `NGramQueue` is a queue and not a stack. The generated features are used in enriched bag-of-words document representation.

For illustration, we show in Figure 3.2 the accumulated number of features during the described process of feature generation on Yahoo documents that compose the whole Yahoo hierarchy. The left hand-side graph shows for each pass of the *i-gram* generation the influence of the number of included Yahoo documents to the number of new features. For example, in the first pass (the lowest curve in the left hand-side of Figure 3.2) about 100,000 features are generated after including 20,000 documents, about 200,000 features after including 30,000 documents and about 320,000 features after including all 49,600 documents. After reducing the number of features by pruning infrequent features we get about 70,000 features representing single words (1-grams) to start with in the second pass over documents. In the second pass (the steepest curve in the left hand-side of Figure 3.2) all pairs of words (2-grams) that appear in the documents and consist of words included in the set of 1-grams are added. As can be seen from Figure 3.2, the second pass adds many infrequent features resulting with more than 1,400,000 features that after pruning infrequent features reduce to about 200,000 features. We stop with 5-grams since, in the fifth pass only about 10,000 features are added (the almost flat curve in the left hand-side of Figure 3.2) and about 6,000 of them are deleted as infrequent features. The right hand-side graph in Figure 3.2 shows the same process of feature generation over time. Influence of pruning infrequent features after each pass is even more evident here. It can be also clearly seen that the highest number of features is added in the second pass where 2-grams are generated, while in the fifth pass the curve showing the increasing number of features is almost flat (the right most part of the right hand-side graph in Figure 3.2).

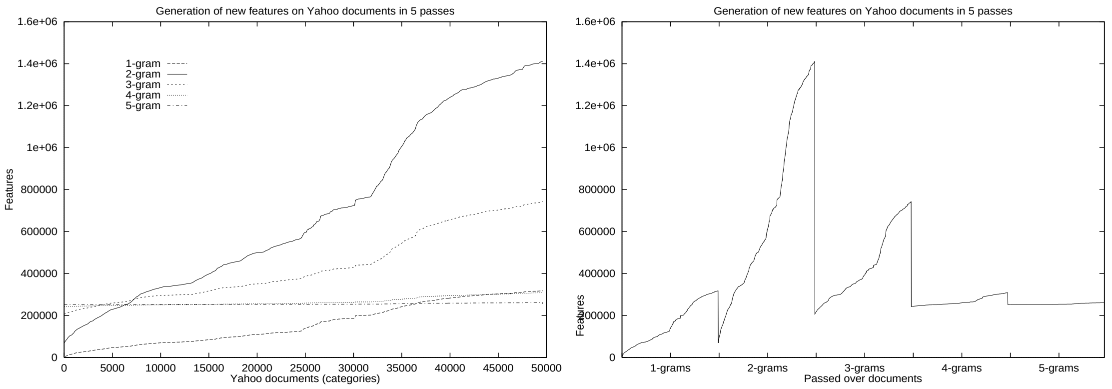

Figure 3.2: Influence of included Yahoo documents to the accumulated number of features (left hand-side graph) and the process of feature generation (right hand-side graph). Notice that the two graphs represent the same results in two different ways. At the end of each pass over documents all the features that occur less than 4 times are deleted.

Using document topic categories

We constructed a model for automatic document categorization from the existing Web hierarchy named Yahoo as described in Section 5.2. In our approach we take as input documents given in the hierarchy of topic categories and define a subproblem for each of the categories. The final result of learning is a set of specialized classifiers. Each classifier can distinguish between documents that should be classified in the category it represents, and other documents. When classifying a new document, each category from the hierarchy is assigned a probability as a result of consulting the corresponding subproblem classifier.

This model can be used as background knowledge that assigns to each document a list of categories describing the document content. Categories can be used as features in the feature-vector representation of documents. Possible approach is to represent categories with real-valued features each containing probability that a document belongs to the corresponding category. Then a naive Bayesian classifier for learning on numeric and symbolic features can be applied as described in [24]. The other possible approach is to map the probability assigned to the i-th category $P(Category_i)$ into a Boolean feature $F(Category_i)$ showing whether the document belongs to the corresponding category. This enables the usage of the naive Bayesian classifier on discrete-values features as described in Section 3.2.1. In this mapping $CtgTsh$ is the threshold for probability of category.

$$F(Category_i) = \begin{cases} true; & P(Category_i) > CtgTsh \\ false; & otherwise \end{cases} \tag{3.1}$$

In future work we plan to use document topic categories based on the Yahoo hierarchy as new features in document representation. We also plan to try using some of the existing electronic thesaurus to map words into more general categories to get higher level features. In this way, we would use the background knowledge that is not specialized for the Web documents as the Yahoo hierarchy is. The current system is based on statistic approach and does not include natural language specific information (eg. stemming, grammar analysis) except the usage of the English "stop-list". This enables mixing documents written in different natural languages but without making connection between their meaning. In further work we plan to support different natural languages including electronic dictionary and mapping words into one target language. In this way, knowledge model will be able to get information from documents that are not necessarily written in the same natural language as used in the induced model.

3.1.2 Learning on enriched document representation

We enrich document representation by adding word sequences and document topic categories. Instead of using word-vector to represent document, we use feature-vector generated by extending word-vector by new features. The process of forming document representation is illustrated in Figure 3.3. Feature construction using word sequences is performed on html-documents as described in Section 3.1.1. The resulting features represent not only single words (1-grams) but also word sequences (n-grams). Feature construction using document topic categories is performed on documents represented by word sequences as described in Section 3.1.1. Features representing word sequences are used to form the first part of the feature-vector, while the second part consists of features representing Yahoo categories generated using the constructed model for document categorization. Before using the machine learning algorithm we apply the feature selection by assigning a score to each feature and selecting only a small subset of highly scored features (see Chapter 4).

Experimental comparison of learning using bag-of-words representation (word sequences of length 1) and learning using the enriched document representation is given in Table 3.1. The experiments are performed on three domains defined from the Yahoo hierarchy, where the problem is document categorization and keyword assignment (see Chapter 5). Domain characteristics are given in Table 2.2. We compare influence of including different length word sequences in the document representation. We used Odds ratio for feature scoring and different vector sizes (vector size 1 meaning select as many highly scored features as there are features in positive examples), one of the best performing pruning settings (score threshold = 0.7, common features = 6) and probability threshold for Recall and Precision 0.95 (see Section 5.3 for explanation of the parameter values). It can be seen from Table 3.1 that with the growing word sequence length in most cases the performance in rank improves or remains stable. The highest improvement is achieved when going from single words (1-grams) to word pairs (2-grams). Closer look to the scores assigned by the feature scoring measure (Odds ratio) shows that some 2-grams are among highly scored features.

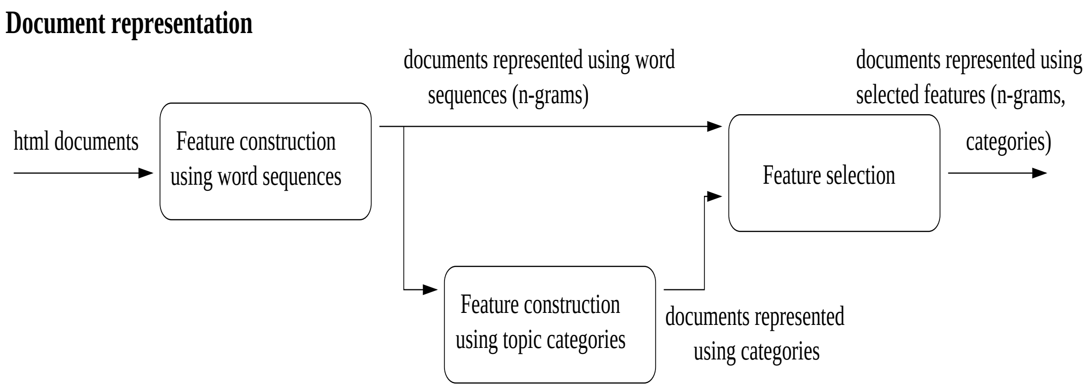

Figure 3.3: Generation of enriched document representation that includes features obtained by using word sequences and document topic categories. The example problem domain is prediction of clicked hyperlinks from the Web documents visited by the user.

3.2 Learning algorithms

As described in Section 1.2.3 different learning algorithms have been used on text data. In our experiments we apply a naive Bayesian classifier ([73]) and the k-Nearest Neighbor ([2, 88, 73]) on text data. In Chapter 6 we propose the modification of the k-Nearest Neighbor algorithm that we use in experiments involving domains with mutually dependent class attributes.

3.2.1 Naive Bayesian classifier

Experimental comparison of the naive Bayesian classifier to computationally more intensive alternatives on text data given by Pazzani and Billsus in [84] shows that it performs at least as well as these approaches achieving the highest Classification accuracy on most tested domains. The naive Bayesian classifier used in their comparison assumes that each document is represented as a Boolean word-vector. We refer to this document representation as a *Df-representation* and give formula (3.3) for estimating the probability of class value using the naive Bayesian classifier on this representation. In experiments by Pazzani and Billsus [84] for each word from a vocabulary a feature is defined and the feature independence assumption is used for different words. Feature subset is selected using Information gain (see Section 4.2.2) and for each selected feature information about its occurrence in the document is given in a Df-representation of the document (0 meaning the word does not occur, 1 meaning the word occurs).

Slightly different document representation is used in this thesis. Section 3.2.1 illustrates the differences between the two document representations used for classification by the naive Bayesian classifier. We use the naive Bayesian classifier on text as proposed in [40, 73] where the feature independence assumption is used for different words as well as for different occurrences of the same word. In this approach documents are represented with bag-of-words (unigram model of text is used) where a feature is defined for each word position in the document having word at that position as the feature value. Recall that in the bag-of-words representation only words, and not their positions, are considered. Thus a document can be seen as a set of words, where each word is assigned a frequency - the

Domain name	n-gram length: n	Median of the correct category Rank	Prob.	Average F2-measure on keywords	on categories	Vector size
Comp.	1	**14.2**± 1.17	**0.95**± 0.01	**0.59** ± 0.007	**0.31** ± 0.003	1
	1	13.6 ± 0.93	0.92± 0.02	0.59 ± 0.006	0.30 ± 0.004	1.5
	1	12.7 ± 0.92	0.90± 0.02	0.59 ± 0.006	0.27 ± 0.003	2
	2	**11.8**± 0.86	**0.98**± 0.006	**0.60** ± 0.006	**0.33** ± 0.001	1
	2	11.4 ± 0.98	0.97± 0.008	0.60 ± 0.006	0.32 ± 0.002	1.5
	2	10.8 ± 0.58	0.97± 0.008	0.60 ± 0.006	0.31 ± 0.002	2
	3	**11.7** ± 1.02	0.98± 0.007	0.60 ± 0.006	0.33 ± 0.002	1
	3	11.6 ± 0.93	0.97± 0.008	0.59 ± 0.009	0.32 ± 0.002	1.5
	3	11.2 ± 0.73	0.95± 0.001	0.58 ± 0.009	0.30 ± 0.002	2
	4	**11.6** ± 0.91	0.98± 0.007	0.60 ± 0.006	0.33 ± 0.002	1
	4	11.4 ± 0.98	0.97± 0.007	0.60 ± 0.006	0.32 ± 0.003	1.5
	4	11.6 ± 0.93	0.95± 0.013	0.59 ± 0.009	0.30 ± 0.003	2
	5-10	**11.7** ± 0.89	0.98± 0.007	0.60 ± 0.007	0.33 ± 0.002	1
	5-10	11.4 ± 1.04	0.97± 0.009	0.60 ± 0.007	0.32 ± 0.003	1.5
	5-10	11.4 ± 0.93	0.95± 0.014	0.59 ± 0.012	0.30 ± 0.003	2
Edu.	1	**11.7**± 1.09	**0.94**± 0.019	**0.53** ± 0.007	**0.39** ± 0.009	1
	2	**9.9** ± 1.55	**0.96**± 0.025	**0.55** ± 0.007	**0.43** ± 0.008	1
	3	10.7 ± 1.69	0.94± 0.040	0.55 ± 0.007	0.43 ± 0.008	1
	4	10.9 ± 1.55	0.94± 0.039	0.55 ± 0.007	0.44 ± 0.008	1
	5-10	10.8 ± 1.57	0.94± 0.039	0.55 ± 0.006	0.44 ± 0.008	1
Ref.	1	**4.6** ± 0.25	**0.98**± 0.006	**0.58** ± 0.009	**0.37** ± 0.008	1
	2	**3.6** ± 0.25	**0.99**± 0.002	**0.64** ± 0.006	**0.42** ± 0.010	1
	2	3.5 ± 0.22	0.96± 0.010	0.61 ± 0.009	0.32 ± 0.010	1.5
	3	3.6 ± 0.25	0.99± 0.002	0.64 ± 0.006	0.42 ± 0.009	1
	3	3.6 ± 0.25	0.95± 0.007	0.59 ± 0.010	0.32 ± 0.010	1.5
	4-10	3.6 ± 0.25	0.99± 0.002	0.64 ± 0.006	0.42 ± 0.008	1

Table 3.1: Results of experiments of using different value of n in n-grams (length of word sequences). We observe performance on two problems: document keyword assignment (F_2-measure) and document category prediction (correct category rank and probability, F_2-measure). Pruning parameters are set to (0.7, 6), probability threshold is set to 0.95. Reported are results for Odds ratio, as one of the best performing feature scoring measure and for the best performing vector sizes. The value of standard error is also given, since all the reported results are averages over five runs on independent testing sets.

number of times a word occurred in the document. We denote this document representation here as a *Tf-representation.* The assumption about feature independence used in the naive Bayesian classifier is here incorrect. According to [22] this does not necessarily mean that the classifier will have poor performance because of that.

The formula used to predict probability of class value C for a given document Doc given in [40, 73] is the following:

$$P(C|Doc) = \frac{P(C)\Pi_{F_j \in V} P(F_j|C)^{TF(F_j,Doc)}}{\sum_i P(C_i)\Pi_{F_l \in V} P(F_l|C_i)^{TF(F_l,Doc)}} \tag{3.2}$$

$$P(F_j|C) = \frac{1 + TF(F_j, C)}{|V| + \sum_i TF(F_i, C)}$$

Where product goes over all features from vocabulary V used in the representation of documents (all selected features), $P(C)$ is the probability of class value C, $P(W_j|C)$ is the conditional probability that word W_j represented by feature F_j occurs in a document given the class value C here estimated using Laplace probability estimate, $TF(F_i, C)$ is the frequency of word W_i in documents having class value C, $|V|$ is the vocabulary size and here equals to the number of all different features (words) used in the document representation, $TF(F_j, Doc)$ is the frequency of word W_j in document Doc. Laplace probability estimate is used here not because we would have to estimate probability from a small sample but because of the fact that many word sequences occur only in one class value. In other words it is not that $\sum_i TF(F_i, C)$ is small but $TF(F_j, C) = 0$ for many features. Since $TF(F_j, Doc) = 0$ for all the words that are not contained in document Doc, the predicted class probability can be estimated by taking into account only words (in our case selected features) contained in document Doc as follows:

$$P(C|Doc) = \frac{P(C)\Pi_{F_j \in Doc} P(F_j|C)^{TF(F_j,Doc)}}{\sum_i P(C_i)\Pi_{F_l \in Doc} P(F_l|C_i)^{TF(F_l,Doc)}}$$

Where product goes over all the features that occur in the representation of document Doc (have frequency > 0 in a feature-vector). We are using features to represent words and word sequences occurring in a document. Naive Bayesian classifier is simply used on all these features in similar way as used on features representing single words. Notice that this means even stronger violation of the feature independence assumption used in the naive Bayesian classifier.

Illustration of two document representations

We use two small artificial datasets to show the difference in the classification results of the naive Bayesian classifier using two alternative document representations. One document representation Tf-representation, is the same as used in [40, 73] and in this thesis (see Equation 3.2). When the naive Bayesian classifier is used on this document representation, word frequency is considered and only words that occur in a document contribute to the classification result.

The other document representation Df-representation, is the same as used in [84] (see Equation 3.3). Instead of taking into account word frequency, Df-representation uses binary features saying if word occurred in a document or not. The classification result of the naive Bayesian classifier used on this representation is influenced by all the words from the vocabulary. The probability of class value C using Df-representation of document is predicted using the following formula:

$$P(C|Doc) = \frac{P(C)\Pi_{F_j \in V} P(Doc(F_j)|C)}{\sum_i P(C_i)\Pi_{F_l \in V} P(Doc(F_l)|C_i)} \tag{3.3}$$

$$P(Doc(F_j)|C) = \frac{1 + N(Doc(F_j), C)}{2 + |D_C|}$$

Where $P(Doc(F_j)|C)$ is the conditional probability that the j-th feature has the same value as in Doc given the class value C here estimated using the Laplace probability estimate, $N(Doc(F_j), C)$ is the number of documents that have class value C and the same value of the j-th feature as document Doc, $|D_C|$ is the number of documents with class value C, 2 is the number of possible feature values (word occurs, word does not occur).

Illustration of the two described document representations is given on two artificial data sets, each having 10 training documents (3 positive and 7 negative) and 10 testing documents (5 positive and 5 negative). The two domains differ only in the training documents. Domain named *'Single'* contains training documents where each word occurs at most once in the same training document. In the domain named *'Multiple'* some words occur more than once in the same training document. We use Laplace probability estimate for $P(F_j|C)$ (needed when Tf-representation is used) and $P(Doc(F_i)|C)$ (needed when Df-representation is used). For illustration we show that the poor results are achieved when the relative frequency is used instead of Laplace estimate.

Training document for the two domains are the following:

Domain 'Single'	Domain 'Multiple'
+ feature subset selection	+ feature subset selection feature some feature
+ some feature values	+ some feature values is values values
+ while feature selection	+ while feature selection feature
- values of learning	- values of learning machine values
- learning is always	- learning is always always values learning learning
- for always is	- for always is machine for
- machine learning of	- machine learning of values machine
- always use values	- always use values for
- for some values	- for some values values
- most learning values	- most learning values values values values

Testing documents are the same for both domains, constructed to test different levels of confidence in class value. For instance, the first testing document "feature selection values" is characteristic for positive documents while the fourth testing document "use some feature" is rather marginal. The fifth testing document "most feature use feature for feature" is evidently positive if term frequency is taken into account (word "feature" that is characteristic for positive documents repeats three times). The following testing documents are used in experiments:

```
Testing documents
+ feature selection values
+ while some subset
+ selection values subset
+ use some feature
+ most feature use feature for feature
- selection of machine
- feature values for machine
- feature values use values for values
- learning while learning feature learning
- some machine is machine
```

Table 3.2 gives the classification results of experiments in the two domains using the Laplace probability estimate. For each testing example predicted probability for the class value 'positive' (+) is given. Experiments show that on both domains Tf-representation that considers multiple word occurrence when used by the Naive Bayesian classifier (see Equation 3.2) gives better results than Df-representation (see Equation 3.3). The main difference in results between the two document representations is for the testing documents that have multiple occurrences of the same word. More specifically, on a positive

Testing document	Domain *'Single'*, $P(+)$ using Tf-rep.	using Df-rep.	Domain *'Multiple'*, $P(+)$ using Tf-rep.	using Df-rep.
1.	0.88	0.87	0.93	0.84
2.	0.86	0.85	0.92	0.91
3.	0.78	0.75	0.80	0.69
4.	0.75	0.74	0.91	0.85
5.	0.97	**0.33**	0.99	**0.30**
6.	0.43	0.37	0.28	0.24
7.	0.38	0.27	0.24	0.08
8.	0.19	0.27	0.19	0.16
9.	0.18	**0.67**	0.18	**0.71**
10.	0.16	0.15	0.12	0.24

Table 3.2: Classification results using two different document representations on artificial domains *'Single'* (left) and 'Multiple' (right). Notice that for Tf-representation all testing documents are correctly classified, while for Df-representation two documents are misclassified (predicted probability is shown in boldface numbers).

document "most feature use feature for feature" and a negative document "learning while learning feature learning". Both contain multiple occurrences of the words characteristic for the correct class value. In the first document word "feature" and in the second word "learning". Other document words are either words characteristic for the incorrect class value ("for" in the first document, "feature" in the second) or some non-characteristic words ("most" and "use" in the first document, "while" in the second document).

If relative frequency is used instead of Laplace probability estimate, $P(W_j|C) = \frac{TF(W_j,C)}{\sum_i TF(W_i,C)}$ in Tf-representation and $P(W_j = Doc(W_j)|C) = \frac{DF(W_j=Doc(W_j),C)}{|D_C|}$ in Df-representation, for all testing examples at least for one class value is predicted probability 0. The consequence is poor classification result on both document representations. The first three and the last testing examples are correctly classified and probability 1 is predicted for the correct class value. For the rest of testing examples both class values are predicted probabilities 0. This is caused by multiplication with 0 that was expected, since the conditional probabilities are estimated using the relative frequency instead of the Laplace estimate.

3.2.2 k-Nearest Neighbor

The naive Bayesian classifier assumes that features are independent, which is obviously incorrect in the case of features representing words contained in a document. This assumption is not used by k-Nearest Neighbor algorithm that bases its class prediction for a given document on the subset of k most similar examples. In classification of documents into different topics [100] k-Nearest Neighbor was used on text data so that the classification returned by each neighbor is weighted by its similarity. It predicts probability of class value for a given document Doc based on the class probability distribution assigned to the k most similar documents.

$$P(C|Doc) = \frac{\frac{1}{k}\sum_{l=1}^{k} similarity(Doc, D_l)P(C|D_l)}{\sum_i \frac{1}{k}\sum_{l=1}^{k} similarity(Doc, D_l)P(C_i|D_l)} \tag{3.4}$$

Where D_l are neighboring documents. In general, one document can be assigned several class values (topics) with different probability $P(C_i|D_l)$. In our case this probability is 0 or 1, since each document has only one of the possible class values. Similarity between two documents is here measured as usually in information retrieval by cosine of angle between their representation vectors:

$$cos\theta(\vec{X},\vec{Y}) = \frac{\sum_i X_i Y_i}{\sqrt{\sum_j X_j^2 \sum_l Y_l^2}} \tag{3.5}$$

In our experiments k-Nearest Neighbor algorithm is used only in modified version. Namely, we modified k-Nearest Neighbor algorithm and used it in experiments involving domains with mutually dependent class attributes as described in Chapter 6.

Chapter 4

Feature subset selection

4.1 Approaches to feature subset selection

In propositional learning a problem domain is given by a set of examples, where each example is described with a class value and a vector of feature values. Features used to describe examples are not necessarily all relevant and beneficial for inductive learning. Additionally, a high number of features may slow down the induction process while giving similar results as obtained with a much smaller feature subset. Feature subset selection is commonly used when learning on text data, since most text domains are characterized by several tens of thousands of features.

The selection of a subset of features to be used in inductive learning has already been addressed in machine learning. The problem of selecting a subset of features can be seen as an optimization problem. The whole search space for optimization contains all possible subsets of features, meaning that its size is 2^F, where F is the number of features. Different methods have been developed and used for feature subset selection in statistics, pattern recognition and machine learning, using different search strategies and evaluation functions. The following search strategies are commonly used in feature subset selection:

forward selection — start with an empty set and greedily add features one at a time,

backward elimination — start with a feature set containing all features and greedily remove features one at a time,

forward stepwise selection — start with an empty set and greedily add or remove features one at a time,

backward stepwise elimination — start with a feature set containing all features and greedily add or remove features one at a time,

random mutation — start with a feature set containing randomly selected features, add or remove features randomly selected features one at a time and stop after a given number of iterations.

According to John et al. [42] there are two main approaches to feature subset selection used in machine learning: filtering approach and wrapper approach. In information retrieval and thus also in text-learning the whole process of feature subset selection is simplified with the assumption of feature independence. In this way the solution quality is traded for the time needed to a find solution, justified by the large number of features usually present in text domains.

4.1.1 Filtering approach

In the filtering approach illustrated in Figure 4.1, a feature subset is selected independently of the learning method that will use the selected features. There are methods in statistics that are used to reduce the number of features, like principal components analysis or factor analysis [70] that will not be addressed in this thesis. Here we briefly describe several feature subset selection algorithms used in machine learning that are based on the filtering approach.

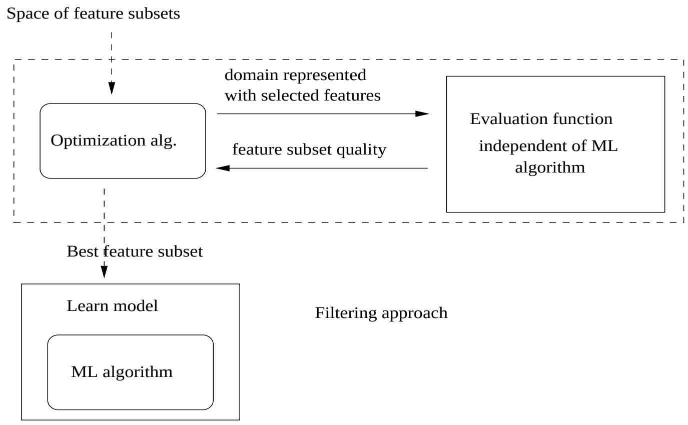

Figure 4.1: Illustration of the filtering approach to feature subset selection. Evaluation function does not include characteristics of the machine learning algorithm that will use the selected features.

Almuallim and Dietterich [5] developed several feature subset selection algorithms including a simple exhaustive search and algorithms that use different heuristics. They based their feature subset evaluation function on conflicts in class value occurring when two examples have the same values for all the selected features. In the first version of the feature subset selection algorithm called FOCUS, all the feature subsets of increasing size are evaluated until a *sufficient* subset is encountered. Feature subset Q is said to be sufficient if there are no *conflicts*. In other words, if there is no pair of examples that have different class values and the same values for all the features in Q. The successor of that algorithm FOCUS-2 prunes the search space, thus evaluating only promising subsets. Both algorithms assume the existence of a small set of features that forms a solution and their usage on domains with a large number of features can be computationally infeasible. This is the reason why search heuristics are used in the next versions of the algorithm resulting in good but not necessarily optimal solutions. The idea is to use forward selection until a sufficient subset is encountered. The proposed heuristic algorithms differ in the evaluation function they use. One of the heuristic algorithms uses conditional class entropy of training examples when positioned into $2^{|Q|}$ groups such that the examples in each group have the same truth assignment to the features in Q. $Entropy(Q) = -\sum_{k=0}^{2^{|Q|}-1} P(Group_k) \sum_i P(C_i|Group_k) \log_2 P(C_i|Group_k)$, where Q is the current feature subset we are evaluating, $P(Group_k)$ is the probability of the k-th group of examples containing training examples that all have the same truth assignment to the features in Q, $P(C_i|Group_k)$ is the conditional probability of the i-th class value given the k-th group of examples. The search stops when the previously described class entropy equals 0. The other two heuristic algorithms they developed are based on the number of conflicts caused by representing examples using only selected features. The search stops when there are no conflicts (the sufficient feature subset is encountered). Cardie [17] uses decision tree induction for feature subset selection. The idea is that only the features that appear in the induced decision tree are selected for learning using Nearest Neighbor algorithm. In this algorithm the optimization process is based on feature selection measure (in this case Information gain ratio) that is used in the greedy search to induce decision tree. Kira and Rendell [48] assigned weight to each feature and used nearest neighbor algorithm to update the weights. Training set is randomly sampled and the weights updated based on the difference between each selected example and its two nearest neighbors: the nearest hit - nearest neighbor having the same class value and the nearest miss - nearest neighbor having the opposite class value. Koller and Sahami [52] proposed an algorithm for feature subset selection that uses backward elimination to eliminate predefined number of features. The idea is to select subset of features that keeps class probability distribution as close as possible to the original distribution that is obtained using all the features. Evaluation function used to select the next feature to be deleted is based on Cross entropy described in Section 4.2.3. The algorithm finds for each feature a subset of K features such that the feature is approximated to be conditionally independent of the remaining features. Reducing K to 0 means having much faster algorithm that is actually the same as the feature selection algorithms commonly used on text-data (see Section 4.1.3). Liu and Setiono [68] used random sampling to search the space of all possible feature subsets. The predetermined number of subsets is evaluated and the smallest having the inconsistency rate below the given threshold is selected. They define inconsistency rate as the average difference between the number of examples with the same values of all the selected features and the number of examples among them having the same, most frequent class value among these examples. Noise handling is enabled by setting the threshold to some positive value. If the threshold is set to 0, the evaluation is based on consistency check as used in [5]. Pfahringer [85] proposed using the Minimum Description Length as an evaluation function of feature subset. In order to calculate the Minimum Description Length of a feature subset, training examples are represented with a *simple decision table* that contains only features from the subset. He used a search that adds or removes features one at a time. The search starts with the randomly selected feature subset found to be the best among a fixed number of randomly selected feature subsets. He reported that his method performs at least as well as the wrapper approach applied on the simple decision tables and scales up better to large number of training examples and features.

4.1.2 Wrapper approach

The idea of the wrapper approach is to select feature subset using the evaluation function based on the same learning algorithm that will be used for learning on domain represented with the selected features (see Figure 4.2 for illustration). Instead of using subset sufficiency, entropy or some other explicitly defined evaluation function, a kind of 'black box' function is used to guide the search. Namely, evaluation function for each candidate feature subset returns estimated quality of the model induced by the target learning algorithm using the feature subset. The idea is similar to automated model selection, where pruning parameters for decision tree induction are set by an optimization algorithm [75, 76]. This can result in a rather time consuming process, since for each candidate feature subset that is evaluated during the search, the target learning algorithm is usually applied several times (eg. 10 in case of 10-fold cross validation used for estimating model quality). Here we briefly describe several feature subset selection algorithms developed in machine learning that are based on the wrapper approach.

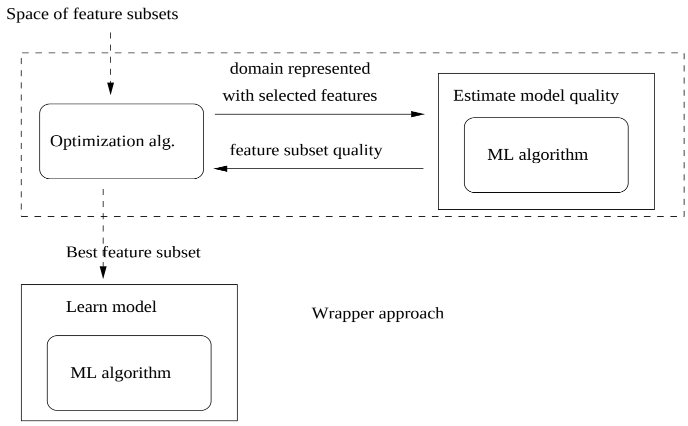

Figure 4.2: Illustration of the wrapper approach to feature subset selection. Evaluation function is using the machine learning algorithm that will use the selected feature subset.

Aha and Bankert [3] used a wrapper approach for feature subset selection in instance-based learning. They proposed a new search strategy that performs beam search using a kind of backward elimination. Namely, instead of starting with an empty feature subset, their search randomly selects a fixed number of feature subsets and starts with the best among them. Bala et al. [8] and Cherkauer and Shavlik [20] used a wrapper approach for feature subset selection for decision tree induction. Their method uses genetic algorithm to perform the search. Caruana and Freitag [18] developed a wrapper feature subset selection method for decision tree induction. They proposed a new search strategy that uses adding and removing features (backward stepwise elimination) and additionally at each step removes all the features that were not used in the decision tree induced for the evaluation of the current feature subset. John et al. [42] defined notion of strong and weak relevance of a feature and used a wrapper feature subset selection for decision tree induction. Skalak [96] used a wrapper approach for feature subset selection and for selecting a subset of examples to be stored in instance-based learning. Instead of using deterministic search strategy as used in [5, 18, 42] he used random mutation. Vafaie and De Jong [99] used a kind of wrapper approach where instead of using AQ15 for evaluation and learning, the evaluation is performed using the approximation of model quality estimated by AQ15 that selects feature subset which maximally separates classes using Euclidean distance. Their method uses genetic algorithm to perform the search.

4.1.3 Approach used on text data

Most of the feature subset selection approaches used in machine learning are not designed for situations with a large number of features. The usual way of learning on text defines a feature for each word that occurred in the training documents. This can easily result in several tens of thousands of features. Most methods for feature subset selection that are used on text are very simple compared to the described methods developed in machine learning. Basically, some evaluation function that is applied to a single feature is used. All the features are independently evaluated, a score is assigned to each of them and the features are sorted according to the assigned score. Then, a predefined number of the best features is taken to form the solution feature subset (see Figure 4.3). In our experiments described in Section 4.3 we observe the influence of the number of selected features to the performance of different feature scoring measures.

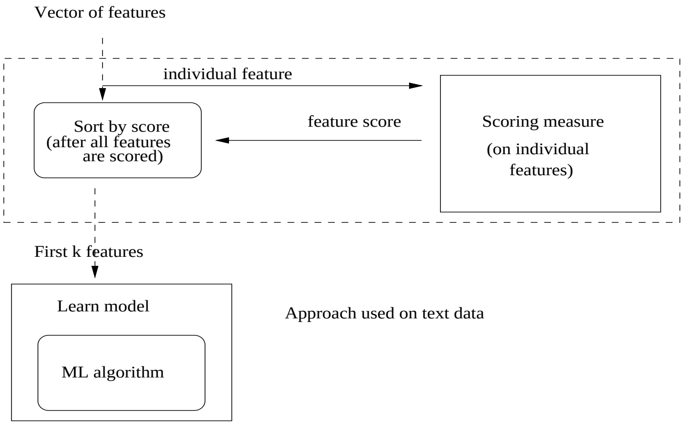

Figure 4.3: Illustration of the approach to feature subset selection commonly used on text data. Each feature is scored independently of other features. Features are sorted according to their score and the first k features are selected to form the solution feature subset.

One of the experimental question is how many best features to use in a particular domain. We have also addressed that question for our problem domain observing the influence of the used number of best features to the system performance. The other question is what scoring (evaluation function) to use since learning on text data has some specifics that should be taken into account when selecting feature subset.

In our first experiments on learning from text data [77] words were scored using Information gain (4.5) and learning was performed using the naive Bayesian classifier on frequency vectors (see Equation 3.2). It turned out that many features with high score were not very useful for our classification problem. Namely, Information gain assigns higher score to the features that make better distinction between class values. One of the practical problems we noticed in our experiments if small feature subset was used, is that a new document that should be classified, often contained very few or even non of the selected features and thus was classified to the majority class value (negative). This pointed out that in case of using majority class value as the default classification, efforts should be put in getting better description of the other class values. The problem here is to identify documents that have the positive class value, meaning that one class value (the target class value) is distinguished from the other class value. This additionally confirms that we should focus on the target, and here minority, class value. This is also confirmed by experiments reported in Section 4.3, where the best results are achieved using Odds ratio and its variants. These measures are characterized by favoring features characteristic for positive examples (see Section 4.2.6).

4.2 Feature scoring in text domains

Scoring of individual features in feature subset selection approach used on text data can be performed using some of the measures used in machine learning for feature selection during the learning process. For instance, Information gain used in decision tree induction [87] can be used to assign a score to features. In machine learning, several measures are known appropriate for different kinds of class and feature values. In our case we have binary-valued class (is the document relevant for the target concept?) and binary-valued features (does a feature occur in the document?). Notice that for the purpose of feature subset selection, frequency is replaced by occurrence or non-occurrence of a feature. Since many measures used in machine learning are based on Information theory [71] we first briefly explain some basic terms. The rest of this section gives description of different feature scoring measures that we experimentally compared including known measures that were already used for feature selection in text domains and some new measures we propose here.

4.2.1 Some basic terms from information theory

The amount of information can be defined as the number of bits needed to identify an event X_i that occurs with the probability $P(X_i)$.

$$I(X_i) = -\log_2 P(X_i)$$

The expected amount of information needed to identify an event is called entropy $H(X)$. Entropy can be calculated using the following formula:

$$H(X) = -\sum_i P(X_i) \log_2 P(X_i) \tag{4.1}$$

Where $P(X_i)$ is the probability of event X_i. By definition if $P(X_i) = 0$ then $P(X_i) \log_2 P(X_i) = 0$. The average amount of information about X contained in Y can be defined as the *expected cross entropy* or equivalently as difference between the entropy $H(X)$ and the conditional entropy $H(X|Y)$.

$$I(X,Y) = \sum_i P(X_i) I(X_i, Y) = H(X) - H(X|Y) \tag{4.2}$$

Where $I(X_i, Y)$ is *cross entropy* sometimes also referred to as *conditional mutual information* [71]. Mutual information between X_i and Y_j is defined as $\log_2 \frac{P(X_i,Y_j)}{P(X_i)P(Y_j)} = \log_2 \frac{P(Y_j|X_i)}{P(Y_j)} = \log_2 \frac{P(X_i|Y_j)}{P(X_i)}$, where $P(X_i, Y_j)$ is the joint probability of events X_i and Y_j and $P(Y_j|X_i)$ is the conditional probability of event Y_j given X_i. In case of independent events $P(X_i, Y_j) = P(X_i)P(Y_j)$ mutual information between them equals 0 since in that case $\log_2 \frac{P(X_i,Y_j)}{P(X_i)P(Y_j)} = \log_2 \frac{P(X_i)P(Y_j)}{P(X_i)P(Y_j)} = \log_2 1 = 0$. *Cross entropy* (conditional mutual information) is calculated using the following formula:

$$I(X_i, Y) = \sum_j P(Y_j|X_i) \log_2 \frac{P(Y_j|X_i)}{P(Y_j)} \tag{4.3}$$

From the definition of average amount of information that is given using entropy it can be seen that $I(X,Y) = I(Y,X)$ as follows:

$$I(X,Y) = H(X) - H(X|Y) = H(X) - [H(X,Y) - H(Y)] = H(Y) - H(Y|X) = I(Y,X)$$

$$H(X|Y) = -\sum_j P(Y_j) \sum_i P(X_i|Y_j) \log_2 P(X_i|Y_j)$$

For the sake of completeness we give the transformation showing that the expected cross entropy is equivalent to the difference between the entropy $H(X)$ and the conditional entropy $H(X|Y)$ as stated in Equation (4.2). The given transformation uses definition of conditional probability $P(X|Y) = \frac{P(X,Y)}{P(Y)}$.

$$\begin{aligned}
I(X,Y) &= \textstyle\sum_i P(X_i) I(X_i, Y) \\
&= \textstyle\sum_i P(X_i) \sum_j P(Y_j|X_i) \log_2 \frac{P(Y_j|X_i)}{P(Y_j)} \\
&= \textstyle\sum_i \sum_j P(X_i, Y_j) \log_2 \frac{P(X_i,Y_j)}{P(X_i)P(Y_j)} \\
&= \textstyle\sum_i \sum_j P(X_i, Y_j) \log_2 \frac{P(X_i,Y_j)}{P(Y_j)} + \sum_i \sum_j P(X_i, Y_j) \log_2 \frac{1}{P(X_i)} \\
&= \textstyle\sum_i \sum_j P(X_i|Y_j) P(Y_j) \log_2 P(X_i|Y_j) - \sum_i \left[\sum_j P(X_i, Y_j)\right] \log_2 P(X_i) \\
&= \textstyle\sum_j P(Y_j) \sum_i P(X_i|Y_j) \log_2 P(X_i|Y_j) - \sum_i P(X_i) \log_2 P(X_i) \\
&= -H(X|Y) + H(X) = H(X) - H(X|Y)
\end{aligned} \tag{4.4}$$

4.2.2 Information gain

Expected cross entropy is in machine learning commonly referred to as *Information gain* [86] (see Equation (4.4)). Its definition is usually given using the difference of class entropy in data set before and after the usage of the feature F for splitting the data set into subsets.

$$IGain(F) = I(C,F) = H(C) - H(C|F) = H(F) - H(F|C) = I(F,C) \tag{4.5}$$

4.2.3 Cross entropy for text

Cross entropy for text is used on binary-valued features, where only one feature value is important. An example domain is learning over text documents, where features indicate occurrence of words and only cases when the word occurred are important. Here we denote that value of feature F='true' as W, trying to make its meaning intuitive (word sequence W occurred in a document).

Expected cross entropy used in text-classification experiments [53] is similar to Information gain (4.5). The difference is that instead of calculating average over all possible feature values, only the value denoting that word (or in our case word sequence) W occurred in a document is considered. The summation part giving Cross entropy is the same (W denoting one of the feature values is used here instead of F that is used in Information gain formula). Expected cross entropy as used in text-classification is the same as *j-measure* [49] used in machine learning. This modified version of Cross entropy is given with the following formula:

$$CrossEntropyTxt(F) = P(W) \sum_i P(C_i|W) \log \frac{P(C_i|W)}{P(C_i)} \tag{4.6}$$

By definition of entropy given in Section 4.2.1, if $P(X_i) = 0$ then $P(X_i) \log_2 P(X_i) = 0$. In our case there are two possible problems: $P(C_i|W) = 0$ and $P(C_i) = 0$. The first case is handled by definition and the second is reduced to the first since if $P(C_i) = 0$ then also $P(C_i|W) = 0$.

Slightly modified Cross entropy for text was used as feature selection measure in text-classification [47] named 'Key word checker' $KCH(F)$. In this measure instead of $P(W)$ that is used in Cross entropy for text, $tanh(\frac{DF(W)}{\alpha})$ is used: $KCH(F) = tanh(\frac{DF(W)}{\alpha}) \sum_i P(C_i|W) \log \frac{P(C_i|W)}{P(C_i)}$. Where $DF(W)$ is the number of documents containing word W, α is a user given parameter. All other variables are the same as in $CrossEntropyTxt$ (4.6).

4.2.4 Mutual information for text

$Mutual\ information$ used in text-classification experiments [64, 101] is based on mutual information between two variable values known from information theory. $Mutual\ information\ for\ text$ is calculated as mutual information for a fixed feature value (word W occurred) averaged over all class values.

$$MutualInfoTxt(F) = \sum_i P(C_i) \log \frac{P(W|C_i)}{P(W)} \tag{4.7}$$

4.2.5 Weight of evidence for text

One of the new feature scoring measures we propose is $Weight\ of\ evidence\ for\ text$ that is based on the average absolute weight of evidence. Weight of evidence captures the difference between the probability of class C_i occurring given the j-th feature value and the probability of class C_i. Average weight of evidence for feature F is calculated over all class values [49] using the following formula: $WeightOfEvid(F) = \sum_i P(C_i) \times \sum_j P(F_j) |\log \frac{odds(C_i|F_j)}{odds(C_i)}|$, where $odds(C_i|W)$ and $odds(C_i)$ are calculated using formula (4.10).

We propose that when used on text, instead of calculating average over all possible values of feature F, only the value denoting that word W occurred in a document is considered.

$$WeightOfEvidTxt(F) = \sum_i P(C_i) \times P(W) \times |\log \frac{odds(C_i|W)}{odds(C_i)}| \tag{4.8}$$

4.2.6 Odds ratio

Odds ratio is used for binary-valued class problems on binary-valued features, where the goal is to make a good prediction for one of the class values [89]. This measure is commonly used in information retrieval, where the problem is to rank out documents according to their relevance for the positive class value using occurrence of different words as features. Ranking of document D with respect to its relevance for a given query that is used in information retrieval can be defined as follows [89]: $Ranking(D, C_1) = \log \frac{P(C_1|D)}{P(C_2|D)} = \log \frac{P(C_1)P(D|C_1)}{P(C_2)P(D|C_2)}$. Where the problem has binary-valued class with the values: C_1 meaning document is relevant and C_2 meaning document is irrelevant. Each document can be represented as a vector of binary-valued features recording if corresponding word occurred in a document. Using that and the incorrect but common assumption of independence between words, the following formula is obtained: $Ranking(D, C_1) = \log \frac{P(C_1)\Pi_j P(W_j|C_1)}{P(C_2)\Pi_j P(W_j|C_2)} = \sum_j weight(F_j)Z_j + Const$. Where D is document, C_1 is the positive (target) class value, $P(W_j|C_i) = P(F_j = true|C_i)^{Z_j}(1 - P(F_j = true|C_i))^{1-Z_j}$ is the conditional probability of the j-th feature value given the i-th class value, $weight(F_j)$ is the weight of word associated with feature F, $Z_j = 1$ if $F_j = true$ and 0 otherwise. Word weight used in the above formula is called Odds ratio.

$$OddsRatio(F) = \log \frac{odds(W|C_1)}{odds(W|C_2)} \tag{4.9}$$

$$odds(X_i) = \begin{cases} \frac{\frac{1}{n^2}}{1-\frac{1}{n^2}}; & P(X_i) = 0 \\ \frac{1-\frac{1}{n^2}}{\frac{1}{n^2}}; & P(X_i) = 1 \\ \frac{P(X_i)}{1-P(X_i)}; & P(X_i) \neq 0 \wedge P(X_i) \neq 1 \end{cases} \tag{4.10}$$

We handle singularities as usually handled in Information retrieval and proposed in [95]: $P(X_i) = 0$ is replaced with $P(X_i) = \frac{1}{n^2}$ and $P(X_i) = 1$ is replaced with $P(X_i) = 1 - \frac{1}{n^2}$, where n is the number of examples. Notice that Odds ratio is not averaged over all feature values but only the value recording that word occurred is considered.

4.2.7 Variants of Odds ratio

We propose several variants of Odds ratio which was found as the most promising feature scoring measure in our preliminary experiments. As can be seen from Section 4.2.6, Odds ratio does not take into account frequency or probability of word occurrence. Our experiments show that Cross entropy for text gives better results than Mutual information for text while the main difference between them is that Cross entropy for text takes into account the probability of word occurrence $P(W)$ (see Equations (4.6) and (4.7)). Thus we propose a variant of Odds ratio that is weighted by the probability of word occurrence. The idea was to test if taking into account word probability would improve results of Odds ratio as in the case of Mutual information. Our experiments show that the results of Odds ratio and it's variant that includes probability of word occurrence are not significantly different (see Section 4.3).

$$WeightedOddsRatio(F) = P(W) \times OddsRatio(W) \tag{4.11}$$

We tried several measures inspired by Odds ratio to experimentally test what are the important properties of Odds ratio. We tested two simple measures that we call $Log - probability\ ratio$ and $Exp - probability\ difference$. The idea was to see if these simplified formulas still have similar behavior as Odds ratio (confirmed by experiments in Section 4.3). Both measures aim at capturing difference in the conditional probability of word occurrence in different class values. This makes them appropriate for problems where one of the class values is the target value and we want to identify examples having the target class value (eg. identifying relevant documents).

$$LogProbRatio(F) = \log \frac{P(W|C_1)}{P(W|C_2)} \tag{4.12}$$

$$ExpProbDiff(F) = e^{P(W|C_1) - P(W|C_2)} \tag{4.13}$$

Odds ratio assigns the lowest score to the words that are frequent in irrelevant documents (negative examples). In some cases we might want to use such 'very negative' words to help recognizing that some document containing them is not relevant despite the other words in it. In order to accommodate that, we propose *Conditional odds ratio* that is divided in two parts. The first part is Odds ratio with the changed roles of the class value and covers the cases when a word is highly probable in documents having negative class value ($P(W|C_2) - P(W|C_1) > InvTsh$), while the second part is regular Odds ratio. The threshold used to switch between these two parts is set according to the importance of the class values.

$$CondOddsRatio(F) = \begin{cases} InvWgt \times \log \frac{odds(W|C_2)}{odds(W|C_1)}; & P(W|C_2) - P(W|C_1) > InvTsh \\ \log \frac{odds(W|C_1)}{odds(W|C_2)}; & otherwise \end{cases} \tag{4.14}$$

Conditional odds ratio parameters

In our experiments, after trying several threshold values we used $InvTsh = 0.01$ as appropriate for our domains. In Conditional odds ratio, the threshold is used for checking the difference between the conditional probability of feature given the non-target class value $P(W|C_2)$and given the target $P(W|C_1)$ class value. Thus, we decided to test domain dependent threshold values, based on the feature probability calculated as if all the features were equally probable $InvTsh = \frac{1}{\#features}$. In most cases, the combinations that get over the threshold are $P(W|C_1) = 0$, $P(W|C_2) > InvTsh$. We experimentally compared different thresholds set to the feature probability multiplied by different factors (1,10,100,...). In order to reduce the influence of the features characteristic for the negative class examples, we multiply their score by the inverse weight $InvWgt$ that we experimentally assign value 0.01, after testing $0.001, 0.01, 0.1$.

The problem we used for testing the parameter setting is document categorization described in Section 5.1. The used three domains *'Computer and Internet'*, *'Education'* and *'Reference'* are generated from the Yahoo hierarchy (see Table 2.2 for domain characteristics). Experimental setting is the same as described in Section 5.3, with category pruning set to (0.7, 3), probability threshold set to 0.95 and vector size set to 1. Summary of experimental results testing different parameter settings (threshold and weight) is given in Table 4.1. It can be seen that on all three domains there is no significant differences for weight ($InvWgt$ set to 0.001-0.1). The best results on *'Computers and Internet'* are achieved for $InvTsh$ set to 0.0131 or 0.131 (100 to 1000 times $\frac{1}{7631}$), on *'Education'* for $InvTsh$ set to 0.0313 or 0.313 (100 to 1000 times $\frac{1}{3198}$) and on *'References'* for $InvTsh$ set to 0.0108 or 0.108 (10 to 100 times $\frac{1}{928}$). From that we got our parameter setting by selecting the setting that works well for all tested domains.

Domain name	Median of the correct category		Average F2-measure		Average considered categories	Parameters	
	Rank	Prob.	on keywords	on categories		InvTsh	InvWgt
Comp.	10.8 ± 0.86	0.94 ± 0.02	0.60 ± 0.007	0.28 ± 0.003	186 ± 3.2	0.000131	0.001
	10.4 ± 0.75	0.91 ± 0.02	0.61 ± 0.015	0.26 ± 0.003	177 ± 2.9		0.01
	15.0 ± 1.05	< 0.001	0.51 ± 0.010	0.11 ± 0.003	78 ± 1.7		0.1
	11.0 ± 0.77	0.93 ± 0.01	0.60 ± 0.007	0.29 ± 0.003	186 ± 3.0	0.00131	0.001
	10.5 ± 0.77	0.91 ± 0.02	0.61 ± 0.020	0.28 ± 0.003	177 ± 3.1		0.01
	17.5 ± 1.40	< 0.001	0.55 ± 0.009	0.21 ± 0.004	82 ± 1.6		0.1
	12.5 ± 1.1	**0.98** ± 0.006	0.55 ± 0.007	0.34 ± 0.002	246 ± 3.4	**0.0131**	0.001
	12.9 ± 0.95	**0.98** ± 0.007	0.55 ± 0.007	0.34 ± 0.002	246 ± 3.4		0.01
	14.6 ± 1.10	**0.97** ± 0.007	0.55 ± 0.007	0.34 ± 0.002	230 ± 3.0		0.1
	12.0 ± 0.84	**0.99** ± 0.003	0.55 ± 0.007	0.34 ± 0.002	259 ± 3.6	**0.131**	0.001
	11.9 ± 0.83	**0.99** ± 0.003	0.55 ± 0.007	0.34 ± 0.002	258 ± 3.6		0.01
	12.2 ± 1.90	**0.99** ± 0.003	0.55 ± 0.007	0.34 ± 0.002	258 ± 3.6		0.1
Edu.	17.4 ± 1.20	0.10 ± 0.08	0.55 ± 0.008	0.36 ± 0.006	41 ± 1.1	0.000313	0.001
	17.2 ± 1.20	0.07 ± 0.06	0.55 ± 0.008	0.35 ± 0.010	40 ± 1.1		0.01
	20.7 ± 0.97	< 0.001	0.50 ± 0.004	0.15 ± 0.005	29 ± 0.8		0.1
	15.9 ± 0.95	0.43 ± 0.14	0.55 ± 0.007	0.48 ± 0.006	45 ± 1.0	0.00313	0.001
	15.5 ± 0.95	0.37 ± 0.11	0.56 ± 0.007	0.48 ± 0.006	43 ± 1.1		0.01
	18.2 ± 1.70	0.08 ± 0.04	0.58 ± 0.006	0.49 ± 0.006	37 ± 1.0		0.1
	10.0 ± 0.98	**0.99** ± 0.004	0.50 ± 0.007	0.44 ± 0.005	61 ± 1.3	**0.0313**	0.001
	9.7 ± 0.84	**0.99** ± 0.003	0.50 ± 0.007	0.44 ± 0.005	60 ± 1.3		0.01
	10.3 ± 1.30	**0.99** ± 0.004	0.50 ± 0.007	0.45 ± 0.006	60 ± 1.3		0.1
	9.9 ± 1.10	**0.99** ± 0.002	0.48 ± 0.007	0.43 ± 0.005	63 ± 1.4	**0.313**	0.001
	9.7 ± 1.10	**0.99** ± 0.002	0.48 ± 0.007	0.43 ± 0.005	63 ± 1.4		0.01
	10.0 ± 1.10	**0.99** ± 0.002	0.48 ± 0.007	0.43 ± 0.005	63 ± 1.4		0.1
Ref.	3.6 ± 0.25	0.82 ± 0.07	0.58 ± 0.014	0.28 ± 0.010	18 ± 0.5	0.00108	0.001
	3.6 ± 0.25	0.76 ± 0.09	0.57 ± 0.010	0.27 ± 0.008	18 ± 0.5		0.01
	18.9 ±11.7	0.15 ± 0.09	0.50 ± 0.018	0.20 ± 0.004	16 ± 0.4		0.1
	3.2 ± 0.20	**0.99** ± 0.001	**0.64** ± 0.005	**0.42** ± 0.010	22 ± 0.5	**0.0108**	0.001
	3.0 ± 0	**0.99** ± 0.001	**0.64** ± 0.006	**0.42** ± 0.010	22 ± 0.5		0.01
	3.0 ± 0	**0.99** ± 0.001	**0.65** ± 0.005	**0.43** ± 0.010	21 ± 0.4		0.1
	4.4 ± 0.25	0.99 ± 0.001	0.57 ± 0.006	0.37 ± 0.009	26 ± 0.5	0.108	0.001
	4.4 ± 0.25	0.99 ± 0.001	0.57 ± 0.007	0.37 ± 0.009	26 ± 0.5		0.01
	4.4 ± 0.25	0.99 ± 0.003	0.57 ± 0.006	0.37 ± 0.010	26 ± 0.5		0.1

Table 4.1: Experimental comparison of different parameter settings in Conditional odds ratio scoring measure. We observe performance on two problems: document keyword assignment and document category prediction. We give median of the correct category rank and probability, mean value of F_2-measure on both problems. Parameters are set to one of the best performing combinations: pruning (0.7, 3), probability threshold 0.95 and vector size to 1. We report the mean value and standard error, since all the reported results are averages over five runs on independent testing sets.

4.3 Feature subset selection experiments

In our comparison we include eleven feature scoring measures. Information gain (4.5) was included as the well known measure used in machine learning and also some text-learning experiments. Three of the included measures were reported to work well on text data: Cross entropy for text (4.6), Mutual information for text (4.7) and a very simple measure proposed in the text-classification domains [101]. This third measure is calculated either as the number of documents that contain word W referred to as document frequency $DF(W)$ or as the number of occurrences of word W referred to as term frequency $TF(W)$. This measure requires the stop words removal. We use the second definition: $Freq(F) = TF(W)$. Inspired by the way Cross entropy for text is derived from Information gain, we propose Weight of evidence for text (4.8). Preliminary experiments show that Odds ratio is the most promising measure, thus we included four variants of Odds ratio as described in Section 4.2.7. Random scoring measure included as a baseline measure was defined to score each word by a random number between 0 and 1.

The first set of experiments was performed on different domains defined from Yahoo hierarchy as described in Chapter 5. Results of the naive Bayesian classifier are given on domains representing five of the top fourteen Yahoo categories, whose characteristics are given in Table 2.2. *'Entertainment'* having 8,081 class values (nodes in the hierarchy), 30,998 features (selected word sequences) and 79,011 examples (actual Web documents). *'Arts and Humanities'* having 3,085 class values, 11,473 features and 27,765 examples. *'Computers and Internet'* having 2,652 class values, 7,631 features and 23,105 examples. *'Education'* having 349 class values, 3,215 features and 5,406 examples. *'References'* having 129 class values, 923 features and 1,995 examples.

4.3.1 Experimental setting and evaluation

In our experiments we observe the influence of the number of selected features to the system performance. The number of selected features is measured here relatively to the category size expressed by the number of features in positive examples. We refer to this relative number of features as *Vector size*. Recall that we get our categorization results from the set of independent classifiers, each potentially having different number of features. Thus we express the vector size as a factor of the number of features in positive examples, since the set of negative examples is the same for all the classifiers. In this way, a classifier for a larger category is using more features than a classifier for some smaller category, while both classifying the same testing example.

Reported are classification results on independent set of (500 for the bigger and 300 for the two smaller domains) testing examples selected randomly from the actual Web documents accessible from the corresponding hierarchy domain. The reported results are averaged over 5 repetitions using hold-out method, ie., we keep 500 (300) items from Yahoo Web documents out of the training data. The actual Web documents these items point to are used as the testing documents.

We report average Precision and Recall (see Section 2.3.2) for category prediction and for keyword assignment. The data provided in Yahoo hierarchy 5.1 enables mapping from predicted categories to keywords. Precision and Recall for each testing example is calculated based on the set of correct categories (keywords) and the set of predicted categories (keywords). As the predicted categories we take all the categories whose classifier predicts probability over 0.95 that the document is a member of the corresponding category. The predicted set of keywords is obtained as a union of the keywords assigned to all the predicted categories. In other words, Precision and Recall are calculated for the fixed probability threshold that is here set experimentally to 0.95 (see Section 5.3.2 and paper [34]). Notice that one testing example can be originally assigned to several categories, since it can be referenced from several parts of the hierarchy. In such case, all these categories are taken as correct and also used to form the set of correct keywords and compared to the set of predicted categories (keywords). Predicting any of the correct categories is sufficient for the correct classification. Additional to Precision and Recall we report F_2-measure that is commonly used (see Section 2.3.2) when we care more about Recall than about Precision. Namely, for the user that wants to classify a new document, it is easier to ignore few categories among the predicted categories than to check the rest of several hundreds or thousands categories to find the missing ones.

Additionally, we report rank and probability assigned to the correct category. For each testing example we observe a list of categories each assigned a probability as a result of consulting the corresponding subproblem classifier. Sorting categories according to that probability gives ranking that we use to get the rank of the correct category. For instance, in the classification results given in Table 5.4 for the correct category is predicted probability 0.999932 and assigned rank 1. In case of several categories having the same probability, the middle rank is taken, ie. 2 in the case of 3 top categories having the same probability and 1.5 in the case of 2 top categories having the same probability. If there are more correct categories, the one with the highest predicted probability is considered. To get summary results over testing examples, we give median with the lower and upper quartile. The reason for reporting median instead of the mean value is that some of the testing examples are rather non-typical of their category, containing eg., a welcome page or only one sentence asking for language preference or an error message or a page giving redirection. These non-typical examples can be detected by the system from the low probabilities assigned to all categories. In our opinion for the practical usage of our approach for document categorization such non-typical examples should be rejected by the system meaning that human categorization is required. This can also be used to indicate the need for extending the current hierarchy. Here we include them in testing in order to get a fair estimate of the system performance.

In order to enable operational usage of the system on larger domains we include the pruning mechanisms described in Section 5.3.1. When classifying a testing example only the classifiers that share at least 6 highly scored features with the example are considered. We tested different pruning levels (1,...12) on several domains [80] and found that requiring at least 6 features is one of the best performing pruning levels. As highly scored features we take so many best features to get their score summary make 70% of the summary score assigned to all the selected features. In this way we handle situations when there is a small number of features standing out by their score and the situations when the most features have similar score.

4.3.2 Experimental results

Figure 4.4 gives for different scoring measures rank and probability assigned to the correct category on domain *'Reference'*. It can be seen that the best performance in probability is achieved by Odds ratio for vector size 1-1.5 and by Cross entropy for text for vector size 0.5-1. The best performance on *'Reference'* in rank is achieved by Cross entropy for text and Term frequency for vector size 0.5. Slightly worse but still very good performance is achieved by Odds ratio for vector size 1-2. Figure 4.5 shows influence of vector size to the number of categories considered in classification when different scoring measures were used. It can be seen that for Random scoring this number grows with the growing vector size. Similar is true for Information gain, but the initial values are much higher and the growth is slower. For the other measures, the number of considered categories remains the same for vector sizes > 1.5. The exception is Weight of evidence for text that has the smallest number of considered categories growing from 10 for vector size 3 to 18 for vector size 15. Odds ratio is the second best in the low number of considered categories, significantly better than the other measures that achieve comparable results in rank and probability (Cross entropy for text and Tern frequency). Cross entropy for text is the worst in the number of considered categories (65 for vector size 0.5 and almost 80 for the larger vector sizes).

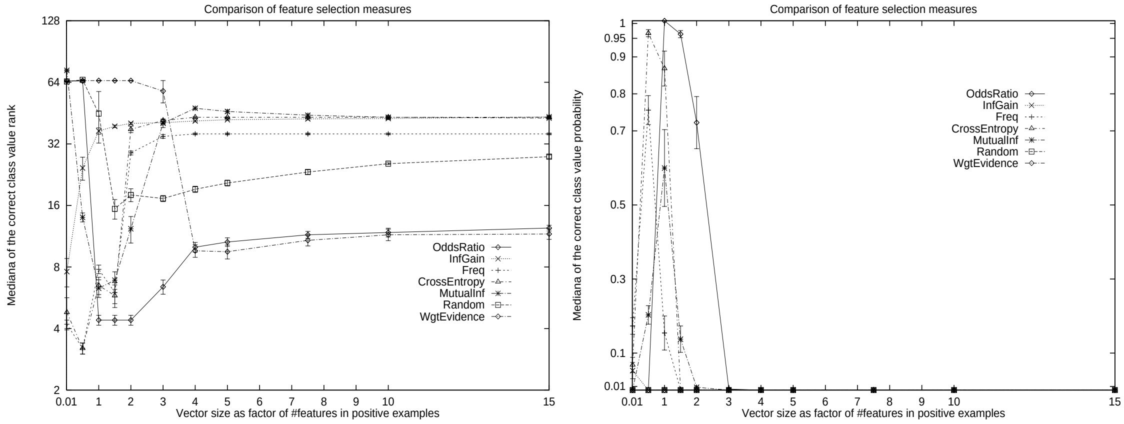

Figure 4.4: Comparison of the correct category rank and probability for different feature scoring measures on domain *'Reference'* defined from the Yahoo hierarchy. The problem is document categorization based on the given hierarchy of categorized documents. Given results are for pruning setting = (0.7, 3) and probability threshold 0.95. We give mean and standard error of median for the correct category rank and probability on domain *'Reference'*. Given are average values over 5 runs with standard error.

In all domains the best performance is achieved when only a small number of features is used (vector size is low) and the features are selected using Odds ratio as a scoring measure (see Tables 4.5, 4.6, 4.7). This is consistent with the results reported in text-learning on the problem of predicting clicked hyperlinks from the set of visited Web documents [78] where the feature selection based on Odds ratio achieved the best results. Similar observation regarding the number of features is reported on text categorization in [101], where the reduction of up to 90% in the number of features resulted in either an improvement or no loss in the system performance.

The best performance in rank and probability over all five domains is achieved here by Odds ratio on the feature subset having relatively small number of features (vector size 1). To get an idea about the actual number of used features, vector size 1 means in average over categories: on *'Entertainment'* 58 out of 30,998 features (0.2%), on *'Arts and Humanities'* 65 out of 11,473 features (0.7%), on *'Computers and Internet'* 42 out of 7,631 features (0.62%), on *'Education'* 85 out of 3,198 features (2.7%), on *'References'* 49 out of 928 features (5.3%). More specifically (see Table 4.5), on *'Entertainment'* the average median of the correct category rank is 16 and the average median of the correct category probability over 0.57, ie. the half of the testing examples are assigned rank up to 16 for the correct category and probability > 0.57. On *'Arts and Humanities'* the average median of the correct category rank is 9.9 and the average median of the correct category probability over 0.99, ie. the half of the testing examples are assigned rank up to 10 for the correct category and probability > 0.99. On *'Computers and Internet'* the average median of the correct category rank is 11.9 and the average median of the correct category probability over 0.98, ie. the half of the testing examples are assigned rank up to 12 for the correct category and probability > 0.98. On *'Education'* the average median of the

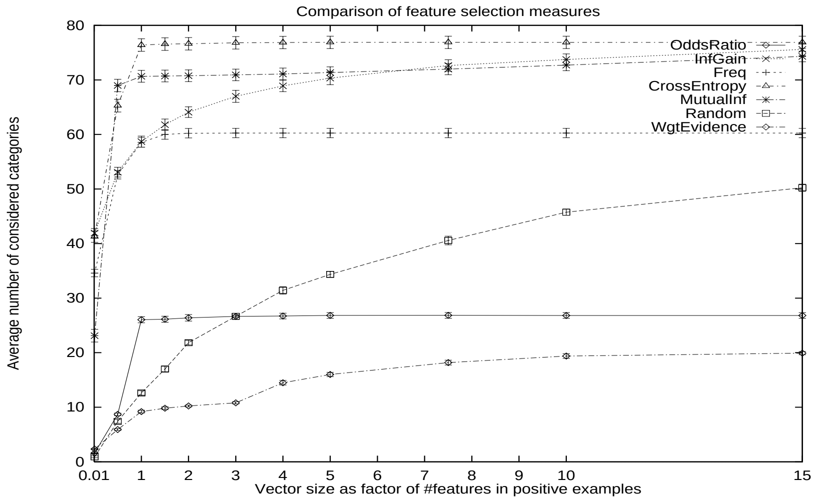

Figure 4.5: Comparison of the number of considered categories for different feature scoring measures on domain *'Reference'* defined from the Yahoo hierarchy. The problem is document categorization based on the given hierarchy of categorized documents. Given results are for pruning setting = (0.7, 3) and probability threshold 0.95. We give mean and standard error of the number of considered categories.

correct category rank is 9.4 and the average median of the correct category probability over 0.98, ie. the half of the testing examples are assigned rank up to 9.4 for the correct category and probability > 0.98. On *'References'* the mediana of the correct category rank is from 3.8 the mediana of the correct category probability over 0.99, ie. the half of the testing examples are assigned rank up to 3.8 and probability > 0.99 for the correct category. There is no significant difference in performance between Odds ratio and any of the four variants of Odds ratio we tested (see Table 4.2). The lowest number of categories was considered by Log-probability ratio for text and Conditional Odds ratio on two out of the three domains. For illustration we show in Figure 4.6 the average value and standard error on *'Reference'* for Odds ratio and its variants. This shows that we manage to keep the most important characteristics of Odds ratio when forming the new measures. It also shows that we didn't get any significant improvement by including probability of word occurrence (Weighted Odds ratio) nor by including features characteristic for negative examples (Conditional Odds ratio).

Cross entropy for text and Term frequency achieved similar results as Odds ratio on two out of five domains. The new measure we named Weight of evidence for text achieved on all domains good rank but low probability (< 0.01). Mutual information for text performed well on *'References'*, while Information gain performed similar as Random on the two bigger domains. Observation of standard error of rank and probability on all five domains confirms that the best performing Odds ratio and Cross entropy for text are significantly better than the other tested measures. Observation of the average number of considered categories during the classification shows that Odds ratio is considering more than 3 times less categories than Cross entropy for text.

The performance on predicting category measured by F_2-measure (a combination of Precision and Recall) is worse on all five domains than the performance on predicting keywords (compare Tables 4.6, 4.7). This was expected, since when evaluating cate-

Domain name	Scoring measure	Median of the correct category Rank	Probability	Average considered categories	Vector size
Comp.	Log. prob. ratio for text	11.7 ± 0.89	0.96 ± 0.008	**114** ±2.0	1
	Exp. prob. diff for text	12.3 ± 0.88	0.99 ± 0.004	385 ±8.0	1
	Weighted Odds ratio	11.3 ± 0.77	0.98 ± 0.007	196 ±4 0	1
	Conditional Odds ratio	11.8 ± 0.97	0.97 ± 0.007	**116** ±2.0	1
	Odds ratio	11.9 ± 0.96	0.98 ± 0.007	138 ±2.1	1
Edu.	Log. prob. ratio for text	10.5 ± 1.70	0.94 ± 0.04	**34** ±1.0	1
	Exp. prob. diff for text	9.2 ± 0.97	0.99 ± 0.001	83 ±1.6	1
	Weighted Odds ratio	9.8 ± 1.00	0.99 ± 0.002	60 ±1.5	1
	Conditional Odds ratio	10.3 ± 1.90	0.94 ± 0.04	**35** ±1.1	1
	Odds ratio	9.4 ± 0.98	0.99 ± 0.002	65 ±1.4	1
Ref.	Log. prob. ratio for tex	4.4 ± 0.25	0.99 ± 0.001	26 ±0.5	1
	Exp. prob. diff for text	3.8 ± 0.20	0.99 ± 0.001	23 ±0.5	0.5
	Weighted Odds ratio	3.4 ± 0.30	0.99 ± 0.001	**18** ±0.4	0.5
	Conditional Odds ratio	4.4 ± 0.25	0.99 ± 0.001	30 ±0.5	1
	Odds ratio	4.4 ± 0.25	0.99 ± 0.002	26 ±0.5	1

Table 4.2: Comparison of variant of Odds ratio used for feature selection measures on domains formed from Yahoo hierarchy. The problem is document categorization based on the given hierarchy of categorized documents. We give the correct category rank and probability for vector size 1, as one of the best performing vector sizes. Given results are for pruning setting = (0.7, 3), probability threshold 0.95 and vector size 1. Given are average values and standard error.

gory prediction we count as wrong all the predicted categories that don't exactly match any of the correct categories. In this way we ignore the fact that some of these categories might be just a bit more general than the correct category. For instance, predicting category `'Reference:Libraries:Library and Information Science'` instead of `'Reference:Libraries:Library and Information Science:Information Retrieval'` is counted as wrong. When assigning a set of keywords, the set of keywords assigned by prediction of a more general category is a subset of the set of correct keywords. The previous prediction will in this case be evaluated as partially correct, since some of the predicted keywords (words in the category name) are correct. On both problems and on all domains Odds ratio is again among the best performing scoring measures. More precisely (see Tables 4.6, 4.7), on *'Entertainment'* Odds ratio achieves F_2 of 0.59, Precision of 0.44 and Recall of 0.80 on keyword assignment and F_2 of 0.30, Precision of 0.41 and Recall of 0.34 on predicting document category. On *'Arts and Humanities'* Odds ratio achieves F_2 of 0.59, Precision of 0.40 and Recall of 0.83 on keyword assignment and F_2 of 0.32, Precision of 0.36 and Recall of 0.38 on predicting document category. On *'Computers and Internet'* Odds ratio achieves F_2 of 0.60, Precision of 0.40 and Recall of 0.84 on keyword assignment and F_2 of 0.33, Precision of 0.36 and Recall of 0.57 on predicting document category. On *'Education'* Term frequency and Odds ratio achieve the highest value of F_2 (0.55 and 0.48 respectively) on keyword assignment, while on category prediction the best result is achieved by Mutual information for text (F_2 is 0.45, Precision is 0.52 and Recall is 0.53). On *'Reference'* the best performing measure according to F_2 is Odds ratio again on both problems, achieving F_2 of 0.64, Precision of 0.51 and Recall of 0.81 on keyword assignment and F_2 of 0.42, Precision of 0.46 and Recall of 0.51 on

Domain name	Scoring measure	Average on keywords F1-measure	F2-measure	Average on categories F1-measure	F2-measure	Vs
Comp.	Odds ratio	0.46 ± 0.006	0.60 ± 0.006	0.30± 0.002	0.33 ± 0.002	1
	Log. prob. ratio for text	0.48 ± 0.007	0.61 ± 0.007	0.31± 0.007	0.33 ± 0.002	1
	Exp. prob. diff for text	0.40 ± 0.006	0.54 ± 0.007	0.27± 0.002	0.32 ± 0.002	1
	Weighted Odds ratio	0.44 ± 0.006	0.59 ± 0.006	0.29± 0.001	0.33 ± 0.002	1
	Conditional Odds ratio	0.47 ± 0.007	0.61 ± 0.006	0.31± 0.002	0.33 ± 0.002	1
Edu.	Odds ratio	0.33 ± 0.008	0.48 ± 0.008	0.32± 0.005	0.43 ± 0.005	1
	Log. prob. ratio for text	0.42 ± 0.009	0.56 ± 0.008	0.36± 0.005	0.44 ± 0.007	1
	Exp. prob. diff for text	0.33 ± 0.009	0.48 ± 0.008	0.30± 0.006	0.41 ± 0.006	1
	Weighted Odds ratio	0.37 ± 0.009	0.51 ± 0.007	0.32± 0.004	0.42 ± 0.006	1
	Conditional Odds ratio	0.42 ± 0.009	0.50 ± 0.008	0.36± 0.006	0.44 ± 0.008	1
Ref.	Odds ratio	0.42 ± 0.004	0.57 ± 0.006	0.32± 0.004	0.41 ± 0.007	1
	Log. prob. ratio for text	0.43 ± 0.004	0.58 ± 0.006	0.29± 0.005	0.37 ± 0.008	1
	Exp. prob. diff for text	0.52 ± 0.005	0.65 ± 0.004	0.41± 0.005	0.48 ± 0.009	0.5
	Weighted Odds ratio	0.54 ± 0.004	0.67 ± 0.004	0.41± 0.006	0.46 ± 0.007	0.5
	Conditional Odds ratio	0.43 ± 0.004	0.57 ± 0.006	0.30± 0.005	0.37 ± 0.010	1

Table 4.3: Comparison of variant of Odds ratio used for feature selection measures on domains formed from Yahoo hierarchy. We observe performance (F2-measure) on two problems: document keyword assignment and document category prediction for the best performing vector sizes (Vs). The values of F1-measure are included to show how the same feature scoring measures would compare in case we would have problem where Precision and Recall are equally important for the system performance. We give the correct category rank and probability for vector size 1, as one of the best performing vector sizes. Given results are for pruning setting = (0.7, 3), probability threshold 0.95 and vector size 1. Given are average values and standard error.

category prediction. Observation of standard error on all five domains confirms that the best performing measures are significantly better than the other tested measures.

Figures 4.7, 4.8 show how the same feature scoring measures compare if different parameter for combining Precision and Recall is used in F_β-measure. It can be seen that Odds ratio is among the best performing measures on both problems for all tested values of parameter β in F_β-measure. Mutual information for text is the best performing measure on both problems, if performance is measured using $F_{0.5}$ or F_1. On keyword assignment (Figure 4.7), Odds ratio, Cross entropy for text and Term frequency are the best performing measures. Information gain is one of the worst performing measures together with Random scoring and followed by Weight of evidence for text. On category prediction Odds ratio is clearly better than the other measures if using F_2, F_4, F_8 or F_{16}. Except Random scoring, all other measures achieve the best results for vector size around 1 on both problems.

4.3.3 Discussion

In our experiments the best performing features scoring measures are Odds ratio and its variants, while the worst performing are Random and Information gain. The other scoring measures we tested perform comparable or worse than Odds ratio and better than Information gain. A closer look to the highly scored features by Odds ratio and by Information gain (see Section 4.4) explains the huge difference in their performance and

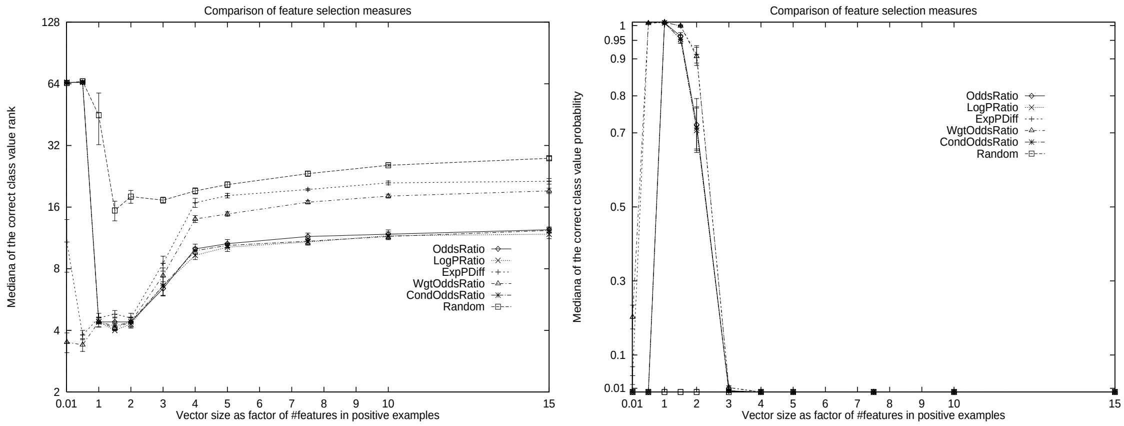

Figure 4.6: Comparison of Odds ratio and its variants on domain *'References'* defined from the Yahoo hierarchy. The problem is document categorization based on the given hierarchy of categorized documents. Given results are for pruning setting = (0.7, 3) and probability threshold 0.95. We give median of the correct category rank (left graphs) and probability (right graphs).

poor performance of Information gain. While Odds ratio favors features characteristic for positive examples (high $P(F|\text{'}pos')$), Information gain doesn't make distinction between the class values. Since we have unbalanced class distribution with most examples having negative class value (for most categories over 90%), most of the features are characteristic for negative examples. This means that most of the features highly scored by Information gain are characteristic for negative class value (just the opposite of Odds ratio!).

Previous findings vs. our findings

Yang and Pedersen [101] give experimental comparison of five measures for feature selection in text categorization on word-vectors representing documents. Their experiments confirm our observation that a rather small feature subset should be used since it gives either better or as good results as large feature subset or all the features. We also agree in the observation that a simple Frequency of feature (used after stop words removal and calculated either using term frequency or document frequency) achieves very good results. For larger subsets (vector size ≥ 4) the performance in rank of all methods became insensitive to the additional features (see Figure 4.4). For instance, on *'Reference'* for larger feature subsets, Odds ratio and Weight of evidence for text achieve about the same rank around 14, Random achieves rank 20-30, Term frequency around 40 while the other three measures achieve rank about 50. For larger feature subsets (vector size > 3) the predicted probability of the correct category is almost 0 for all the measures.

Information gain: good or bad for text-learning? Our results disagree in the performance of Information gain. Information gain was one of the best performing measures on domains used by Yang and Pedersen [101], while we found it performing similar or worse than random on our domains. The reason for that we find in the difference in our domains and classification algorithms.

We both use learning for document categorization but different approach to domain definition is used. Their machine learning domain is defined to include one class value for each category, while we split the problem into subproblems each corresponding to one category and having binary-valued class. The result of learning is in our case a set of specialized classifiers instead of one huge classifier including the union of features characteristic for different categories. Each of our subproblems is characterized by a highly unbalanced class distribution where the minority class values is the target class value (a document is a member of the category).

From the formula for Information gain it can be seen that there is no difference between the class values or between the feature values. In domains with unbalanced class distribution this means that most of the highly scored features are characteristic for the majority class value. Here we also have highly unbalanced feature value distribution. When calculating Information gain we observe two values for each feature (word sequence): occurs or does not occur in a document: $InfGain(F) = P(W)\sum_i P(C_i|W)\log\frac{P(C_i|W)}{P(C_i)} + P(\overline{W})\sum_i P(C_i|\overline{W})\log\frac{P(C_i|\overline{W})}{P(C_i)} = CrossEntropyTxt(F) + P(\overline{W})\sum_i P(C_i|\overline{W})\log\frac{P(C_i|\overline{W})}{P(C_i)}$. The probability that a word sequence occurs in a document $P(W)$ is rather small. Since no difference is made between the feature values, most of the highly scored features are features with the majority feature value ($P(\overline{W})$ is high). Intuitively, when classifying a new document, a better classification results are expected if the classification is based on the features (word sequences) that occur in a document. It is possible that absence of some words in a document is very informative and this is taken into account by the new feature scoring measure we named Conditional Odds ratio (4.14). The problem is that the classification based mostly on the absence of words is usually harder and requires larger feature subset than the classification based on word occurrences.

The other important issue is the used classification algorithm. Yang and Pedersen [101] used k-Nearest Neighbor and Linear Least Square Fit mapping. The specific of the Naive Bayesian classifier we are using is that it considers only features that occur in a classification document (see Section 3.2.1). This means that a highly scored feature should be features that will probably occur in new documents (have high $P(W)$). If $P(\overline{W}) >> P(W)$ then high value of Information gain means that the second part of Information gain formula $P(\overline{W})\sum_i P(C_i|\overline{W})\log\frac{P(C_i|\overline{W})}{P(C_i)}$ is high. In other words, knowing that W does not occur in a document brings useful information about the class value. On the other hand, the used classification algorithm is based only on the features that occur in a document. This means that most of the selected features will probably not occur in a document ($P(\overline{W})$ is high). If some of such features occur in a document, then the value of cross entropy between the class variable and the feature occurrence is important. In other words the first part of the formula is important but the feature was selected because of the high value in the second part of the formula. Our hypothesis that a good feature selection measure should take into account characteristics of the used classification algorithm is confirmed by the experiments. Cross entropy for text achieves very good results in our experiments. It equals Information gain calculated only for one of the feature values - W meaning word does occur.

In [101] the tested scoring measures are compared across several criteria: favoring common features, using class value, using feature absence and experimental performance. They conclude that the three measures (Frequency, Information gain and χ^2 statistics) achieving the best results on their domains (Reuters-22173, a subset of MEDLINE) all favor common features.

Conclusions and open questions

Table 4.4 gives characterization of the feature scoring measures compared in our experiments. We also observe whether the feature scoring measure makes difference between the class values, since this is important for our domains having one of the class values as the target class value. The best results are achieved by Odds ratio and its variants that assume the problem has binary-valued class and one of the class values is the target class value (asymmetric misclassification costs). Between these measures no significant difference in performance was observed. The next group of measures achieving good results all favor common features (Cross entropy for text, Term frequency, Weight of evidence for text). Mutual information for text differs from Cross entropy for text only in not favoring frequent features and achieves worse results. Information gain differs from Cross entropy for text only in using feature absence and achieves poor results.

Our conclusion is that in general the most important characteristics of a good feature scoring measure for text are: favoring common features and considering domain and algorithm characteristics. For learning algorithms that ignore feature absence (eg. the naive Bayesian classifier on Tf-document representation 3.2.1 we are using), it is important that a scoring measure also ignores feature absence. For the domains with binary-valued class where one class value is the target class value, the most important characteristics of a good feature scoring measure is to make difference between the class values and favor features characteristic for the target class value. In this case, favoring common features is not an important issue. Namely, Weighted odds ratio that favors common features is not performing better than Odds ratio.

Measure name	target class value	common features	using class value	using feature absence	performance
Odds ratio	**Yes**	No	Yes	No	great
Weighted Odds ratio	**Yes**	**Yes**	Yes	No	great
Log. prob. ratio	**Yes**	No	Yes	No	great
Exp. prob. difference	**Yes**	No	Yes	No	great
Conditional Odds ratio	**Yes**	No	Yes	No	great
Cross entropy for text	No	**Yes**	Yes	**No**	good-great
Term frequency	No	**Yes**	No	No	good
Weight of evidence for text	No	Yes	Yes	No	mild-good
Mutual information for text	No	*No*	Yes	No	mild
Information gain	No	Yes	Yes	*Yes*	poor
Random	No	No	No	No	poor

Table 4.4: Comparison of feature scoring measures by their characteristics and performance on out domains.

There are some open questions left. Here we used only the naive Bayesian classifier as one of the classification algorithms reported to work well on text domains [84]. The naive Bayesian classifier uses the same conditional probability as used in Odds ratio for scoring the features. In this way, the selected features are features expected to have the greatest influence to the posterior probability of class values returned by the naive Bayesian classifier. The question is how would some other classifier such as k-Nearest Neighbor perform in combination with Odds ratio or some of the other feature scoring measures. The approach used here for learning on hierarchy of examples is not the only possible approach. It would be interesting to compare it with some other approaches such as learning a separate classifier for each split in the text hierarchy and not for each node in the hierarchy as performed here. We have performed some initial experiments that indicated problems with categorization of more specific documents. Namely, for large hierarchies many words should be considered in the splits closer to the root of the hierarchy. The characteristic words contained in some specific nodes get almost lost when combined with more frequent words from other nodes. In this way, higher splits fail to direct the specific document to the right part of the hierarchy. If a specialized classifier is used instead as proposed in this thesis, the characteristic words are used by the specific classifier that is directly used on the document. Further experiments are also needed to see how the proposed learning from hierarchy performs on some other data. We show that only a small subset of features is needed for each of the specialized classifier. The question is how the proposed approach performs in comparison to the 'flat' approach where for each category one class value is defined and in what relation is the union of the selected features in all specialized classifiers to the number of features selected in a classifier constructed using the 'flat' approach.

4.4 Example of feature scoring and classification on text hierarchy

For illustration we give more detailed classification results Classification results including five highly ranked categories and the corresponding keywords (included in the name of each category) are given for the three best performing feature scoring measures: Odds ratio, Cross entropy for text, Term Frequency, and for Information gain as a well known measure from machine learning and here achieving poor results. Our experiments show that Odds ratio achieves the best results on the same domain for several vector sizes, eg., on *'References'* vector size 1-2, while Cross entropy for text achieves comparable results only for one vector size, eg., on *'References'* vector size 1. To show differences between these two measures we give in Tables 4.8, 4.9 the classification results for vector sizes 0.5, 1, 2.

We report result for one testing example that belongs to the domain *'References'*. This example is given in Table 5.3 and originally belongs to the category named `'Reference:Libraries:Library and Information Science:Information Retrieval'`. For Information gain and Frequency classification results for vector sizes 0.5 and 1 are given in Table 4.10. Table 4.11 gives the highly scored features from this category sorted by the assigned score for each of the four previously mentioned scoring measures. For instance, the highly scored feature by Odds ratio represents word `"IR"`, it is assigned score 5.28 and has the conditional probability in positive class examples 0.075 and in negative class examples 0.0004.

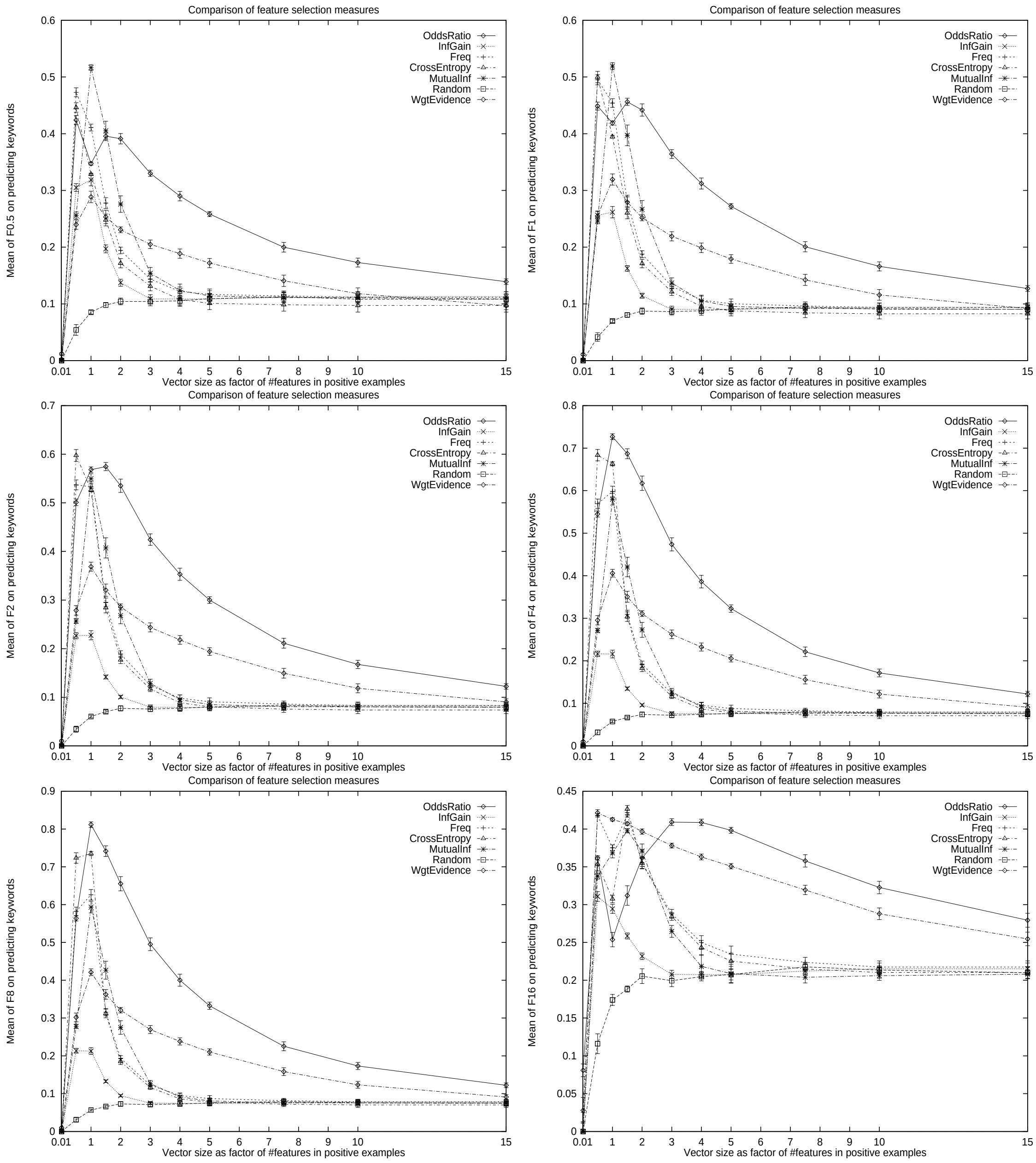

Figure 4.7: Comparison of F2-measure values for different feature scoring measures on domain *'Reference'* defined from the Yahoo hierarchy. The problem is document keyword assignment based on the given hierarchy of categorized documents and assigned keywords. Given results are for pruning setting = (0.7, 3) and probability threshold 0.95. We give mean and standard error of F-measure using different values of F-measure parameter 0.5, 1, 4, 6, 8, 16).

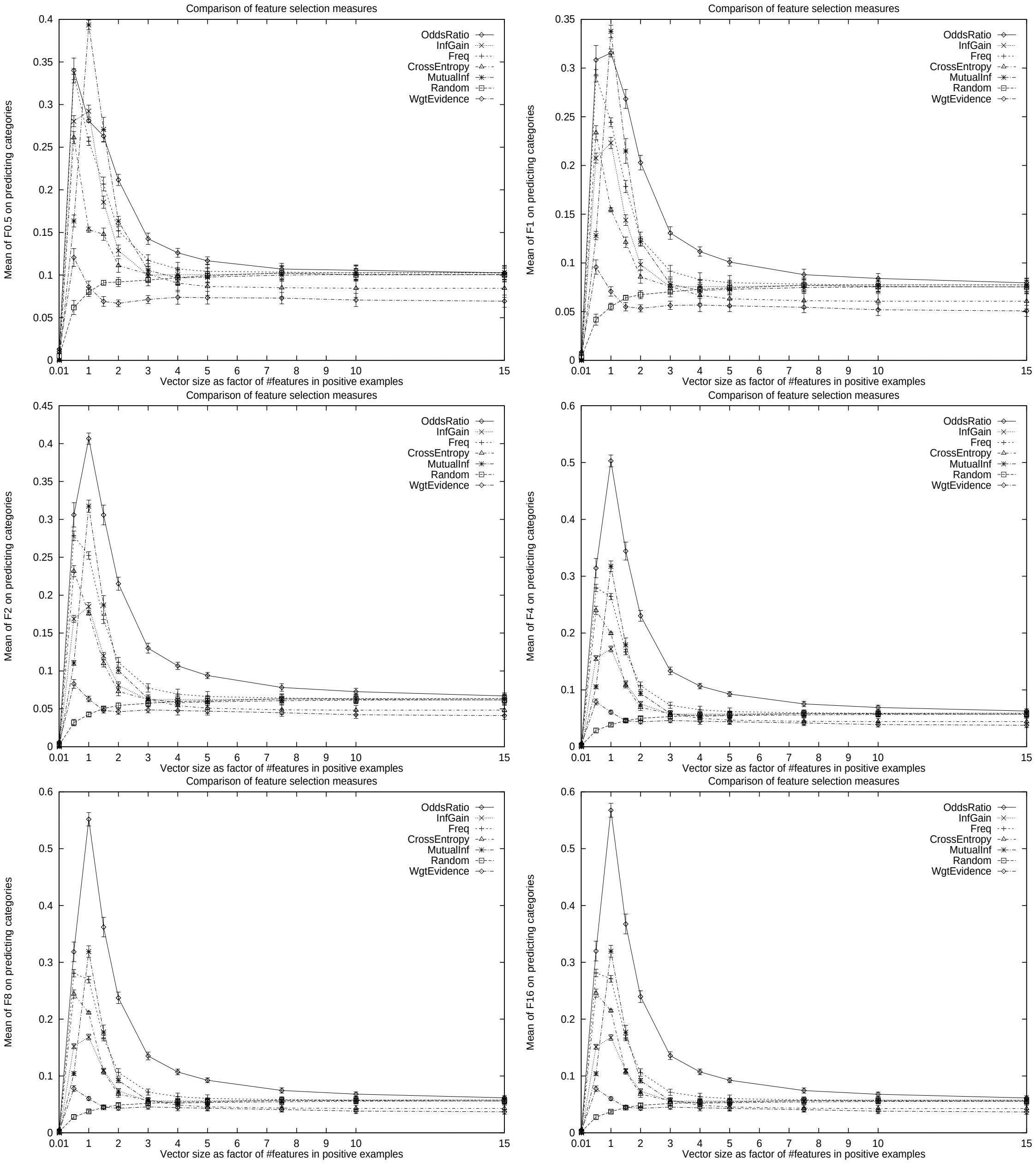

Figure 4.8: Comparison of F2-measure values for different feature scoring measures on domain *'Reference'* defined from the Yahoo hierarchy. The problem is document category prediction based on the given hierarchy of categorized documents. Given results are for pruning setting = (0.7, 3) and probability threshold 0.95. We give mean and standard error of F-measure using different values of F-measure parameter 0.5, 1, 4, 6, 8, 16).

Domain name	Scoring measure	Median of the correct category Rank	Probability	Average considered categories	Vector size
Ent.	**Odds ratio**	**16** ± 1.03	**0.96** ± 0.005	**115** ±2.1	**1**
	Weight of evidence for text	4.0 ± 0.6	0.65 ± 0.068	50 ±1.5	1
	Term frequency	49 ± 1.3	0.13 ± 0.045	756 ±4.1	1
	Random	4050 ± 15.8	< 0.001	30 ±0.8	1
	Information gain	4054 ± 13.5	< 0.001	1423±6.3	1
	Cross entropy for text	4103 ± 12.3	< 0.001	1612 ±15	1
	Mutual information for text	4197 ± 8.5	< 0.001	1461 ±6.2	1
Arts.	**Odds ratio**	**9.9** ± 1.12	**0.99** ± 0.004	**120** ±4.8	**1**
	Term frequency	15.2 ± 0.8	0.79 ± 0.057	299 ±7.1	1
	Weight of evidence for text	5.9 ± 0.6	0.16 ± 0.058	45 ±1.3	1
	Random	1549.2± 0.4	< 0.001	24 ±0.4	1
	Cross entropy for text	1559.8± 2.4	< 0.001	626 ±11	1
	Information gain	1591.9± 4.6	< 0.001	343 ±8.7	1
	Mutual information for text	1620 ± 5.2	< 0.001	558 ±8.6	1
Comp.	**Odds ratio**	**11.9** ± 0.96	**0.98** ± 0.007	**138** ±2.1	**1**
	Weight of evidence for text	10.0 ± 1.2	0.008 ± 0.004	44 ±1.0	1
	Term frequency	89.2 ± 9.6	0.002± 0.001	511 ±14	1
	Random	1332.6± 6.2	< 0.001	27 ±1	1
	Cross entropy for text	1354.7± 4.4	< 0.001	757 ±16	1
	Information gain	1392.8± 1.9	< 0.001	450 ±11	1
	Mutual information for text	1447.5± 6.2	< 0.001	739 ±16	1
Edu.	**Odds ratio**	**9.4** ± 0.98	**0.99** ± 0.002	**35** ±1.4	**1**
	Cross entropy for text	**10.8** ± 0.97	**0.98** ± 0.01	**162** ± 2	**1**
	Term frequency	29.5 ± 3.0	0.048± 0.03	113 ±2.2	0.5
	Weight of evidence for text	28.1 ± 3.80	< 0.001	17 ±0.5	1
	Mutual information for text	42.9 ± 4.12	0.03 ± 0.007	158 ±1.9	1
	Information gain	112.8 ± 3.80	< 0.001	144 ±2.2	1.5
	Random	178.4 ± 0.25	< 0.001	18 ±1.1	1
Ref.	**Cross entropy for text**	**3.3** ± 0.20	**0.99** ± 0.009	**46** ±1.1	**0.5**
	Odds ratio	**3.8** ± 0.38	**0.99** ± 0.002	**15** ±0.6	**1**
	Term frequency	3.8 ± 0.20	0.64 ± 0.07	38 ±0.8	0.5
	Mutual information for text	6.9 ± 0.33	0.54 ± 0.09	51 ±1.0	1
	Weight of evidence for text	9.6 ± 0.07	< 0.001	9.2±0.3	4
	Information gain	27.4 ± 3.8	<0.001	38 ±0.8	0.5
	Random	65.8 ± 0.37	< 0.001	8 ±0.4	1

Table 4.5: Comparison of feature selection measures on domains formed from Yahoo hierarchy. The problem is document categorization based on the given hierarchy of categorized documents. We give the correct category rank and probability for the best performing vector sizes. Feature scoring measures are sorted according to their performance in each domain. Given results are for pruning setting = (0.7, 6) and probability threshold 0.95. Additionally to the average values, we report standard errors, since all the reported results are averages over five runs on independent testing sets.

Domain name	Scoring measure	Average on keyword assignment				Vs
		F1-measure	F2-measure	Precision	Recall	
Ent.	**Odds ratio**	**0.48** ± 0.006	**0.59** ± 0.007	**0.44**± 0.006	**0.80** ± 0.006	1
	Weight of evidence for text	**0.48** ± 0.002	0.52 ± 0.004	0.74 ± 0.003	0.58 ± 0.003	1
	Term frequency	0.39 ± 0.003	0.49 ± 0.12	0.41± 0.006	0.71 ± 0.010	1
	Cross entropy for text	0.29 ± 0.007	0.39 ± 0.007	0.35± 0.003	0.69 ± 0.007	1
	Mutual information for text	0.25 ± 0.005	0.27 ± 0.006	0.57± 0.007	0.38 ± 0.007	1
	Information gain	0.27 ± 0.008	0.22 ± 0.004	0.86± 0.005	0.21 ± 0.006	1
	Random	0.002± 0.001	0.002± 0.001	0.99± 0.006	0.001± 0.007	1
Arts.	**Odds ratio**	0.46 ± 0.003	**0.59** ± 0.005	**0.40**± 0.002	**0.83** ± 0.006	1
	Term frequency	**0.47** ± 0.007	**0.58** ± 0.008	**0.48**± 0.003	**0.77** ± 0.009	1
	Weight of evidence for text	0.50 ± 0.012	0.56 ± 0.012	0.70 ± 0.001	0.62 ± 0.012	1
	Cross entropy for text	0.32 ± 0.003	0.44 ± 0.004	0.33± 0.004	0.75 ± 0.008	1
	Mutual information for text	0.31 ± 0.005	0.35 ± 0.004	0.56± 0.007	0.46 ± 0.006	1
	Information gain	0.25 ± 0.006	0.21 ± 0.005	0.94± 0.002	0.20 ± 0.004	1
	Random	0.0051± 0.001	0.001± 0.001	0.99± 0.001	0.001± 0.001	1
Comp.	**Odds ratio**	0.46 ± 0.006	**0.60** ± 0.006	**0.40**± 0.007	**0.84** ± 0.005	1
	Term frequency	**0.48** ± 0.008	**0.58** ± 0.007	**0.50**± 0.004	**0.74** ± 0.005	1
	Weight of evidence for text	0.46 ± 0.002	0.51 ± 0.11	0.76± 0.007	0.56 ± 0.010	1
	Cross entropy for text	0.37 ± 0.007	0.49 ± 0.008	0.35± 0.004	0.75 ± 0.007	1
	Mutual information for text	0.36 ± 0.003	0.38 ± 0.003	0.62± 0.005	0.45 ± 0.005	1
	Information gain	0.21 ± 0.005	0.17 ± 0.005	0.94± 0.005	0.15 ± 0.005	1
	Random	0.01 ± 0.001	0.01 ± 0.001	0.99 ± 0.001	0.004± 0.001	1
Edu.	**Term frequency**	**0.48** ± 0.007	**0.55** ± 0.007	**0.57**± 0.010	**0.65** ± 0.010	0.5
	Odds ratio	0.33 ± 0.008	**0.48** ± 0.008	**0.36**± 0.010	**0.81** ± 0.005	1
	Mutual information for text	0.40 ± 0.004	0.46 ± 0.007	0.48± 0.010	0.59 ± 0.010	1
	Cross entropy for text	0.32 ± 0.009	0.46 ± 0.007	0.28± 0.010	0.82 ± 0.006	1
	Weight of evidence for text	0.30 ± 0.007	0.34 ± 0.006	0.69± 0.007	0.40 ± 0.005	1
	Information gain	0.13 ± 0.007	0.11 ± 0.006	0.98± 0.002	0.11 ± 0.006	1.5
	Random	0.01 ± 0.002	0.01 ± 0.002	0.99± 0.001	0.003± 0.002	1
Ref.	**Odds ratio**	**0.53** ± 0.006	**0.64** ± 0.006	**0.51**± 0.007	**0.81** ± 0.008	**1**
	Cross entropy for text	0.52 ± 0.008	**0.60** ± 0.010	**0.62**± 0.003	**0.71** ± 0.010	0.5
	Mutual information for text	0.52 ± 0.010	0.55 ± 0.010	0.73± 0.010	0.60 ± 0.020	1
	Term frequency	0.50 ± 0.010	0.53 ± 0.010	0.78± 0.005	0.57 ± 0.010	0.5
	Weight of evidence for text	0.20 ± 0.008	0.22 ± 0.010	0.80± 0.009	0.31 ± 0.010	4
	Information gain	0.25 ± 0.007	0.22 ± 0.007	0.99± 0.002	0.21 ± 0.006	0.5
	Random	0.07 ± 0.006	0.06 ± 0.005	0.99± 0.001	0.05 ± 0.005	1

Table 4.6: Comparison of feature selection measures on domains formed from Yahoo hierarchy. The problem is document keyword assignment. We observe performance (F2-measure) for the best performing vector sizes (Vs). Additionally, we give values of Precision, Recall and F1-measure. The values of F1-measure are included to show how the same feature scoring measures would compare in case we would have problem where Precision and Recall are equally important for the system performance. Feature scoring measures are sorted according to their performance in F2-measure. Given results are for pruning setting = (0.7, 6) and probability threshold 0.95. We report the average values and standard errors.

Domain name	Scoring measure	Average on category prediction F1-measure	F2-measure	Precision	Recall	Vs
Ent.	**Odds ratio**	**0.29**± 0.002	**0.30** ± 0.003	**0.41** ± 0.004	**0.34** ± 0.003	**1**
	Term frequency	0.24± 0.003	0.27 ± 0.003	0.38 ± 0.003	0.34 ± 0.003	1
	Mutual information for text	0.22± 0.004	0.23 ± 0.004	0.57 ± 0.006	0.29 ± 0.007	1
	Information gain	0.25± 0.002	0.20 ± 0.003	0.87 ± 0.002	0.17 ± 0.002	1
	Cross entropy for text	0.15± 0.002	0.18 ± 0.002	0.29 ± 0.005	0.28 ± 0.005	1
	Weight of evidence for text	0.14 ± 0.003	0.11 ± 0.002	0.67 ± 0.005	0.10 ± 0.003	1
	Random	0.001± 0.0001	0.001± 0.0002	0.99 ± 0.007	0.001±0.0002	1
Arts.	**Odds ratio**	**0.29**± 0.002	**0.32** ± 0.004	**0.36** ± 0.005	**0.38** ± 0.004	**1**
	Term frequency	0.28± 0.002	0.29 ± 0.003	0.43 ± 0.004	0.34 ± 0.003	1
	Mutual information for text	0.24± 0.005	0.25 ± 0.005	0.56 ± 0.006	0.31 ± 0.007	1
	Cross entropy for text	0.18± 0.002	0.22 ± 0.003	0.27 ± 0.005	0.32 ± 0.006	1
	Information gain	0.21± 0.002	0.17 ± 0.002	0.93 ± 0.003	0.15 ± 0.002	1
	Random	0.0012± 0.0001	0.001± 0.0003	0.99 ± 0.006	0.001±0.0002	1
	Weight of evidence for text	0.11 ± 0.003	0.09 ± 0.002	0.55 ± 0.004	0.079± 0.002	1
Comp.	**Odds ratio**	**0.30**± 0.002	**0.33** ± 0.002	**0.36** ± 0.009	**0.57** ± 0.005	**1**
	Term frequency	0.27± 0.003	0.26 ± 0.002	0.45 ± 0.003	0.27 ± 0.003	1
	Mutual information for text	0.25± 0.003	0.24 ± 0.004	0.60 ± 0.006	0.26 ± 0.006	1
	Cross entropy for text	0.19± 0.005	0.21 ± 0.004	0.28 ± 0.004	0.27 ± 0.002	1
	Information gain	0.19± 0.007	0.14 ± 0.006	0.94 ± 0.004	0.12 ± 0.005	1
	Weight of evidence for text	0.09± 0.002	0.07 ± 0.002	0.59 ± 0.010	0.06 ± 0.001	1
	Random	0.001± 0.0003	0.001± 0.0002	0.99 ± 0.001	0.001± 0.0002	1
Edu.	**Mutual information for text**	**0.40**± 0.006	**0.45** ± 0.009	**0.52** ± 0.010	**0.53** ± 0.010	**1**
	Odds ratio	0.32± 0.005	**0.43** ± 0.005	**0.36** ± 0.009	**0.57** ± 0.010	**1**
	Term frequency	**0.40**± 0.010	**0.42** ± 0.010	**0.57** ± 0.010	**0.45** ± 0.020	**0.5**
	Weight of evidence for text	0.05± 0.006	**0.42** ± 0.005	**0.53** ± 0.020	**0.04** ± 0.005	**1**
	Cross entropy for text	0.2 ± 0.006	0.26 ± 0.004	0.23 ± 0.010	0.37 ± 0.005	1
	Information gain	0.09± 0.003	0.07 ± 0.002	0.97 ± 0.003	0.07 ± 0.002	1.5
	Random	0.01± 0.001	0 ± 0.001	0.99 ± 0.002	0.001± 0.001	1
Ref.	**Odds ratio**	**0.37**± 0.007	**0.42** ± 0.009	**0.46** ± 0.009	**0.51** ± 0.012	**1**
	Mutual information for text	0.34± 0.005	0.32 ± 0.005	0.69 ± 0.015	0.32 ± 0.006	1
	Term frequency	0.28± 0.007	0.26 ± 0.070	0.72 ± 0.007	0.26 ± 0.007	0.5
	Cross entropy for text	0.23± 0.005	0.22 ± 0.005	0.50 ± 0.003	0.23 ± 0.005	0.5
	Information gain	0.20± 0.003	0.16 ± 0.002	0.99 ± 0.002	0.14 ± 0.002	0.5
	Weight of evidence for text	0.06± 0.007	0.05 ± 0.006	0.70 ± 0.011	0.05 ± 0.004	4
	Random	0.05± 0.006	0.04 ± 0.005	0.99 ± 0.001	0.04 ± 0.004	1

Table 4.7: Comparison of feature selection measures on domains formed from Yahoo hierarchy. The problem is document category prediction. We observe performance (F2-measure) for the best performing vector sizes (Vs). Additionally, we give values of Precision, Recall and F1-measure. The values of F1-measure are included to show how the same feature scoring measures would compare in case we would have problem where Precision and Recall are equally important for the system performance. Feature scoring measures are sorted according to their performance in F2-measure on keyword assignment. Given results are for pruning setting = (0.7, 6) and probability threshold 0.95. We report the average values and standard errors.

Odds ratio with vector size 0.5:	
'Reference:Libraries'	0.997
'Reference:Libraries:University Libraries'	0.814
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.78**
'Reference:Libraries:Library and Information Science:Institutes'	0.73
'Reference:Libraries:Resources'	0.66
Odds ratio with vector size 1:	
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.999**
'Reference:Libraries'	0.998
'Reference:Libraries:Indices'	0.952
'Reference:Libraries:Library and Information Science:Institutes'	0.938
'Reference:Libraries:Library and Information Science'	0.933
Odds ratio with vector size 2:	
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.999**
'Reference:Libraries'	0.996
'Reference:Libraries:Resources:Indices'	0.883
'Reference:Libraries:Resources:Internet in Libraries'	0.71
'Reference:Dictionaries:Indices'	0.64

Table 4.8: Five categories with the highest predicted probability for Odds ratio. We report results for different size of selected feature subset (three vector sizes: 0.5, 1, 2). Vector size 1 seems to be the best size for our document categorization. Namely, the best performing feature scoring measures achieve the best results when the number of selected features is the same as the number of features that occur in positive documents (vector size 1). The predicted probability assigned to the correct example category is given in boldface.

Cross entropy for text with vector size 0.5:	
'Reference:Libraries'	0.997
'Reference:Libraries:Indices'	0.952
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.928**
'Reference:Libraries:Resources:Indices'	0.883
'Reference:Standards'	0.82
Cross entropy for text with vector size 1:	
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.999**
'Reference:Libraries'	0.987
'Reference:Libraries:Indices'	0.952
'Reference:Libraries:Library and Information Science'	0.933
'Reference:Libraries:Resources:Indices'	0.883
Cross entropy for text with vector size 2:	
'Reference:Libraries'	0.987
'Reference:Libraries:Resources'	0.006
'Reference:Libraries:Resources:Indices'	0.0024
'Reference:Libraries:Indices'	0.0023
'Reference:Libraries:Library and Information Science:Information Retrieval'	**0.0013**

Table 4.9: Five categories with the highest predicted probability for Cross entropy for text. We report results for different size of selected feature subset (three vector sizes: 0.5, 1, 2). Vector size 1 seems to be the best size for our document categorization. Namely, the best performing feature scoring measures achieve the best results when the number of selected features is the same as the number of features that occur in positive documents (vector size 1). The predicted probability assigned to the correct example category is given in boldface.

Information gain with vector size 0.5:	
`'Reference:Libraries'`	0.969
`'Reference'`	0.668
`'Reference:Libraries:Library and Information Science:Information Retrieval'`	**0.077**
`'Reference:Libraries:Resources:Indices'`	0.056
`'Reference:Libraries:Resources'`	0.025
Information gain with vector size 1:	
`'Reference:Libraries'`	0.999
`'Reference'`	0.75
`'Reference:Libraries:Resources'`	0.025
`'Reference:Libraries:University Libraries'`	0.019
`'Reference:Libraries:Library and Information Science'`	0.002
Frequency with vector size 0.5:	
`'Reference:Libraries:Library and Information Science:Information Retrieval'`	**0.978**
`'Reference:Libraries'`	0.967
`'Reference:Libraries:Library and Information Science'`	0.826
`'Reference:Libraries:University Libraries'`	0.669
`'Reference'`	0.668
Frequency with vector size 1:	
`'Reference:Libraries:Library and Information Science:Information Retrieval'`	**0.999**
`'Reference:Libraries'`	0.998
`'Reference:Libraries:Resources:Indices'`	0.88
`'Reference'`	0.75
`'Reference:Libraries:Resources:Internet in Libraries`	0.71

Table 4.10: Five categories with the highest predicted probability for Information gain and Frequency. We report results for different size of selected feature subset (two vector sizes: 0.5, 1). Vector size 1 seems to be the best size for our document categorization. Namely, the best performing feature scoring measures achieve the best results when the number of selected features is the same as the number of features that occur in positive documents (vector size 1). The predicted probability assigned to the correct example category is given in boldface.

sorted by Odds ratio score	**sorted by Cross entropy for text score**
IR 5.28 [0.075 0.0004]	INFORMATION 0.095 [0.12 0.021]
INFORMATION RETRIEVAL 5.13 [0.075 0.0005]	UNIVERSITY 0.081 [0.045 0.028]
RETRIEVAL 4.77 [0.0746 0.0007]	RESOURCES 0.039 [0.03 0.011]
GLASGOW 4.72 [0.03 0.0003]	LIBRARIES 0.038 [0.015 0.026]
ASIA 4.32 [0.03 0.0004]	LIBRARY 0.037 [0.015 0.091]
PACIFIC 4.02 [0.015 0.0003]	REFERENCE 0.031 [0.015 0.012]
INTERESTING 4.02 [0.015 0.0003]	LINKS 0.025 [0.015 0.0082]
EMPHASIS 4.02 [0.015 0.0003]	PAGES 0.021 [0.015 0.0061]
GROUP 3.64 [0.045 0.0012]	INCLUDES 0.02 [0.015 0.0055]
MASSACHUSETTS 3.46 [0.015 0.0005]	RESEARCH 0.0197 [0.023 0.0041]
COMMERCIAL 3.46 [0.015 0.0005]	SCIENCE 0.016 [0.015 0.0039]
REGION 3.1 [0.015 0.0007]	COLLECTIONS 0.015 [0.015 0.0037]
INFORMATION RESOURCES 3.004 [0.015 0.0008]	WEB 0.015 [0.015 0.0037]
SYSTEMS 2.76 [0.015 0.001]	UK 0.015 [0.015 0.0036]
GLOSSARY 2.69 [0.015 0.001]	SITES 0.0128 [0.015 0.0029]
MATERIALS 2.63 [0.015 0.001]	DIGITAL 0.011 [0.015 0.0025]
DOCUMENTS 2.63 [0.015 0.001]	PEOPLE 0.011 [0.015 0.0024]
CENTRE 2.51 [0.015 0.0012]	CATALOGING 0.01 [0.015 0.0021]
LISTS 2.46 [0.015 0.0013]	GROUP 0.0075 [0.045 0.00123]
sorted by Information gain score	**sorted by Frequency score**
LIBRARY 0.46 [0.015 0.091]	INFORMATION 0.14 [0.119 0.0207]
PUBLIC 0.23 [0 0.034]	LIBRARY 0.106 [0.015 0.091]
PUBLIC LIBRARY 0.21 [0 0.029]	RETRIEVAL 0.075 [0.075 0.00068]
UNIVERSITY 0.21 [0.045 0.028]	INFORMATION RETRIEVAL 0.075 [0.075 0.00048]
LIBRARIES 0.197 [0.015 0.026]	IR 0.075 [0.075 0.00041]
INFORMATION 0.17 [0.119 0.021]	UNIVERSITY 0.073 [0.0448 0.028]
REFERENCE 0.117 [0.015 0.012]	GROUP 0.046 [0.0448 0.00123]
RESOURCES 0.11 [0.0299 0.0102]	LIBRARIES 0.04099 [0.015 0.02606]
COUNTY 0.096 [0 0.0089]	RESOURCES 0.04009 [0.0299 0.0102]
INTERNET 0.091 [0 0.00826]	RESEARCH 0.0339 [0.02985 0.004094]
LINKS 0.091 [0.015 0.00819]	PUBLIC 0.03371 [0 0.0337]
SERVICES 0.089 [0 0.0079]	ASIA 0.03026 [0.0299 0.0004094]
STATE 0.087 [0 0.0077]	GLASGOW 0.03012 [0.0299 0.00027]
PAGES 0.076 [0.015 0.0061]	PUBLIC LIBRARY 0.0287 [0 0.0287]
ASSOCIATION 0.075 [0 0.006]	REFERENCE 0.027 [0.015 0.012]
INCLUDES 0.072 [0.015 0.0055]	LINKS 0.023 [0.015 0.0082]
COLLEGE 0.0701 [0 0.0053]	PAGES 0.021 [0.015 0.0061]
DIRECTORY 0.0701 [0 0.0053]	INCLUDES 0.0205 [0.015 0.00553]
ONLINE 0.069 [0 0.0052]	SCIENCE 0.0188 [0.015 0.00389]
SCHOOL 0.069 [0 0.0052]	COLLECTIONS 0.0186 [0.015 0.00368]
	0.0073 [0.015 0.0014]

Table 4.11: Highly score features by four different feature scoring measures: Odds ratio, Cross entropy for text, Information gain, Frequency. The problem is document categorization and the reported results are for one of the subproblems defined for category `‘Reference:Libraries:Library and Information Science:Information Retrieval’`. For each feature we give: word sequence represented by the feature, feature score, [$P(W|‘pos’)$-probability in positive class, $P(W|‘neg’)$-probability in negative class].

Chapter 5

Learning from class hierarchy

Machine learning techniques enable inclusion of domain specific knowledge, usually referred to as background knowledge in the process of learning. Usually, domain knowledge is provided by domain expert in different forms and used either directly by some learning algorithm, eg. inductive logic programming algorithm [81, 63] or transformed and incorporated in domain examples. We incorporate domain specific background knowledge in data representation by generating new features (see Section 3.1).

This Chapter describes the approach we propose for automatic construction of background knowledge from an example domain containing information that can be seen as a background knowledge for our problem of predicting clicked hyperlinks. Section 5.1 describes the hierarchical data structure on which we based the automatic construction of background knowledge. Machine learning techniques that we used to learn from class hierarchy are described in Section 5.2. Experimental evaluation of the proposed approach is given in Section 5.3.

5.1 Domain description

We dynamically constructed the background knowledge using as example domain the existing Yahoo Web hierarchy [27]. The categories in it are human constructed and designed for human Web browsing. Documents that are already classified and used to build the hierarchy are Web documents, making the hierarchy biased toward human knowledge areas that are represented in Web documents. For example, one of the most general categories named *'Bussines and Economy'* is over-represented including more than a third of all documents, while the other category that appears at the same level of the hierarchy named *'Social Science'* includes less than 1% of all documents (see Figure 5.1). In our case this is not expected to be a problem since, in our problems (see Section 2.1) we are dealing with Web documents anyway. Our approach to learning from class hierarchy is not limited to the Yahoo hierarchy and can be used on some of the other hierarchies like thesaurus (eg. Roget thesaurus [91]) that are not Web-oriented.

The Yahoo hierarchy itself (excluding subcategory *'Regional'*) is currently built on approximately a million Web documents located on Internet all around the world. We refer here to these documents as *actual Web documents*. Hyperlinks to those documents are organized in about 50,000 *Yahoo Web documents*. Each Yahoo document represents one of the included categories with the more general categories closer to the root of the

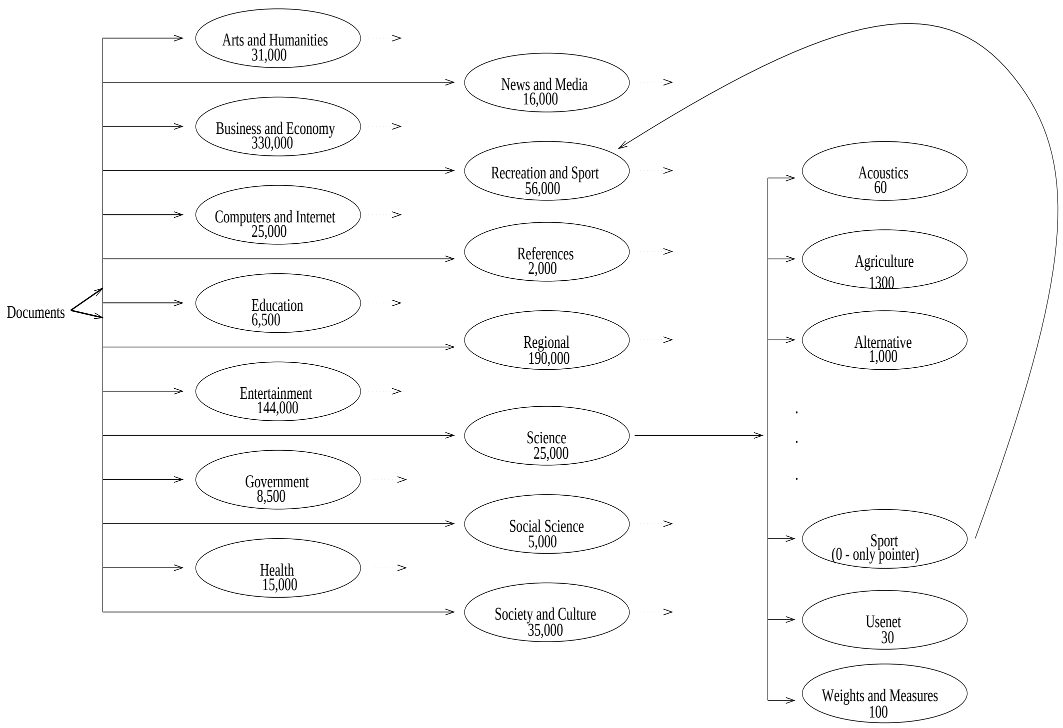

Figure 5.1: Top level of the Yahoo categorization with the first level of subcategories in *'Science'* category. Approximate number of documents in each category (the whole subtree in the hierarchy) is given under the category name (eg. 25,000 documents are classified under *'Science'*). The data is taken from the Yahoo site in UK & Ireland, November 1997.

tree. The category is denoted by keywords, describing category content, that appear on the path from the root of the hierarchy to the node representing the category. For instance, *'Sport'* a subcategory of *'Science'* in Figure 5.1 is named *'Science: Sport'* and in our approach assigned two keywords: *Science, Sport*. In other words, a more specific category is named by adding a keyword to the name of the more general category directly connected to it (one level higher in the hierarchy).

Yahoo documents are connected with hyperlinks forming a hierarchical structure that can be represented as a directed acyclic graph (or a tree with additional connections between some of its nodes). For the sake of simplicity, we will talk about the tree and remind on additional connections only when needed. Each Web document classified in the Yahoo hierarchy appears only once, but there can be several connections in the hierarchy (hyperlinks between Yahoo documents) leading to it. For example, the Yahoo document representing *'Sport'*, a subcategory of *'Science'* includes a hyperlink to the top category *'Recreation and Sport'* (see Figure 5.1). In this way, user still has the impression of a tree structure that nicely matches our intuition about the hierarchical organization of categories. Some nodes at the bottom of the hierarchy contain mostly hyperlinks to the actual Web documents, while the other nodes contain mostly or even only (eg. *'Science: Sport'* in Figure 5.1) hyperlinks to other Yahoo Web documents (nodes in the hierarchy). There are currently fourteen top level Yahoo categories each named by only one keyword. Figure 5.1 shows them with the approximate number of actual Web documents each category is based on. Each of the top categories is further represented with a hierarchical structure of more specific categories. As an example we show in Figure 5.1 the part of the first-level subcategories in *'Science'* top category ranging from *'Acoustics'* (named *'Science: Acoustics'* and assigned two keywords *Science, Acoustics*) to *'Weights and Measures'* (named *'Science: Weights and Measures'* and assigned two keywords *Science, Weights and Measures*).

5.2 Machine learning approach

In order to learn from class hierarchy we induce a classifier for each of the hierarchy nodes (leaf or non-leaf) by collecting word probabilities from all the documents in and below the node. A new document is assigned the class values it fits best. Since each document can be assigned several class values (here each class value represents one topic category), we assign to the document all the class values whose predicted probability is ≥ 0.95. Notice that for each class value an independent prediction is performed on the problem of document being a member of the corresponding topic category or not. Each class value represents one Yahoo category and each category in the Yahoo hierarchy is denoted by a set of keywords. Classifying document in Yahoo hierarchy means assigning categories and at the same time assigning a set of keywords to the document. This can be seen as a sort of automatic document abstraction. Notice that the keyword assignment as proposed here (based on the category assignment) is preconditioned by the existence of the underlining hierarchy of categories and the keywords associated with each category. Assigned categories and keywords can be further used to enrich document representation as proposed in Section 3.1.2.

5.2.1 Related work

The problem domain is represented as a classification hierarchy of examples. There is some work in machine learning addressing domains with hierarchy of features (eg. [94]) but domains with example or classification hierarchy have been only recently explored. The only closely related work we are aware of is the recent work on hierarchically classifying documents by Koller and Sahami [53]. They used the Reuters dataset that 'does not have a predetermined hierarchical classification scheme', so they 'identified labels which tended to subsume other labels, and used those as the higher level topics'. In their hierarchy all documents are placed at the bottom of the hierarchy in tree leaves. Documents are represented as Boolean word-vectors, as described in Section 1.2.1. Features are selected using greedy algorithm that eliminates features one by one using Cross entropy for text (see Section 4.2.3) as a feature scoring measure.

They compared several learning algorithms used in two different approaches to hierarchical classification. The first approach is defined as simply ignoring hierarchical structure (the structure is 'flattened') and constructing one huge classifier with one class for each leaf in the hierarchy. The second approach utilizes the hierarchical structure, dividing classification task into a set of smaller problems corresponding to the splits in the classification hierarchy. Each classifier needs only to distinguish between a small number of categories and can therefore be based only on a small subset of features. Their results are on three domains each having a 3-level hierarchy that is based on up to 1,000 documents and the biggest having 12 nodes.

The Yahoo hierarchy and the defined domains we are using here are much bigger, have a predetermined hierarchical classification structure with documents not necessarily placed only in the hierarchy leaves. This means that some documents sitting in the non-leaf nodes are too general to be classified into any of the existing leaf nodes. A classifier using these non-leaf documents should consider classification into any of the hierarchy nodes (not only leaf nodes). We use the Yahoo hierarchy for two learning problems: assign categories and assign a set of keywords to a document. For a new document, the learned model returns for each category (and the corresponding set of keywords) included in the text hierarchy the probability that the document is its member. One possibility is to use 'flattened' approach and generate one huge classifier with many class values, each value corresponding to one category. Additionally to the high number of class values, a large number of features is needed in order to cover all the variety of the large number of documents included in different categories. As pointed out in [53], a better idea is to split the whole problem into subproblems and use a local feature subset selection on each subproblem. We use the hierarchical structure to define subproblems, each corresponding to the individual Yahoo category. For each of the subproblems, a classifier is constructed that predicts the probability for a document to be a member of the corresponding category and thus be characterized by the corresponding set of keywords.

Documents are represented as feature-vectors as described in Section 5.2.2. A set of positive and negative examples for each subproblem are constructed from the given hierarchical structure as described in Sections 5.2.3,5.2.4. Learning using the naive Bayesian classifier is described in Section 5.2.5. The final result of learning is a set of specialized classifiers each based only on a small subset of features.

5.2.2 Document representation

Document categorization is here based on the usage of machine learning techniques on large text data. We based document representation on feature-vectors using word sequences as described in Section 3.1.1 and given in Figure 5.2. By using word sequences we get on the whole Yahoo hierarchy (excluding *'Regional'*) from about 70,000 features representing single words, about 250,000 features representing word sequences of maximally five words. We interweaved the feature selection mechanisms with the feature construction such that the feature selection removing words that appear in a "stop-list" and removing infrequent features is applied after each pass over documents performed in feature construction (see Section 3.1.1). For instance, take the Yahoo category named `'Reference:Libraries:Library and Information Science:Information Retrieval'` and represented with the Yahoo document in Figure 5.3. This category node includes the text given in Table 5.1 and it is by our system represented as shown in Table 5.1.

The generation of new features with this mechanism is show in Figure 3.2. The resulting features enter the process of feature subset selection. It is performed as described in Chapter 4 using Odds ratio as a feature scoring measure, since it was one of the best performing measures on our text data (see Section 4.3). The 20 most frequent features among 255,602 features used in the whole Yahoo hierarchy represent the following words: `services`, `information`, `page`, `web`, `internet`, `software`, `links`, `systems`, `design`, `university`, `computer`, `products`, `service`, `site`, `home`, `music`, `busi-`

Original text on Yahoo Web document:	The same document represented by frequent n-grams (see Figure 5.2):
1. Top:Reference:Libraries:Library and Information Science:Information Retrieval	1. "REFERENCE LIBRARIES LIBRARY INFORMATION SCIENCE (#3 LIBRARY INFORMATION SCIENCE) INFORMATION RETRIEVAL (#2 INFORMATION RETRIEVAL)"
2. UK Only	2. "UK"
3. Idomeneus - IR & DB repository - These pages mostly contain IR related resources such as test collections, stop lists, stemming algorithms, and links to other IR sites.	3. "IR PAGES IR RELATED RESOURCES COLLECTIONS LISTS LINKS IR SITES"
4. University of Glasgow - Information Retrieval Group - information on the resources and people in the Glasgow IR group.	4. "UNIVERSITY GLASGOW INFORMATION RETRIEVAL (#2 INFORMATION RETRIEVAL) GROUP INFORMATION RESOURCES (#2 INFORMATION RESOURCES) PEOPLE GLASGOW IR GROUP"
5. Centre for Intelligent Information Retrieval (CIIR).	5. "CENTRE INFORMATION RETRIEVAL (#2 INFORMATION RETRIEVAL)"
6. Information Systems Asia Web - provides research, IS-related commercial materials, interaction, and even research sponsorship by interested corporations with a focus on Asia Pacific region.	6. "INFORMATION SYSTEMS ASIA WEB RESEARCH COMMERCIAL MATERIALS RESEARCH ASIA PACIFIC REGION"
7. Seminar on Cataloging Digital Documents	7. "CATALOGING DIGITAL DOCUMENTS"
8. Survey of Information Retrieval - guide to IR, with an emphasis on web-based projects. Includes a glossary, and pointers to interesting papers.	8. "INFORMATION RETRIEVAL (#2 INFORMATION RETRIEVAL) GUIDE IR EMPHASIS INCLUDES GLOSSARY INTERESTING"
9. University of Dortmund - Information Retrieval Group	9. "UNIVERSITY INFORMATION RETRIEVAL (#2 INFORMATION RETRIEVAL) GROUP"

Table 5.1: Example of document representation on a Yahoo document. The original text from the Yahoo document is given (left hand-side) together with the representation of this text by features representing word sequences (right hand-side).

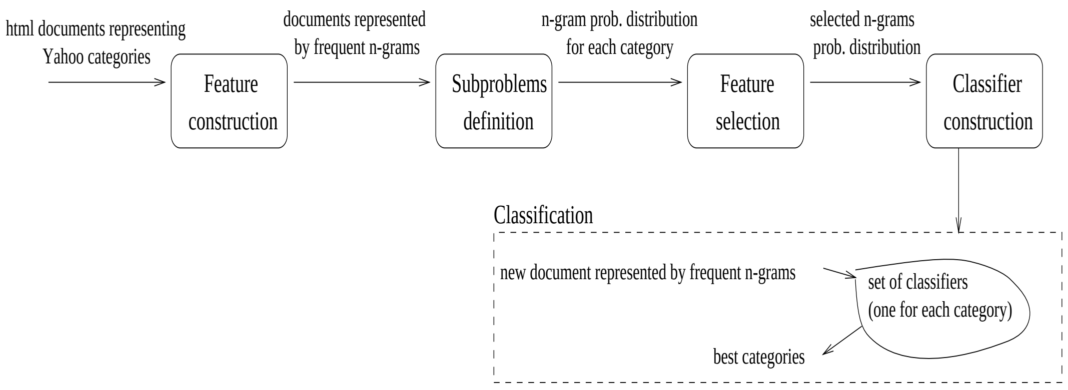

Figure 5.2: Usage of word sequences to enrich document representation when learning on class hierarchy. The approach is used for the problem of category prediction and the problem of keyword assignment.

`ness`, `international`, `online`, `world` having frequency from 37,792 (`services`) to 15,056 (`world`). Features with the lowest frequency among included features (having frequency 4) mostly represent word sequences (eg., `worldwide marketing website design service`, `funded programme provide assistance training`). The proportion of words with higher frequency is much smaller than the low frequency words. For instance, among 255,602 features about 3,300 features have frequency greater than 300, about 8,500 greater than 100 and about 67,000 greater than 10.

5.2.3 Domain examples

For each node (leaf or non-leaf) in the hierarchical structure a separate classifier is induced. The idea is that each classifier can distinguish between documents that should be assigned the category it represents and other documents. These other documents could in general be some random Web documents. Starting from that, we define a set of negative examples as examples from the whole domain hierarchy. There are also some other possibilities to choose negative examples. For instance, use documents randomly chosen from the Web. We use the domain hierarchy instead to get more focused classifiers. Since the proportion of positive documents is for most categories very small compared to the set of all domain documents, our set of negative documents is formed from all domain documents. This contributes to the efficiency and seems not to harm our experimental results.

The set of positive examples is constructed for each classifier separately from the examples in the corresponding category. All training examples (positive and negative) are constructed from the information included in the Yahoo documents only. We define category examples as items appearing on the Yahoo document representing the category and all the examples in its subcategories. Our assumption here is that an item on Yahoo document contains words representative for actual Web document it points to. In this way we reduce the time and space needed to collect and store training data. The actual Web documents are used as testing examples selected randomly from all the Web documents accessible from the hierarchy domain. For example, the Yahoo category *'Science: Computer Science: Artificial Intelligence'* is represented with the Yahoo Web document in Figure 5.4. For this category positive examples are constructed from items containing hy-

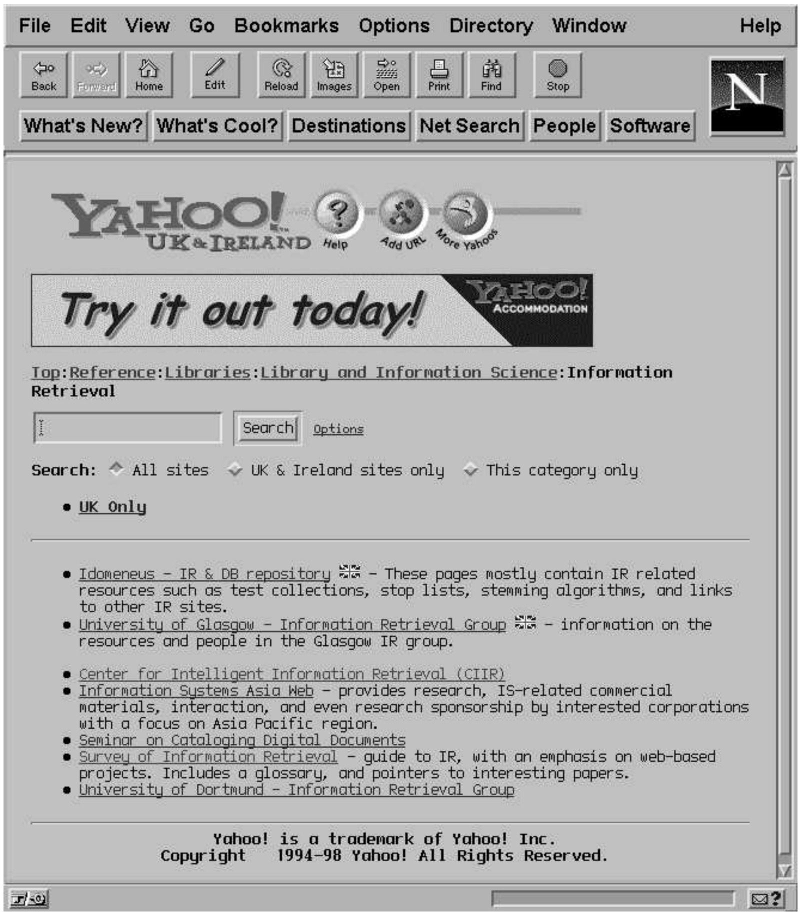

Figure 5.3: Yahoo Web document representing category *'Reference:Libraries:Library and Information Science:Information Retrieval'* used in illustration of document representation in Table 5.1.

perlinks to the Yahoo documents representing subcategories ranging from *'Artificial Life'* to *'Web-based'*, items containing hyperlinks to the actual Web documents (*'Enterprise Project, The'*, *'VISION project'*, ...) and all the examples contained in its subcategories.

In the Yahoo hierarchy some categories include parts of the other categories (additional connections in the tree structure pointing to the subtrees originally included in some other parts of the tree). These tree nodes that have more incoming connections are copied as shows example for *'Computer Science'* in Figure 5.6 that represents our transformation of the part of Yahoo hierarchy in Figure 5.5. When generating training examples we imagine that there are as many copies of a subtree as there are additional hyperlinks pointing to it.

5.2.4 Weight examples

Positive examples for each category are formed from the examples in the category node and the examples in the nodes below it representing its subcategories. These nodes and consequently their examples form a hierarchical structure that we map to a single set of examples. Different mappings are possible. A simple one is to ignore the hierarchical

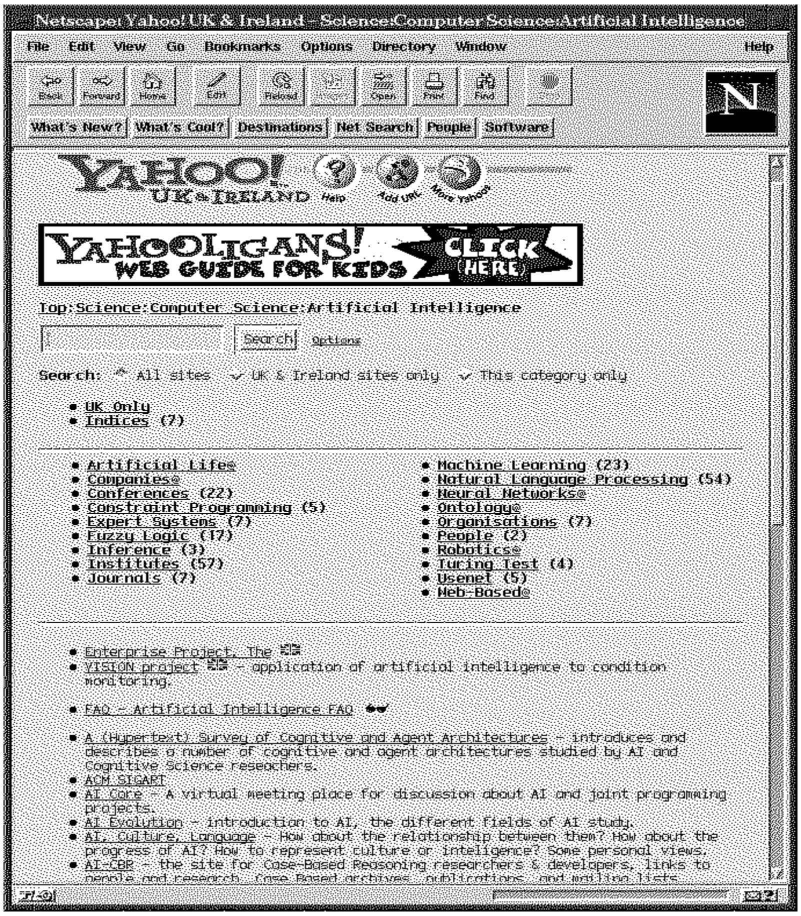

Figure 5.4: Yahoo Web document representing category *'Science: Computer Science: Artificial Intelligence'* contains 3 keywords in its name. The category is represented as a tree node that contains pointers to the subtrees ranging from *'Artificial Life'* to *'Web-based'* and items pointing to the actual Web documents.

structure of the examples and form a set of positive examples as a union of all the examples that appear in the category node or anywhere in its subtrees. The potential problem we see with this approach is that large subtrees would completely cover smaller subtrees, even if those are in the higher levels of the hierarchy. Ignoring hierarchical structure of examples means that we are ignoring information included by human constructing the hierarchy, that reflects in the more general and important categories being higher in the hierarchy. For example, one of the top categories *'Computers and Internet'* among others includes subtree entitled *'Year 2000 Problem'* having only 100 examples. This subtree is small compared to some other subtrees included at the same hierarchy level, like for instance, *'Computer Science'* having 1,600 examples (see Figure 5.5). Despite the size *'Year 2000 Problem'* was obviously considered important enough to appear immediately at the second level of the whole Yahoo hierarchy. More detailed illustration of this problem is given on artificial data represented with the tree in Figure 5.7. To take into account the hierarchical structure of examples, we assign weight to each subtree and use it for weight examples in the subtree. The examples are propagated to the higher hierarchy

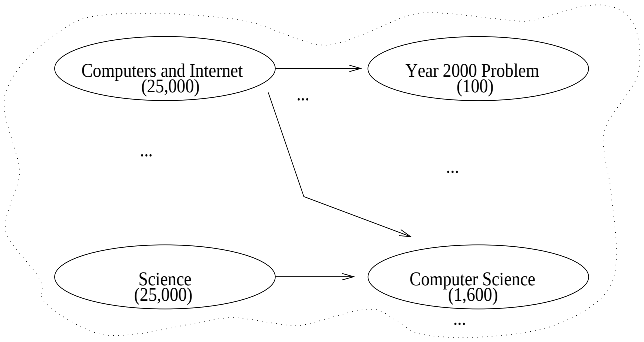

Figure 5.5: Part of the Yahoo hierarchy showing that some smaller subtrees like *'Year 2000 Problem'* can appear at the same hierarchy level with more than 10 times bigger subtrees like *'Computer Science'*. Their examples should be combined together carefully in order to prevent that examples from the bigger subtree 'cover' examples from the smaller one.

levels trough their probability distribution over features they contain since, probability distribution over features is sufficient for constructing the naive Bayesian. The probability of word sequence W in the category represented with tree T rooted at node N that has k subtrees is estimated in the root of the tree from the weighted examples. If we weight examples by the probability of a subtree or node in the tree T, the probability of each word sequence can be estimated using the following recursive formula:

$$P(W|T) = \sum_{i=1}^{k} P(W|SubT_i) \times P(SubT_i|T) + P(W|N) \times P(N|T)$$

Where $\sum_{i=1}^{k} P(SubT_i|T) + P(N|T) = 1$. This formula is used recursively starting from the hierarchy subtree rooted at the node that represents the category under consideration. The recursion stops in the leaves. According to the above formula the probability of each word sequence contained in a leaf Yahoo Web document is calculated as $P(W|Leaf)$. Since the leaf node has no subtrees, $\sum_{i=1}^{k} P(SubT_i|Leaf) = 0$. Probability of word sequence W in node N is estimated using Laplace probability estimate from the frequency of word sequence $TF(W)$ in the training examples generated for node N as follows $P(W|N) = \frac{1+TF(W|N)}{|V|+\sum_i TF(W_i|N)}$, where $|V|$ is the vocabulary size and here equals to the number of all different word sequences used in the document representation.

For instance, take the category named `'Reference:Libraries:Library and Information Science:Information Retrieval'` represented with the text on Yahoo Web document given in the illustration of document representation in Section 5.2.2. The probability distribution for this category $P(W|'pos')$ is given in Table 5.2.

There are different weight functions that can be used when examples are propagated to the higher hierarchy level. Probability of a subtree $P(SubT_i|T)$ approximated by the number of included examples (alternatively the number of included features or included categories) is a possible weight function. Using a logarithm to approximate a subtree size is the other possibility that we expected to give better results, since it reduces the influence of the big difference in the number of examples contained at the same-level subtrees. We have experimentally compared seven weight functions each based on one of the following:

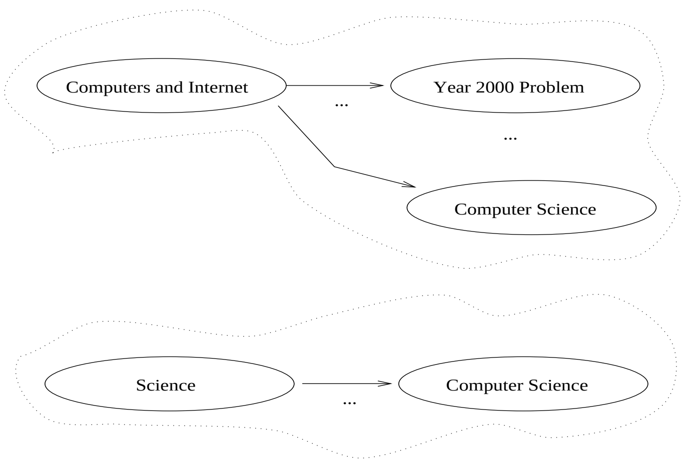

Figure 5.6: Part of two subproblems defined on the Yahoo hierarchy showing how we handle multiple incoming connections when splitting problem into subproblems and defining training examples for each category node. Figure 5.5 shows the original structure that is the tree with the two incoming connections to *'Computer Science'*.

the number of examples, the number of features, the number of categories, logarithm of each of the previous three and a constant weight that means ignoring the hierarchical structure. Experimental results are given in Section 5.4. Since no significant difference in the experimental results on randomly selected testing documents was observed, the weight function we are using here is the normalized logarithm of the number of examples included in the subtree as one of the three functions based on the logarithm that according to our intuition (see the illustration belove and Figure 5.7) are expected to be the most appropriate.

$$Weight(SubT_j) = \frac{\ln(1 + Exs(SubT_j))}{\sum_{i=1}^{k} \ln(1 + Exs(SubT_i)) + \ln(1 + Exs(Node))} \tag{5.1}$$

$$Weight(Node) = \frac{\ln(1 + Exs(Node))}{\sum_{i=1}^{k} \ln(1 + Exs(Subtree_i)) + \ln(1 + Exs(Node))} \tag{5.2}$$

For illustration of potential problems when propagating examples to the higher hierarchy levels consider probability of a single word w_1 at the top of an artificial example subtree given in Figure 5.7. Notice that *Node-4* and *Node-2* have the same number of examples, while *Node-1* is much bigger. The weight of subtree should help to avoid almost complete 'coverage' of *Node-4* by much bigger subtree rooted at *Node-3*. It should also take into account that *Node-4* is on the higher level of hierarchy and thus more important in *Subtree-5* than *Node-2*, even though the two are of the same size containing 10 examples each. Defining subtree weight as the probability of the subtree approximated from the number of contained examples results with the following probability of word w_1 in *Subtree-5*:

$$\begin{aligned}
P(w_1|Subtree_5) = {} & \tfrac{10}{10010} \times P(w_1|Subtree_4) + \\
& \tfrac{10000}{10010} \times P(w_1|Subtree_3) + 0 \times P(w_1|Node_5) \\
= {} & 0.001 \times P(w_1|Node_4) + 0.999 \times [\tfrac{9985}{10000} \times P(w_1|Node_1) + \\
& \tfrac{10}{10000} P(w_1|Node_2) + \tfrac{5}{10000} P(w_1|Node_3)] \\
= {} & 0.001 \times P(w_1|Node_4) + 0.998 \times P(w_1|Node_1) + \\
& 0.001 \times P(w_1|Node_2) + 0.0005 \times P(w_1|Node_3)
\end{aligned}$$

word sequence represented by a feature	conditional probability in positive examples
INFORMATION 0.119	INFORMATION RESOURCES 0.015
INFORMATION RETRIEVAL 0.074	CENTRE 0.015
IR 0.074	SITES 0.015
RETRIEVAL 0.074	WEB 0.015
GROUP 0.045	LINKS 0.015
UNIVERSITY 0.045	REGION 0.015
ASIA 0.03	COLLECTIONS 0.015
RESEARCH 0.03	DOCUMENTS 0.015
GLASGOW 0.03	LIBRARY 0.015
RESOURCES 0.03	PEOPLE 0.015
LIBRARY INFORMATION SCIENCE 0.015	PAGES 0.015
CATALOGING 0.015	COMMERCIAL 0.015
PACIFIC 0.015	SCIENCE 0.015
DIGITAL 0.015	REFERENCE 0.015
MATERIALS 0.015	GLOSSARY 0.015
RELATED 0.015	INCLUDES 0.015
SYSTEMS 0.015	GUIDE 0.015
EMPHASIS 0.015	UK 0.015
INTERESTING 0.015	LISTS 0.015
MASSACHUSETTS 0.015	LIBRARIES 0.015

Table 5.2: Conditional probability distribution of features given the positive class value.

The resulting probability does not meet our expectations described earlier. Probability distribution from *Node-4* has the same influence as the probability distribution from *Node-2* and this is very small (0.001) compared to the influence of *Node-1* (0.998). Since *Node-4* is higher in the hierarchy, we would like that its examples have grater influence in *Subtree-5* than the examples from *Node-2*. Using the weight defined in Equations (5.1, 5.2) results with the following calculation that is consistent with our expectations:

$$\begin{aligned} P(w_1|Subtree_5) = {} & 0.26 \times P(w_1|Subtree_4) + 0.99 \times P(w_1|Subtree_3) + \\ & 0 \times P(w_1|Node_5) \\ = {} & 0.26 \times P(w_1|Node_4) + 0.99 \times [0.687 \times P(w_1|Node_1) + \\ & 0.179 \times P(w_1|Node_2) + 0.134 \times P(w_1|Node_3)] \\ = {} & 0.26 \times P(w_1|Node_4) + 0.68 \times P(w_1|Node_1) + \\ & 0.177 \times P(w_1|Node_2) + 0.133 \times P(w_1|Node_3) \end{aligned}$$

In the resulting probability distribution influence of *Node-1* distribution (0.68) is still higher that the influence of the *Node-4* (0.26), but not as much higher as in the case when weight of subtree is defined as subtree probability (0.998 and 0.001). The influence of distribution in *Node-4* (0.26) that is higher in hierarchy than *Node-2* is higher than the influence of *Node-2* (0.177) even though these two nodes have the same number of example. This definition of node and subtree weight meets our expectations.

5.2.5 Learning on text data

A separate classifier for each of the defined subproblems is induced using the naive Bayesian classifier on Tf-representation of text data as described in Section 3.2.1. The final result of learning is a set of specialized classifiers. We use these classifiers as background knowledge for learning to predict clicked hyperlink as described in Section 3.1.2.

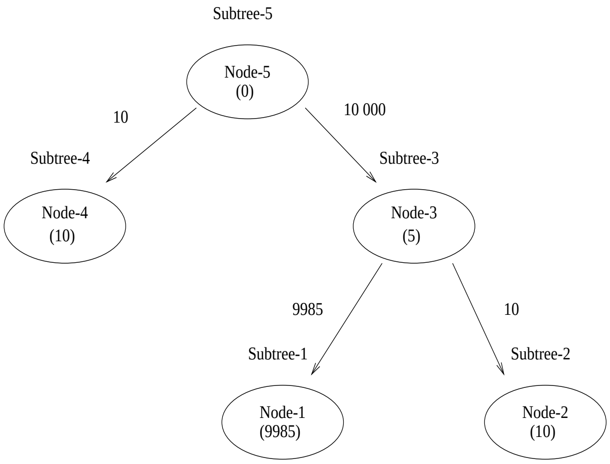

Figure 5.7: Artificial hierarchy where number of examples given in each node $Exs(Node)$ equals the number of hyperlinks to actual Web documents contained in the node ignoring its subtrees (eg. *Node-3* contains 5 examples), while the number of examples given on the edges between nodes $Exs(Subtree_i)$ is the summary number of examples in the whole subtree (eg. *Subtree-3* contains 9,985+10+5=10,000 examples).

Our document representation is extended and includes features representing not only single words but also word sequences that are handled in feature selection and learning the same way as the single words. The formula used to predict probability of class value C for a given document Doc is given in Equation (3.2).

feature	frequency	feature	frequency	feature	frequency
ASIA	3	SYSTEMS	2	NATIONAL	1
INFORMATION	2	PAGE	1	RESOURCES	1
RESEARCH	2	CONFERENCES	1	UNIVERSITY	1
WEB	2	WWW	1	SINGAPORE	1
SITES	2	DIVISION	1	TOPICS	1

Table 5.3: Testing example represented with the included features and their frequencies.

For instance, take the testing example represented with the features given in Table 5.3. Using Odds ratio as the feature scoring measure, vector size 1, pruning level 3 and taking all the predictions that have predicted probability over 0.9 (see Section 5.3 for parameter explanation), this example originally belonging to the category named `'Reference:Libraries:Library and Information Science:Information Retrieval'` and characterized by four keywords: *Reference, Libraries, Library and Information Science, Information Retrieval* is classified to the categories given in Table 5.4. Out from the classification results, the following keywords are predicted for this testing example: *Reference, Libraries, Library and Information Science, Information Retrieval, Indices, Institutes* giving Recall 1 (all the correct keywords are predicted) and Precision 0.67 (4 out of the 6 predicted keywords are correct).

category name	predicted probability
`'Reference:Libraries:Library and Information Science:Information Retrieval'`	**0.999932**
`'Reference:Libraries'`	0.998682
`'Reference:Libraries:Indices'`	0.952278
`'Reference:Libraries:Library and Information Science:Institutes'`	0.938801
`'Reference:Libraries:Library and Information Science'`	0.932688

Table 5.4: Classification results on the testing example originally belonging to category `'Reference:Libraries:Library and Information Science:Information Retrieval'`.

5.3 Experimental results

Similar as in experiments from Section 4.3, we observe the influence of the number of features (vector size) on independent set of testing examples selected randomly from the actual Web documents accessible from the corresponding hierarchy domain. The reported results are averaged over 5 runs using hold-out method with non-overlapping testing sets. The actual Web documents these items point to are used as the testing documents.

We report the average F_2-measure per document calculated from Precision and Recall (see Section 2.3.2) on the problem of category prediction and the problem of keyword assignment. Precision and Recall for each testing example is calculated as described in Section 4.3. Additionally, we report rank and probability assigned to the correct category as described in Section 4.3.

Results of the naive Bayesian classifier using Odds ratio for feature subset selection (as suggested in [78] and Chapter 4) are given in Table 5.5. Domains used in our experiments represent three out of the fourteen top level Yahoo categories: *'Computers and Internet'*, *'Education'*, *'References'*. Section 2.2 gives domain characteristics including Table 2.2 with information about the number of nodes, the number of features showing also how many features represent different length word sequences, the number of examples (documents), the average distance between nodes measured by the number of connections between any two nodes in the hierarchy and the average number of features in positive documents over different categories.

5.3.1 Pruning unpromising categories

The pruning we use here is based on the observation that for a fixed testing example, many classifiers return very low probability that the example is a member of their category. Based on the characteristics of the naive Bayesian classifier and the included information we prune unpromising categories. Each example is represented here as a feature-vector and each category classifier is represented with probability distribution over features. In the classification, category contributes only the probabilities of features that are included in the testing example. From the given naive Bayesian formula it can be seen that additionally to the prior probability of class value, only the features that are common to the example and the category under consideration contribute to the classification result. The goal of classification using this set of specialized classifiers is to find categories that example potentially belongs to. We reduce the number of categories to be considered in the classification by using inverted index. Namely, we assign to each feature a list of categories it appears in with probability > 0.

Domain name	Median of the corr. category rank	Median of the corr. category prob.	Average F2-measure on key-words	Average F2-measure on cate-gories	Average considered categories	Pruning level Common features	Pruning level Score threshold
Comp.	12.4	0.99	0.51	0.32	776 ± 3.5	1	0.70
	12.2	0.99	0.54	0.34	301 ± 3.5	3	0.70
	11.9	0.98	**0.59**	0.33	**138**± 2.1	**6**	**0.70**
	12.4	0.99	0.51	0.32	1255 ± 4.6	1	0.85
	12.8	0.99	0.51	0.33	647 ± 5.2	3	0.85
	12.7	0.99	0.55	0.34	344 ± 4.0	6	0.85
min.prun.	12.5	0.99	0.51	0.32	2148 ± 9.0	1	1.0
Edu.	**8.6**	0.99	0.44	0.42	143 ± 2.2	1	0.70
	9.4	0.99	0.48	0.43	**64**± 1.4	**3**	**0.70**
	10.9	0.94	**0.55**	**0.44**	35 ± 1.1	6	0.70
	8.9	0.99	0.44	0.41	198 ± 2.6	1	0.85
	9.0	0.99	0.45	0.42	**114**± 2.0	**3**	**0.85**
	10.0	0.99	0.49	**0.44**	67 ± 1.8	6	0.85
	10.4	0.97	**0.54**	**0.45**	46 ± 1.3	12	0.85
min.prun.	**8.9**	0.99	0.44	0.41	303 ± 2.1	1	1.0
Ref.	5.3	0.99	0.52	0.37	55 ± 0.6	1	0.70
	4.4	0.99	0.57	0.41	26 ± 0.6	3	0.70
	3.6	0.99	**0.64**	**0.42**	**15**± 0.6	**6**	**0.70**
	3.2	0.98	**0.67**	0.40	**11**± 0.6	**9**	**0.70**
	3.6	0.97	**0.68**	0.39	**9**± 0.6	**12**	**0.70**
	5.2	0.99	0.51	0.37	72 ± 0.8	1	0.85
	5.2	0.99	0.54	0.38	41 ± 0.6	3	0.85
	4.4	0.99	0.59	**0.42**	24 ± 0.5	6	0.85
	3.8	0.99	0.62	**0.44**	**16**± 0.5	**9**	**0.85**
	3.6	0.98	**0.65**	**0.43**	**13**± 0.8	**12**	**0.85**
min.prun.	5.2	0.99	0.51	0.37	108 ± 0.8	1	1.0

Table 5.5: Best result using pruning and without pruning on five domains formed from the Yahoo text hierarchy. We observe performance on two problems: document keyword assignment (F_2-measure) and document category prediction (correct category rank and probability, F_2-measure). The given average number of considered (and its standard error) categories reflects the speed up in classification. Pruning level is given by two parameters: the required number of features common to a classifier and a testing example and the score threshold for selecting the highly scored features. Reported results are for vector size 1 meaning that the number of selected features is the same as the number of features that occur in positive examples. By minimal pruning (min.prun.) we mean here requiring one common feature, since our classifier in case of 0 common features (no pruning) returns the majority class value ('negative'). Recall that zero common features means that testing document and the current category share no words.

When classifying testing example only classifiers that share a required number of highly scored features with the example enter the classification process. More specifically, a classifier is required to assign a non-zero conditional probability given the positive class value $P(W|pos) > 0$, to the required number of highly scored features that are contained in the testing example. Namely, this conditional probability is used by the naive Bayesian classifier (see Formula 3.2) when calculating probability that the testing example is a member of the corresponding category and that it should be assigned the corresponding set of keywords. In case that non of the features contained in the testing examples is assigned a non-zero probability by a classifier, a prior probability is returned (in our case for most classifiers the prior probability of positive class value is << 0.1). To select highly scored features, we sort features according to the assigned score and take so many best features as needed that their summary score represents ScoreTsh*100% of the summary score assigned to all the selected features. We vary the number of required common features from 1, giving minimal pruning, to 12, meaning that at least 12 highly score features from the subproblem domain have to occur in the testing document. We also vary the threshold on the summary score $ScoreTsh$ from 1 (consider all the selected features) to 0.4 (consider as many best features as needed that their summary score represents 40% of the summary score assigned to all the selected features).

In our experiments we observe influence of the pruning level (score threshold and number of common features) and the number of selected features (vector size) to the categorization results over independent set of (500 for *'Computers and Internet'* and 300 for *'Education'* and *Reference'*) testing examples selected randomly from the actual Web documents accessible from the corresponding hierarchy domain. The reported are results of testing where we removed 500 (300) items (a brief document description including hyperlink) from the Yahoo Web documents out of the training data. The actual Web documents, these items point to, are used as the testing documents.

Results of the naive Bayesian classifier are given on domains representing three of the top fourteen Yahoo categories (see Table 2.2 for the data description). Table 5.5 gives the best results using proposed pruning mechanism and the best results using no pruning. It can be seen that the best results are achieved when only a small number of features is used (vector size 1), the number of required common features is 3-6 and the summary score threshold is 0.7-0.85. To get an idea about the actual number of used features, vector size 1 means in average over categories: on *'Entertainment'* use 58 best features (0.2% of all domain features), on *'Arts and Humanities'* use 65 best features (0.7%), on *'Computers and Internet'* use 42 best features (0.62%), on *'Education'* use 85 best features (2.7%), on *'Reference'* use 49 best features (5.3%).

The best value of F_2-measure is achieved on all tested domains when pruning is used. Lower value achieved in the case of no pruning is mainly due to the lower value of Precision caused by assigning high probability to unreliable categories (based on only few features shared between category and testing document). The best value of F_2-measure on keyword assignment is around 0.61, meaning typically Recall of 0.79 and Precision of 0.47, ie., 79% of the correct keywords are correctly predicted while 47% of the all predicted keywords are correct. The best value of F_2-measure on category prediction is around 0.42, meaning typically Recall of 0.46 and Precision of 0.48.

When any of the best pruning settings is used a median of the correct category rank improves or remains about the same. For instance, on *'References'* the median of the correct category rank is in average 3.6, meaning that for the half of the testing examples the correct category is assigned rank up to 4. In case of no pruning, rank up to 6 is assigned instead. Median of the predicted probability remains the same on all tested domains if any of the best pruning setting is used.

From the efficiency point of view, the most important observation is improvement in the number of considered categories that drops to 5%-12% of all domain categories. We can observe that trend is to have smaller percentage of the considered categories on larger domains using the same pruning setting (0.7, 6) (eg., 5% on *'Computers and Internet'*

(138 out of 2652 categories considered), 10% on *'Education'* (35 out of 349 categories) and 12% on *'Reference'* (15 out of 129 categories)).

The best pruning settings are selected from the results on vector size 0.01 to 20 and pruning setting 0.4-1.0, 1-12. When selecting the best settings we considered median with lower and upper quartiles for the correct category rank and probability. For illustration we show in Table 5.6 results of one out of five runs on domain *'Computers and Internet'* for different vector sizes. For each vector size, the best results are given. When the number of selected features is higher than the number of features in positive examples (vector size >1), overall performance drops, except for the median of the correct category rank that improves. The drop in performance is especially evident from the probability predicted for the correct category, that drops from 0.99 to 0.12 (see Table 5.6). Given results are for the best pruning settings. For other pruning levels, the upper quartile of the correct category rank grows rather fast. For instance, for vector size 3, pruning with 9 to 12 common features and 0.85 score threshold results with the upper quartile for rank of 1330 (instead of 181 as given in Table 5.6 that is obtained for the same vector size using the same score threshold of 0.85 but requiring only 6 common features).

Median of correct category						Pruning level		
Rank			Probability			Common	Score	Vector
(LQ)	median	(UQ)	(LQ)	median	(UQ)	features	threshold	size
(4)	1326	(1330)	(0)	0	(0.007)	1-12	0.4-1	0.01
(3)	15	(107)	(<0.001)	0.64	(>0.99)	1	0.85	0.5
(4)	**15-16**	(106-145)	(0.07-0.01)	**0.99**	(>0.99)	**1-3**	**0.70**	**1**
(4-3)	**15-16**	(109-175)	(0.1-0.05)	**0.99**	(>0.99)	**1-6**	**0.85**	**1**
(3)	**15-16**	(107-145)	(0.05-0.01)	**0.98-0.96**	(>0.99)	**1-3**	**0.70**	**1.5**
(3)	**15-16**	(113-175)	(0.06-0.01)	**0.99-0.96**	(>0.99)	**1-6**	**0.85**	**1.5**
(3)	**15**	(107-133)	(0.05-0.01)	**0.98-0.96**	(>0.99)	1-**3**	**0.70**	**2**
(3)	**15**	(112-174)	(0.05-<0.001)	**0.98-0.95**	(>0.99)	**1-6**	**0.85**	**2**
(3)	15	(107-145)	(0.02-<0.001)	0.97-0.94	(>0.99)	1-3	0.70	3
(3)	15-16	(121-181)	(0.03-<0.001)	0.97-0.92	(>0.99)	1-6	0.85	3
(3)	15	(106-143)	(0.01-<0.001)	0.94-0.90	(>0.99)	1-3	0.70	4
(3)	15-16	(117-175)	(0.02-<0.001)	0.94-0.90	(>0.99)	1-6	0.85	4
(3)	15	(109-142)	(0.01-<0.001)	0.92-0.88	(>0.99)	1-3	0.70	5
(3)	15	(139-177)	(0.01-<0.001)	0.92-0.88	(>0.99)	1-6	0.85	5
(3)	12	(143-156)	(0.01-<0.001)	0.87-0.74	(>0.99)	1-3	0.70	7.5
(3)	12-13	(156-175)	(0.02-<0.001)	0.87-0.74	(>0.99)	1-6	0.85	7.5
(2)	11	(179-158)	(<0.001)	0.66-0.48	(>0.99)	1-3	0.70	10
(2)	11-12	(231-195)	(<0.001)	0.66-0.48	(>0.99)	1-6	0.85	10
(2)	11	(217-209)	(<0.001)	0.30-0.17	(>0.99)	1-3	0.70	15
(2)	11	(357-264)	(<0.001)	0.30-0.17	(>0.99)	1-6	0.85	15
(2)	12-14	(315-232)	(<0.001)	0.12-0.05	(>0.99)	1-3	0.70	20
(2)	12-10	(451-317)	(<0.001)	0.12-0.05	(>0.99)	1-6	0.85	20

Table 5.6: Results of pruning on one run on domain *'Computers and Internet'*, showing influence of different number of selected features (vector size). We observe performance on the problem of document category prediction and report the median of the correct category rank and probability. Additional to the median value, we give lower (LQ) and upper (UQ) quartile, to make the decrease in performance with the growing vector size more evident. Pruning level is given by two parameters: the required number of features common to a classifier and a testing example and the score threshold for selecting the highly scored features.

Figure 5.8 shows influence of the vector size and required common features to the number of considered categories. As expected the number of considered categories drops with increasing pruning (lower score threshold and higher number of required features). It can be seen that score thresholds 0.7 and 0.85 reduce the number of considered categories to less than a half. Notice, that vector size below 1 and 6 common features are required has the similar influence as stronger pruning, since documents represented with fewer features usually have less common features with the category classifier. It is interesting that the number of considered categories remains the same for all vector sizes ≥ 1 (left graphs in Figure 5.8).

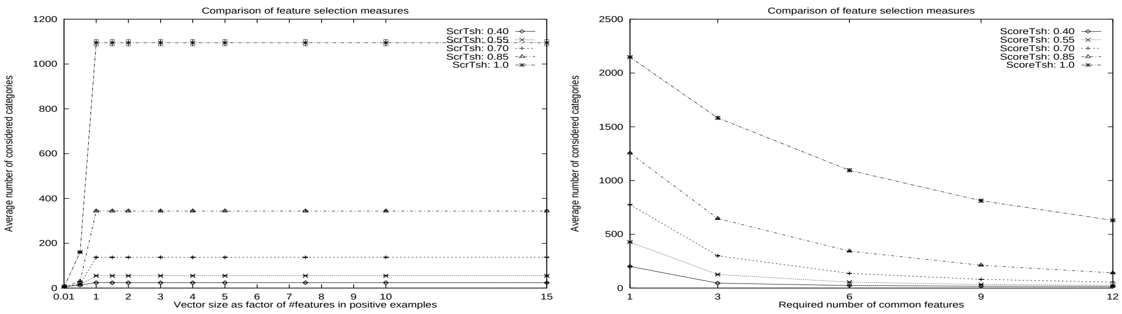

Figure 5.8: Comparison of different pruning thresholds when pruning unpromising categories on domain *'Computers and Internet'*. We give the average number of considered categories when changing vector size for the fixed number of required common features: 6 (left) and when changing the number of common features for the fixed vector size: 1 (right).

Figure 5.9 shows influence of the vector size and required common features to the correct category rank and probability. It can be seen that for vector size 1 and required 6 common features, all score thresholds achieve about the same rank, while probability is the best for score threshold ≥ 0.7 (upper graphs in Figure 5.9). Influence of the required number of common features (lower graphs in Figure 5.9) shows that requiring up to 6 features does not significantly influence the rank for all tested score thresholds. The same is true for the probability, but only for score threshold ≥ 0.7. From that we can conclude that requiring up to 6 common features and using score threshold ≥ 0.7 is the best pruning setting. Inside that, the combination (0.7, 6) gives the strongest pruning and thus the lowest number of considered categories. In most of our experiments we used this pruning setting.

Figure 5.10 shows value of F_2-measure on keyword (left graphs) and category prediction (right graphs). Influence of the vector size (upper graphs) shows that when requiring 6 common features, the highest F_2 values are achieved for vector size around 1 and for score threshold ≤ 0.7 on keyword assignment and ≥ 0.7 on category prediction. On keyword assignment, for vector size 1-5, stronger pruning (eg., score threshold 0.4 instead of 1.00) means for about 0.10 higher F_2 value. This indicates that the pruned categories were contributing bead keywords and thus lowering F_2 value by lowering precision. On category prediction score threshold < 0.7 results with a low probability predicted for the correct category thus giving also low F_2 values. Influence of the required number of common features (lower graph in Figure 5.10) shows that the highest F_2 is achieved for score threshold 0.55 and requiring ≥ 6 common features on keyword assignment. On the other

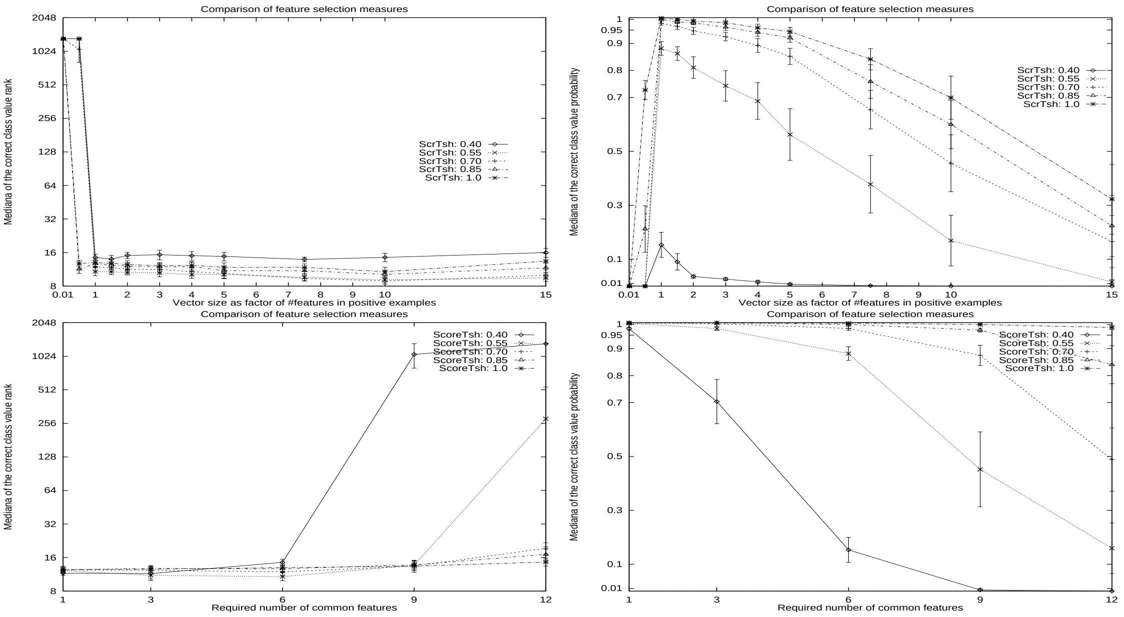

Figure 5.9: Comparison of different pruning thresholds when pruning unpromising categories on domain *'Computers and Internet'*. We show influence of the changing number of required common features for vector size 1 (upper graphs) and of changing vector size for 6 common features (lower graphs) to the median of the correct category rank (left) and probability (right).

hand, on category prediction, the highest F_2 is achieved for score threshold 0.85 requiring $\geq$ 6 common features. The best parameter setting for category prediction suggested by the value of F_2-measure is consistent with the best setting based on the correct category rank and probability (0.7-0.85, 6).

5.3.2 Probability threshold for Precision and Recall

In order to calculate Precision and Recall per document we need to set a threshold on the predicted probability in order to map probability prediction to classification. Here we give experimental comparison of different probability thresholds on three domains for the problem of keyword assignment and the problem of category prediction. We varied vector size from 0.01 (for most categories this means selecting only 1 feature) to 12 (ie., selecting 12 times the number of features in positive examples). Predicted probability threshold used to calculate Precision and Recall (see Section 4.3) was varied from 0.5 to 0.99. In Tables 5.7 and 5.8 we give results on three domains for vector size 1, as one of the best performing vector sizes, and one of the best performing pruning settings (0.7, 3). We don't report as the best results the values when Precision and Recall are equal, since in our opinion the problem domain is such that we care more about Recall than about Precision. Namely, predicted keywords that are not correct are often similar in meaning to the correct keywords, eg. *Roget_s_Tthesaurus* in the above example is similar to *Thesauri*. From Tables 5.7, 5.8 it can be seen that probability threshold 0.95 achieves good results on both problems (keyword assignment and category prediction).

Mean values with standard errors for F_2-measure for different vector sizes and differ-

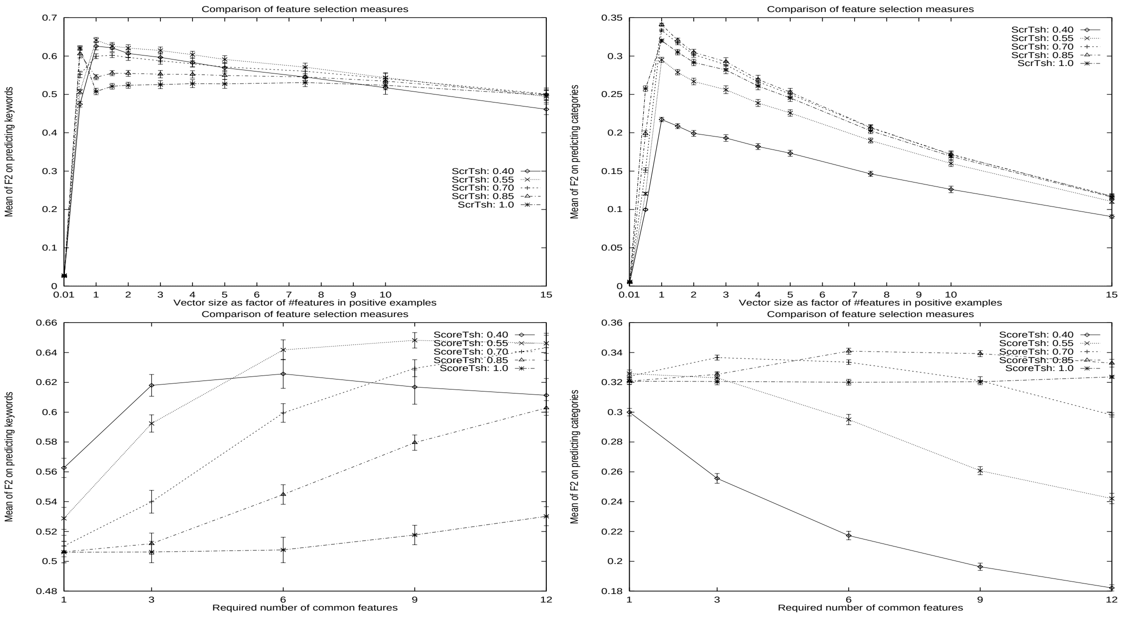

Figure 5.10: Comparison of F2-measure values on keyword assignment and document categorization when pruning unpromising categories on domain *'Computers and Internet'*. We show influence of the changing number of required common features for vector size 1 (upper graphs) and of changing vector size for 6 common features (lower graphs) to the mean value of F2-measure on the problem of keyword prediction (left) and on the problem of document categorization (right).

ent probability thresholds are given in Figure 5.11. It can be seen that the higher number of features (eg., vector size 5 instead of vector size 1), given the same score threshold results with lower value of F_2 on category prediction. This is consistent with the previous observations that the higher number of features (vector size > 1) means lower probability predicted for the correct category. For the same vector size higher probability threshold on keyword assignment results with a higher value of F_2 while on category prediction it results with a lower value of F_2. Actually on both problems higher value of probability threshold causes drop in Recall (less from the correct keywords and categories are recognized) and increase in Precision (more conservative prediction means less mistakes and lower number of 'false positive'). From Tables 5.7, 5.8 it can be seen that on document prediction in general Recall is much lower than on keyword assignment. Lower probability threshold on document prediction (Table 5.7) causes high increase in Recall. For instance, on *'Computers and Internet'* for score threshold 0.99 and 0.50 Recall is 0.35 and 0.65 respectively. The same change in the score threshold on keyword assignment (Table 5.8) results with a smaller difference in Recall (change from 0.82 to 0.92).

In future work we plan to apply the described approach to document categorization on the whole Yahoo hierarchy (50,000 categories and about 250,000 features). An interesting extension of the proposed document categorization is by considering the information about hierarchical structure of categories also in the model quality estimate. Currently, this information in considered when problem is divided into subproblems and in the generation of training examples. We expect that the classification of some non-typical testing examples (eg. page redirection) can be improved by expanding the example by documents

Domain name	Median of the corr. category Rank	Prob.	Average on category prediction F2	Precision	Recall	Probability threshold
Comp.	12.2 ± 1.90	0.99 ± 0.003	0.29 ± 0.002	0.35 ± 0.005	0.35 ± 0.007	0.99
			0.31 ± 0.002	0.33 ± 0.004	0.39 ± 0.007	0.98
			0.32 ± 0.002	0.31 ± 0.003	0.41 ± 0.007	0.97
			0.33 ± 0.001	0.31 ± 0.003	0.43 ± 0.007	0.96
			0.34 ± 0.001	**0.30** ± 0.004	**0.45** ± 0.007	**0.95**
			0.36 ± 0.001	0.28 ± 0.004	0.50 ± 0.007	0.90
			0.40 ± 0.002	0.24 ± 0.005	0.60 ± 0.004	0.70
			0.40 ± 0.004	0.21 ± 0.005	0.65 ± 0.003	0.50
Edu.	9.4 ± 0.98	0.99 ± 0.002	0.40 ± 0.007	0.33 ± 0.009	0.57 ± 0.010	0.99
			0.40 ± 0.007	0.31 ± 0.008	0.62 ± 0.010	0.98
			0.40 ± 0.007	0.30 ± 0.009	0.64 ± 0.010	0.97
			0.40 ± 0.006	0.29 ± 0.009	0.66 ± 0.010	0.96
			0.40 ± 0.005	**0.28** ± 0.008	**0.67** ± 0.010	**0.95**
			0.40 ± 0.005	0.26 ± 0.005	0.71 ± 0.010	0.90
			0.40 ± 0.004	0.21 ± 0.005	0.75 ± 0.010	0.70
			0.40 ± 0.004	0.19 ± 0.005	0.76 ± 0.010	0.50
Ref.	4.4 ± 0.25	0.99 ± 0.002	0.36 ± 0.010	0.37 ± 0.008	0.48 ± 0.010	0.99
			0.37 ± 0.010	0.35 ± 0.007	0.52 ± 0.010	0.98
			0.40 ± 0.010	0.33 ± 0.006	0.54 ± 0.010	0.97
			0.40 ± 0.008	0.33 ± 0.005	0.56 ± 0.010	0.96
			0.40 ± 0.007	**0.32** ± 0.005	**0.57** ± 0.010	**0.95**
			0.42 ± 0.006	0.31 ± 0.004	0.62 ± 0.010	0.90
			0.44 ± 0.004	0.27 ± 0.002	0.69 ± 0.010	0.70
			0.45 ± 0.004	0.26 ± 0.003	0.73 ± 0.010	0.50

Table 5.7: Results of experiments for different probability thresholds used to calculate Precision and Recall and consequently the value of F_2-measure. We observe performance on document category prediction. We give median of the correct category rank and probability, average values of F_2-measure, Precision and Recall. Pruning parameters are set to one of the best performing combinations: (0.7, 3) and vector size to 1. We also report the value of standard error, since all the reported results are averages over five runs on independent testing sets.

accessible from it in one step (using hyperlinks from the example document). These untypical examples can be automatically recognized by the system from the low probability assigned to all the categories.

Domain	Average on keyword assignment			Probability
name	F2	Precision	Recall	threshold
Comp.	0.57 ± 0.007	0.40 ± 0.008	0.82 ± 0.004	0.99
	0.56 ± 0.008	0.37 ± 0.006	0.84 ± 0.005	0.98
	0.55 ± 0.009	0.35 ± 0.006	0.85 ± 0.005	0.97
	0.54 ± 0.008	0.33 ± 0.005	0.86 ± 0.005	0.96
	0.54 ± 0.008	**0.32** ± 0.005	**0.86** ± 0.004	**0.95**
	0.52 ± 0.008	0.28 ± 0.005	0.88 ± 0.004	0.90
	0.48 ± 0.007	0.22 ± 0.004	0.91 ± 0.005	0.70
	0.45 ± 0.007	0.19 ± 0.005	0.92 ± 0.004	0.50
Edu.	0.50 ± 0.008	0.33 ± 0.010	0.81 ± 0.004	0.99
	0.49 ± 0.008	0.30 ± 0.009	0.83 ± 0.005	0.98
	0.49 ± 0.008	0.28 ± 0.009	0.84 ± 0.005	0.97
	0.48 ± 0.008	0.27 ± 0.009	0.85 ± 0.006	0.96
	0.48 ± 0.007	**0.26** ± 0.009	**0.85** ± 0.006	**0.95**
	0.47 ± 0.008	0.23 ± 0.005	0.87 ± 0.005	0.90
	0.43 ± 0.007	0.19 ± 0.004	0.90 ± 0.004	0.70
	0.41 ± 0.007	0.17 ± 0.004	0.91 ± 0.003	0.50
Ref.	0.56 ± 0.010	0.44 ± 0.007	0.78 ± 0.010	0.99
	0.56 ± 0.010	0.40 ± 0.010	0.81 ± 0.010	0.98
	0.57 ± 0.008	0.38 ± 0.009	0.83 ± 0.009	0.97
	0.57 ± 0.007	0.37 ± 0.009	0.84 ± 0.009	0.96
	0.57 ± 0.006	**0.36** ± 0.008	**0.85** ± 0.008	**0.95**
	0.57 ± 0.004	0.33 ± 0.005	0.89 ± 0.005	0.90
	0.56 ± 0.002	0.29 ± 0.003	0.92 ± 0.003	0.70
	0.54 ± 0.002	0.27 ± 0.002	0.94 ± 0.002	0.50

Table 5.8: Results of experiments for different probability thresholds used to calculate Precision and Recall and consequently the value of F_2-measure. We observe performance on document keyword assignment problem. We give average values of F_2-measure, Precision and Recall together with standard errors. Pruning parameters are set to one of the best performing combinations: (0.7, 3).

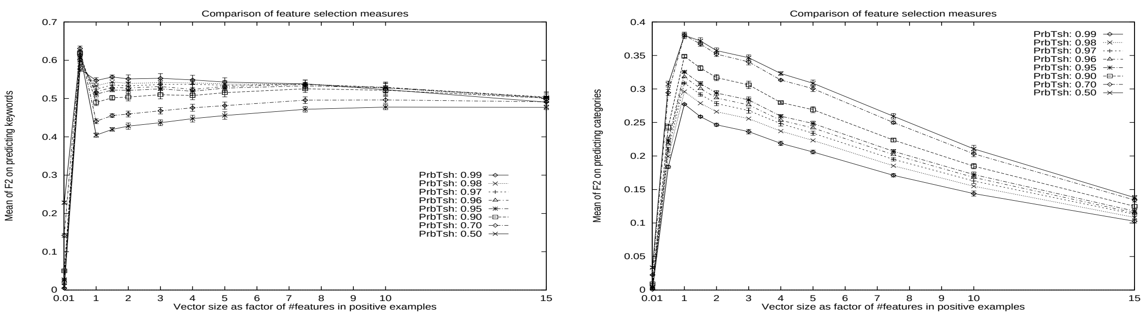

Figure 5.11: Comparison of different values of probability threshold for Recall and Precision on domain *'Computers and Internet'*. We give the mean value of F2-measure on keyword assignment (left graphs) and category prediction (right graphs).

5.4 Experimental comparison of weight functions

We experimentally compared seven example weight functions proposed in Section 5.2.4 on randomly selected testing examples. Experiments show that there is no significant difference between the functions (see Table 5.9). Our explanation is that in average on randomly selected examples the effect of bigger subtrees 'covering' smaller subtrees is not evident. Namely, when randomly selecting examples, most examples belong to the bigger subtrees (larger categories) and the misclassification of the examples belonging to the smaller subtrees has small influence to the overall result. This hypothesis can be verified by experiments on testing examples that would be selected mostly or only from the smaller subtrees. Even though in average all the weight functions seems to work well, we decided to use the function that we expect to behave well also in non-typical cases, as described in Section 5.2.4.

Domain name name name	Weight method	Median of the corr. category Rank	Median of the corr. category Probability	Average F2-measure on keywords	Average F2-measure on categories
Comp.	Const	12 ± 0.84	0.97 ± 0.007	0.61 ± 0.006	0.32 ± 0.01
	Examples	10.5 ± 0.50	0.97 ± 0.007	0.61 ± 0.006	0.28 ± 0.01
	log Examples	**12.2** ± 0.86	**0.97** ± 0.007	**0.61** ± 0.006	**0.33** ± 0.005
	Features	11.2 ± 0.74	0.97 ± 0.007	0.61 ± 0.006	0.31 ± 0.01
	log Features	10.8 ± 0.58	0.97 ± 0.007	0.61 ± 0.006	0.30 ± 0.01
	Categories	10.3 ± 0.66	0.97 ± 0.007	0.61 ± 0.006	0.27 ± 0.006
	log Categories	11.7 ± 1.90	0.97 ± 0.006	0.61 ± 0.006	0.29 ± 0.03
Edu.	Const	11.7 ± 2.37	0.91 ± 0.07	0.55 ± 0.005	0.42 ± 0.02
	Examples	12.1 ± 2.22	0.90 ± 0.07	0.56 ± 0.008	0.36 ± 0.02
	log Examples	**10.9** ± 1.65	**0.94** ± 0.04	**0.55** ± 0.006	**0.44** ± 0.08
	Features	11.9 ± 2.34	0.90 ± 0.07	0.56 ± 0.008	0.40 ± 0.03
	log Features	12.2 ± 2.18	0.90 ± 0.07	0.56 ± 0.008	0.38 ± 0.02
	Categories	12.1 ± 2.22	0.87 ± 0.07	0.56 ± 0.008	0.34 ± 0.005
	log Categories	11.5 ± 1.66	0.93 ± 0.03	0.56 ± 0.008	0.38 ± 0.04
Ref.	Const	4.0 ± 0.32	0.99 ± 0.007	0.63 ± 0.006	0.40 ± 0.02
	Examples	4.8 ± 0.37	0.95 ± 0.017	0.60 ± 0.010	0.37 ± 0.01
	log Examples	**3.8** ± 0.37	**0.99** ± 0.001	**0.64** ± 0.006	**0.42** ± 0.009
	Features	4.4 ± 0.25	0.97 ± 0.017	0.62 ± 0.010	0.40 ± 0.02
	log Features	4.4 ± 0.25	0.97 ± 0.015	0.60 ± 0.010	0.38 ± 0.02
	Categories	5.0 ± 0.45	0.92 ± 0.021	0.58 ± 0.009	0.36 ± 0.01
	log Categories	5.0 ± 0.45	0.93 ± 0.026	0.59 ± 0.020	0.41 ± 0.05

Table 5.9: Result of experiments for different example weight methods using category pruning parameters (0.7, 6), probability threshold 0.95, vector size 1 and Odds ratio as a feature selection measure. We observe performance on two problems: document keyword assignment (F_2-measure) and document category prediction (correct category rank and probability, F_2-measure). Additionally, we also report the value of standard error, since all the reported results are averages over five runs on independent testing sets.

Chapter 6

Predicting word occurrences

6.1 Problem description

Consider the problem of helping user browsing the Web as described in Section 2.1. When making a prediction on a new hyperlink, we can either use only the information presented to the user (a part of the text from the document that contains a hyperlink) or try to take advantage of a hyperlink being connected to other document (see Section 2.1). Instead of classifying a given hyperlink, we could first predict the content of the document that is pointed to by the hyperlink and then use such an abstract document to predict if the user is going to click on the hyperlink. Representing documents as word-vectors (using bag-of-words representation, see Figure 1.1) enables using machine learning techniques to predict the frequency of words in the vector. We simplify this problem by predicting word occurrence instead of frequency and reducing the number of words we want to predict. The number of words is reduced by removing words from the "stop-list" and removing infrequent words (see Section 3.1).

One of the main characteristics of this machine learning problem is having a vector of class attributes instead of only one class attribute, where each of the vector elements corresponds to the occurrence of a particular word in the document. Since a document gives a context for words occurring in it, the occurrence of one word is possibly dependent on other words in the document. Thus, we are dealing with the problem of mutually dependent class attributes.

6.2 Learning algorithm

We propose two algorithms for learning on mutually dependent class attributes. The first one is based on k-Nearest Neighbor algorithm described in Section 3.2. The second one is based on Regression tree induction [13, 44]. In our experiments on predicting word occurrences based on the hyperlink that points to the document we used the first algorithm.

6.2.1 An algorithm based on k-Nearest Neighbor

In order to apply k-Nearest Neighbor algorithm on mutually dependent class attributes we modified the algorithm to handle examples that consist of *HyperLink-Doc* pairs, where *HyperLink* includes the actual hyperlink from HTML-document and some of its context

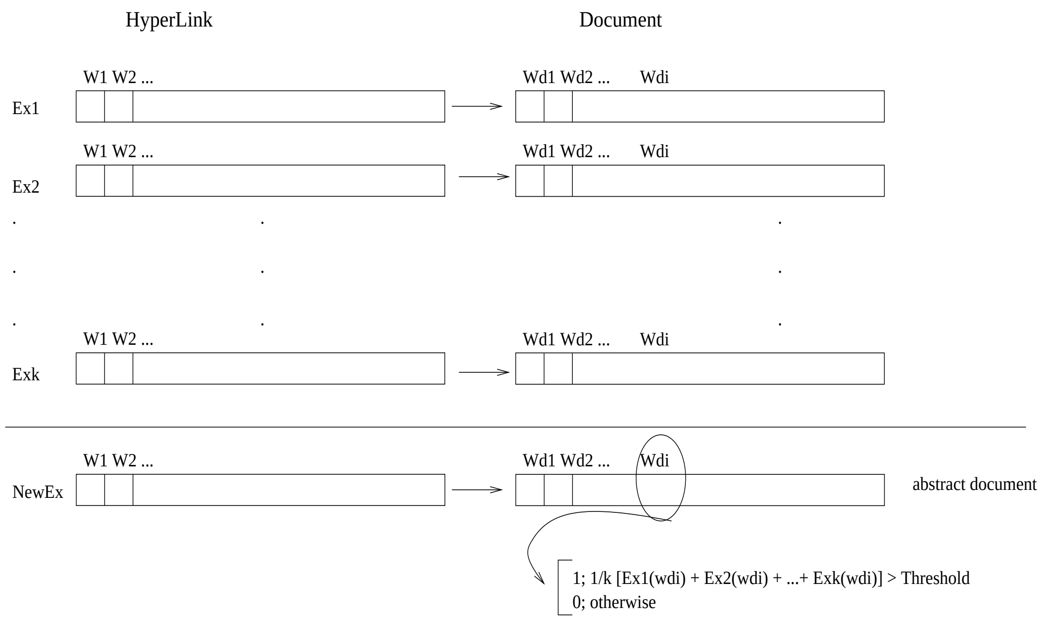

Figure 6.1: Illustration of document content prediction using the abstract document construction based on the hyperlink that points to the document. The approach is based on k-Nearest Neighbor algorithm. The k document pointed to by the k most similar hyperlinks are combined to get an abstract document.

(see Section 3.1) and Doc is a text document pointed to by the hyperlink. Wehn applied on the Yahoo hierarchy, each hyperlink description $HyperLink$ we build for training includes all the text in the hyperlink item included in the Yahoo document and the words from the document category name (written on the top of the Yahoo document that includes the hyperlink). The idea is that the algorithm finds a group of neighboring documents that are 'close' when the corresponding $HyperLinks$ are similar. $HyperLink$ and Doc are represented by two word-vectors. It is possible that multiple hyperlinks point to the same document. In this case, multiple copies of the document are included and no connection between them is made. The document is considered as neighboring if any of the hyperlink pointing to it is one of the nearest neighbors of the hyperlink under classification. If two hyperlinks H_1 and H_2 are neighboring, there can be the third hyperlink H_3 that is neighboring to only one of them, eg., H_1. Namely, the reason that two hyperlinks are neighboring are their common words. The common words of H_1 and H_2 can be completely different from the common words of H_1 and H_3, and there can be an empty set of the common words of H_2 and H_3.

The detected neighboring documents are then represented by an abstract document whose word occurrence is computed as an average of word occurrences of the neighboring documents. For each class (word occurrence) a threshold is set for mapping to a binary-valued class such that length of the abstract (predicted) document is as close as possible to the average length of neighboring documents from which it was generated. Figure 6.1 illustrates the construction of the abstract document. Mutual dependence of class attributes influences the value of Threshold set such that $Length(Abs.Doc) \doteq \frac{1}{k}\sum_i^k Length(Ex_i.Doc)$, where $Abs.Doc$ is the vector representing abstract document.

6.2.2 An algorithm based on Regression trees

Another possibility, we are not using here, instead of modifying the k-Nearest Neighbor algorithm, is to modify Regression tree induction algorithm. In Regression trees each example is described with attributes and a numeric class value. Recursive algorithm is used to build the tree using the top-down approach to split the set of examples into subsets of examples that have similar class value. At each level of recursion, a tree node is generated that splits the current set of examples into subsets that maximize split criterion $SplitCriterion(A) = Impurity(C) - Impurity(C|A)$. In split criterion used in in Regression tree algorithm [13, 44], an estimate of standard deviation of the class values is used as the measure of impurity of a node when searching for the best split of examples: $Impurity(C) = \frac{1}{n}\sum_{ex \in ExSet}(C(ex) - \mu(C|ExSet))^2$ and $Impurity(C|A) = \sum_j P(A_j) \times \frac{1}{n}\sum_{ex \in ExSet_j}(C(ex) - \mu(C|ExSet_j))^2$, where $P(A_j)$ is the probability of the j-th attribute value n is the number of examples in the training set $ExSet = \cup_j ExSet_j$, $ExSet_j$ is the subset of training examples having the j-th value of attribute A, $C(ex)$ is the actual class value of example ex, $\mu(C|ExSet)$ is the mean value of class C in the set of examples $ExSet$.

Recall that in our case of learning mutually dependent class attributes, each example consists of *HyperLink-Doc* pair, where *Doc* is a vector of class values to be predicted. The modification of the learning algorithm we propose is to change the split criterion used to generate each tree node. Instead of minimizing impurity of each node split $Impurity(C|A)$ that is used in a single-class regression tree, similarity of class vectors in each node should be maximized. For each subset of examples calculate the average vector of the class values $\vec{\mu}(\vec{C}|ExSet_j)$ that approximate the mean values $\mu(C|ExSet)$ of the class attributes. The proposed split criterion maximizes a cosine similarity of document vectors to an average document vector.

$$SplitCriterion(A) = \sum_j P(A_j) \times \frac{1}{n} \sum_{ex \in ExSet_j} similarity(\vec{C}(ex), \vec{Avg}(\vec{C}|ExSet_j))$$

Where $\vec{C}(ex)$ is the actual vector of class values of example ex, in our case it is a word vector *Doc* representing document. Dependence of class attributes can be taken into account when mapping predicted class attribute values to the binary values in the same way as used when k-Nearest Neighbor algorithm is applied.

6.3 Preliminary results

In our preliminary experiments we measure the performance of the induced model as the average Precision and Recall per document on the predicted document words. We observe the influence of the number of neighbors varied from 1 to 30 on independent set of randomly selected testing examples.

The procedure for getting domain data from the Yahoo domains given in Table 2.2 is the following. From all the examples included in the text hierarchy, 200 were randomly selected and used here when forming domain examples. In document categorization experiments on data from Table 2.2 (see Sections 4.3 and 5.3), each domain example contains a description of hyperlink and the class value that is the assigned topic category. Here each example consists of two parts: hyperlink description and document the hyperlink points to. The hyperlink part is similar as in the document categorization experiments and includes an item from the Yahoo document. Since the problem here is not categorization or keyword assignment but word occurrence prediction, examples here additionally include the corresponding category name that is in the input text hierarchy given as a set of keywords. The second part of each example is to be predicted and it is the actual document pointed to by the hyperlink. In the problem of document categorization these actual documents were used as testing examples for a model induced from the hyperlinks (see Chapter 5).

The hyperlink parts are represented by word sequences of length up to five (1-grams,..., 5-grams) as described in Section 3.1.1 and given in Table 2.2. The document parts are represented in similar way by word sequences of length up to two, where only the text included in the document parts is considered. Table 6.1 gives for each domain the number of features used as class attributes showing also the number of features representing different length word sequences (1-grams+2-grams). Notice that for all three domains that originally include different number of examples, 200 randomly selected examples are used here. This is also the reason why the number of class attributes (# features in documents in Table 6.1) is similar for all three domains.

The reported results are averaged over 3 runs using 3-fold cross validation method. Results of experiments on three domains generated from Yahoo hierarchy are given in Table 6.1. It can be seen that about 35% of document words are correctly recognized (Recall 0.35) while among all the predicted words only about 18% are correct (Precision 0.18). The best prediction is made when five to ten nearest neighbors are considered.

Domain name	# features in doc. (class attributes)	# of nearest neighbors	Precision of predicted words	Recall of predicted words	F_2
Ref.	2293	1	**0.271**	0.28	**0.256**
	(1919+376)	5	0.166	**0.368**	0.221
		10	0.148	0.354	0.215
		15	0.139	0.339	0.206
		20	0.132	0.327	0.197
		25	0.125	0.303	0.186
		30	0.121	0.291	0.181
Edu.	1920	1	0.151	0.136	0.131
	(1518+402)	5	**0.169**	**0.305**	**0.190**
		10	0.152	**0.307**	**0.199**
		15	0.141	0.285	**0.192**
		20	0.130	0.266	0.183
		25	0.124	0.264	0.177
		30	0.118	0.259	0.173
Comp.	2592	1	**0.19**	0.19	0.189
	(2187+405)	5	**0.15**	**0.34**	**0.206**
		10	0.138	**0.33**	**0.203**
		15	0.125	0.311	0.195
		20	0.118	0.298	0.179
		25	0.115	0.282	0.177
		30	0.11	0.285	0.174

Table 6.1: Average Precision, Recall and F_2-measure for three domains. The problem is predicting occurrence of words in the document based on the hyperlink that points to the document.

6.3.1 Discussion

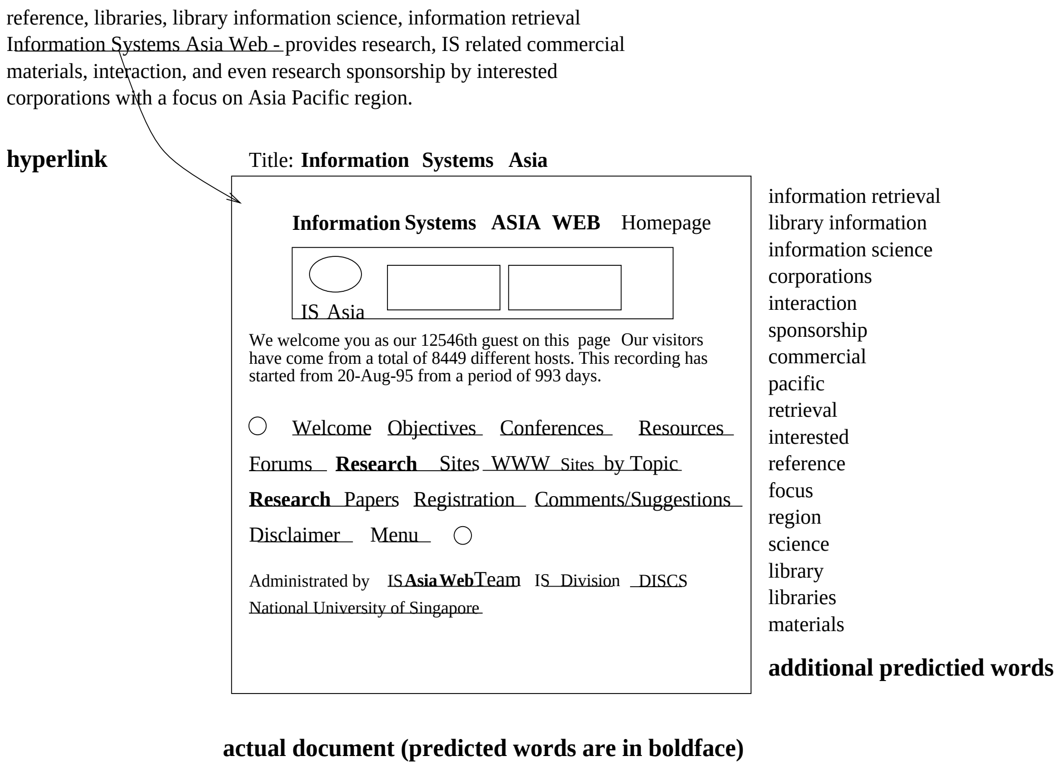

Figure 6.2: Example hyperlink-document pair and the set of word sequences predicted to occur in the document. The prediction is made from the hyperlink that points to the document. The predicted words are shown in two parts: correctly predicted words shown as boldface words on the document and the other predicted words shown in the list of words on the left side of the document.

Figure 6.2 shows an example pair of hyperlink-document and the words predicted by our system. The used hyperlink is given in the upper left corner in Figure 6.2. The same hyperlink was used as example for document categorization problem with the representation given in Table 5.3. The predicted words (word sequences) are shown as two sets of words. The first set contains the predicted words that also occur in the actual document (true positive). In Figure 6.2 these words are shown on the actual document in boldface. The second set contains other predicted words sorted according to their predicted probability (false positive). In Figure 6.2 these words are given in the right hand-side. It can be seen that among the other predicted words that don't occur in the document, the highest probability is assigned to the words that occur in the category name and describe the document content (eg., information retrieval, library information). Namely, the hyperlink description we build for training includes all the text in the hyperlink item and the words from the document category name.

From the example in Figure 6.2 and the results of experiments in Table 6.1 it can be seen that there are quite some predicted words that does not occur in the actual document. The number of predicted words is influenced by the threshold used for mapping the predicted probabilities into classifications. Currently this threshold is set so to get the number of predicted words approximately equal as the average number of words in the neighboring documents (document the nearest neighbor hyperlinks point to). Further experiments are needed to check if changing the threshold to predict smaller number of words will improve the system results.

Chapter 7

Conclusions and discussion

7.1 Summary and discussion

This dissertation proposes methods for using machine learning for analysis and classification of text data. The proposed solutions consider different domain specifics applicable on many similar problems of learning from text and the Web. One is unbalanced class distribution, where less than 1%-10% of examples belong to the target concept. The other is having asymmetric misclassification costs, given only implicitly in the problem. For instance, when predicting clicked hyperlinks, the default prediction is 'unclicked' and most of the hyperlinks should be assigned this prediction. If some of the 'clicked' hyperlinks are assigned false predictions and all the predictions are shown to the user (eg., highlighting hyperlinks predicted as clicked), these false negative 'get lost' in the mass of hyperlinks predicted unclicked (predicted negative). On the other hand, if some of the unclicked hyperlinks are predicted clicked (false positive), it joins the small set of hyperlinks predicted clicked and can be easily found by the user. These problems are considered during the feature subset selection and by model quality estimates used for the results evaluation. We use F_2-measure that is based on Recall and Precision, to estimate the model quality. This measure gives possibility for taking into account the asymmetric misclassification costs by changing the value of parameter (here set to 2) that regulates the trade-off between Recall and Precision. The measure focuses on the target class value and does not suffer because of the unbalanced class distribution. Maximizing F_2 instead of commonly used Classification accuracy influences the appropriateness of the feature scoring measures used for feature subset selection. We propose the usage of Odds ratio as a feature scoring measure for domains where the goal is to maximize performance on one (the target) class value. Additionally, our domains are characterized by a large number of examples and several tens or hundreds of thousands of features requiring efficient methods and careful implementation.

The induced target concept is appropriate for the classification of new documents with the special emphasis on the classification of shortened documents. As example of shortened documents we use the description of the hyperlinks from Web documents. Preliminary experiments on predicting word occurrences by learning mutually dependent class attributes has shown that in many cases this hyperlink description can be successfully used to predict part of the content of document the hyperlink points to. Using the proposed approach we successfully get about 35% to 45% of the document words (Recall is 0.35 to 0.45), while among all the predicted words about 20% are correct (Precision is 0.2). In this way, human knowledge and associations captured on the Web in the text of hyperlinks can be modeled and used for learning on hypertext.

The proposed enriched document representation extends the bag-of-words representation by adding word sequences and document topic categories. Word sequences containing up to five words are generated using a new efficient procedure that reduces the number of features during the feature generation. Topic categories are obtained from the model constructed using the new machine learning method for learning on class hierarchy. When learning from the class hierarchy we get the correct category among the twenty categories with the highest predicted probability out of the several thousands categories. When assigning keywords to a document based on the category prediction we get about 80% of the correct keywords (Recall is 0.80) while about 50% of all the predicted keywords are correct (Precision is 0.50). The models for category prediction and keyword assignment can be used as background knowledge giving higher level information about the document content. Experimental comparison of the document representation extended by including word sequences, with the commonly used bag-of-words representation has confirmed its usefulness. Including word sequences of up to 4 words improve results on category prediction (eg., on *'Computers and Internet'* the correct category rank is 12 instead of 15). We get also an increase in F_2-measure when predicting document category and when assigning keywords to the document. The developed elements of machine learning techniques for learning on class hierarchy can be used on some other hierarchies and extended to enable creation of new topic categories if sufficient support is obtained.

Experimental comparison and analysis of different feature scoring measures on text data has shown the importance of considering problem specifics and learning algorithm characteristics during the feature subset selection. Instead of incorporating them using the computationally intensive 'wrapper approach' to feature subset selection, we carefully select the feature scoring measure and use a rather simple feature subset selection approach. Our experiments show that in general the most important characteristics of a good feature scoring measure for text are favoring common features and not using feature absence if the used classification algorithm is not using feature absence. In the case of domains where one of the class values is the target class value, the most important characteristics of a good feature scoring measure is to discriminate the target class value from the other class values. Several extensions of the existing feature scoring measures are proposed that are appropriate for text domains with several tens or hundreds of thousands of features and can handle unbalanced class distribution and problems with asymmetric misclassification costs. The best performing measures are Odds ratio (and its variants) that discriminate the target class value from the other class values. Very good performance is also achieved by Cross entropy for text and Term frequency that both favor common features. The best results are achieved when only a small number of features is used. By this we mean using 50-100 best features or in other words using only 5%-15% of all features. Additionally to research motivation this work is also motivated by the potential applications of the developed methods. The following Section describes an example application developed here for performing initial experiments and data collection.

7.2 Example application of the proposed methods

Personal WebWatcher [77] project and the developed browsing assistant called PERSONAL WEBWATCHER provided motivation and early experiments for this thesis. The system was installed as a proxy server for Web browser by five CMU researchers. The data was collected from May 1996 to March 1997. Personal WebWatcher is a system (see Figure 7.1) that observes users of the WWW and suggests hyperlinks. The system is structured to specialize for a particular user. It records the addresses of pages requested by the user and highlights hyperlinks that it believes will be of interest. In the learning phase, requested pages are analyzed and a model of user interests is generated/updated. This learned model is used to highlight hyperlinks on the new HTML-pages requested by and presented to the user via Web browser that enables connection to the "proxy" server (eg., NETSCAPE browser).

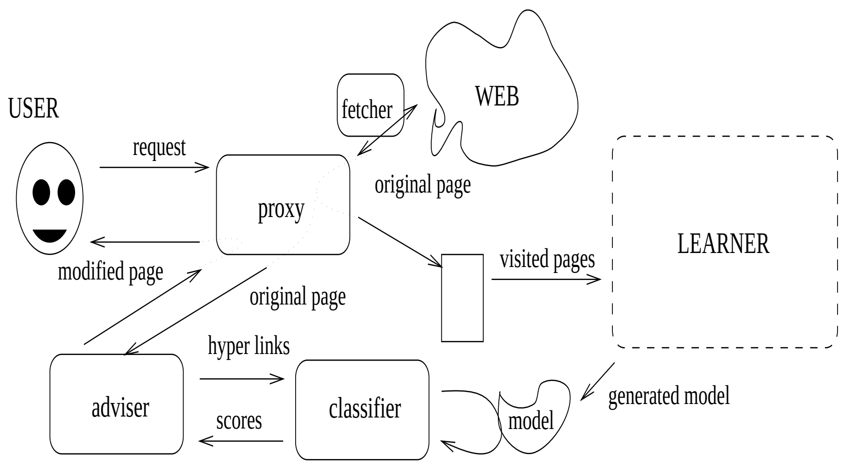

Figure 7.1: Structure of browsing assistant Personal WebWatcher.

Personal WebWatcher consists of two main parts: a **proxy server** that interacts with the user via Web browser and a **learner** that provides the user-model to the server (see Figure 7.1). Communication between them is via disk; the **proxy** saves addresses of visited documents (URLs) and the **learner** uses them to generate model of user interests. The first version of the system was implemented in approximately 2500 lines of Perl code and 1500 lines of C^{++} code. To make the system more time and space efficient we implemented a new version in C^{++} and integrated it with the machine learning system LEARNING MACHINE [33].

The **proxy server** consists of three main parts: **proxy** (additionally calls external **fetcher** code to fetch the page), **adviser** and **classifier**. **Proxy** waits in an infinite loop for a page request from browser. On request, it fetches the requested document and; if it is an HMTL-document adds advice and; forwards the document to the user. To add advice **proxy** forwards the page to **adviser**, that extracts hyperlinks from document and calls external code for **classification** that uses generated user-model. A limited number of hyperlinks that are scored above some threshold are recommended to the user, indicating their scores using graphical symbols placed around each advised hyperlink. For example, in Figure 7.2 three hyperlinks are suggested by PWW: "Machine Learning Information Services" and two project members (Dayne Freitag, Thorsten Joachims). There is a banner at the top of the page showing that PWW is “watching over the user’s shoulder”.

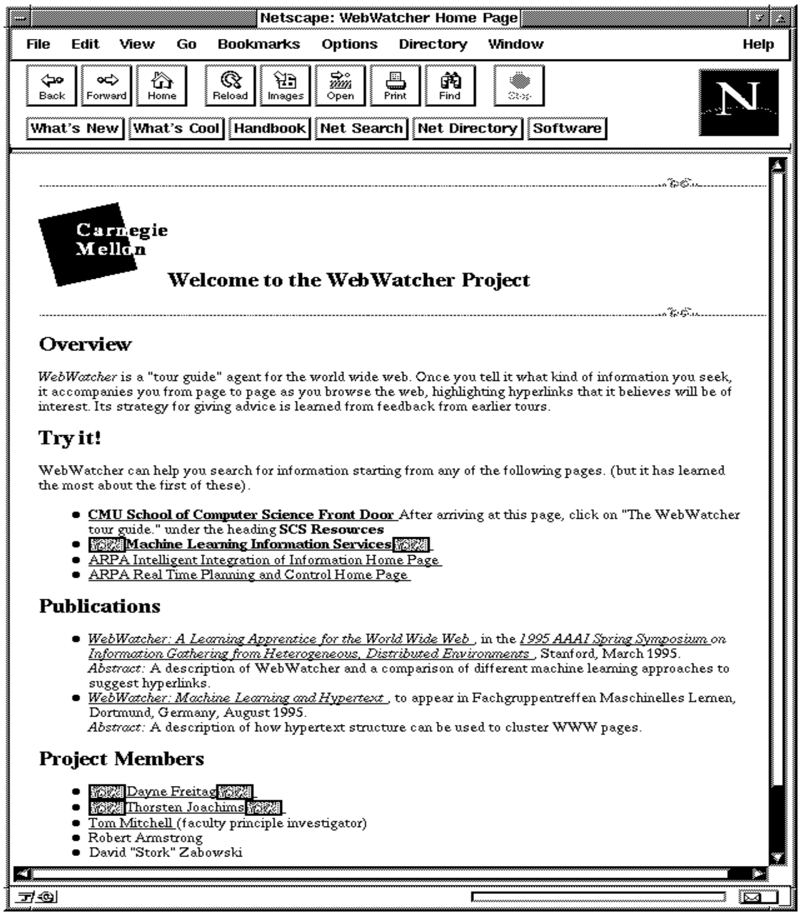

Figure 7.2: Example of HTML-page presented to the user by Personal WebWatcher.

7.3 Contributions

Here we give a brief summary of the contributions of the dissertation:

- Design of elements of machine learning techniques for learning on large sources of unstructured data (text) with sparse coverage of example space, such as HTML-documents on WWW. This includes:
    - The development of a machine learning method for learning from class hierarchies. The method handles a high number of class values, examples and features, by dividing a problem into subproblems based on the hierarchical structure of class values and examples, by applying feature subset selection and by pruning unpromising class values during the classification. Pruning unpromising class values during the classification is based on the minimal number of highly scored features shared by the subproblem classifier and the testing example. This pruning improves classification results while pruning 85%-95% of class values.

  - An extension of document representation that extends the bag-of-words representation by using word sequences generated in an efficient way. Using word sequences of length up to 2 instead of using only single words is more important for the classification results than using longer word sequences.
  - Comparison and analysis of different feature scoring measures for feature subset selection on text data with several tens of thousands of features and unbalanced class distribution; and several extensions of existing feature scoring measures which are appropriate for text domains with asymmetric misclassification costs. By asymmetric misclassification costs we mean that one of the class values is the target class value for which we want to get predictions and we prefer false positive over false negative. The best results over domains and different number of selected features are achieved when features are scored using Odds ratio and only 50-100 best features are used (using 0.2% to 5% of all domain features).

- Evaluation of the proposed guidelines on real-world data collected from the Web:

  1. WWW-documents collected from residential users in Carnegie Mellon University (CMU) HomeNet project,
  2. domains generated from the Yahoo hierarchy of WWW-documents.

The dissertation contributes to *machine learning* as a new method for learning on text data, hypertext, class hierarchy and as such to the related research areas of

- *text-learning* and *information retrieval* proposing enriched document representation, new feature scoring measures, automatic document categorization and abstraction, and document content prediction based on the hyperlink content,
- *data mining* and *intelligent data analysis* proposing efficient methods for the above problems and new method for learning from class hierarchy
- *intelligent user interfaces* and *intelligent agents* as a method for better design of Web browsers and personal content-based software assistants

Within machine learning, it contributes an approach to

- learning on large, distributed text data and hypertext
- learning from class hierarchy

7.4 Further work

The dissertation opens new possibilities for both practical usage of learning on text data and learning from hypertext and for further improvements and extensions of the methods. We propose the following directions for further research and improvements of proposed methods.

- Further extensions to document representation by including background knowledge such as information from the hypertext graph structure, natural language dependent information, html-tagging and xml-tagging, word mapping into higher level categories using electronic thesaurus.

- An extension to multi-lingual system by using electronic dictionaries, thesaurus and constructing background knowledge for class hierarchies that are available on the Web in different natural languages.
- Study and experimental evaluation of the alternative approach to predict interesting hyperlink by learning from positive examples only using eg., association rules [1] or method described in [82].
- Analysis and experimental comparison of different learning algorithms on hypertext.
- The use of arcing [14] for life-long learning of the user profile, where more recent information has greater weight and old information is gradually 'forgotten'. The accumulated knowledge about the user can be used for user customized Web browsing, for cooperation with other user's personal software assistants covering different areas, for support to cooperative learning and information exchange between different user software assistants.
- Study and incorporation of other information about the user and user's actions while browsing the Web to improve Web browsers. One possibility is to observe actions requested from the browser like document scrolling, printing, down-loading and use them as additional information about the documents. Other possibility is to collect more general information about the user, eg., using some questionnaire.
- Extension of the user interface by adding communication with the user that will enable richer data collection. For instance, asking user for explanation of outliers and giving explanation of hyperlink classification, document content prediction and document categorization.
- Development of Web document prefetching mechanism based on the developed method for predicting clicked hyperlinks.

Finally, there is a wide range of potential application of the proposed methods. The proposed document categorization and abstraction can be used in many ways, for instance, to get a brief idea about the documents returned by some Web search engine. Learning from class hierarchy can be used to help manual document categorization based on the predefined hierarchical text structure, as for instance performed in medical informatics. Another study for further work is to form questionnaire and get the evaluation from the users that already expressed interest in using our personal browsing assistant based on the developed methods. This would enable analysis of the proposed methods from the application point of view and stimulate further extensions in this directions. An interesting potential application of the proposed methods is in offering support to user customized distance learning that is recently getting more attention on national levels of different countries and information exchange between the students with similar interests.

Bibliography

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A.I. (1996). Fast Discovery of Association Rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* AAAI Press/The MIT Press, pp. 307-328.

[2] Aha, D.W., Kibler, D. & Albert, M.K. (1991). Instance-Based Learning algorithms. *Machine Learning, 6*, 37-66. Kluwer Academic Publishers.

[3] Aha, D.W., & Bankert, R.L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison, *Proceedings of the AAAI'94 Workshop on Case-Based Reasoning* (pp. 106-112). Seattle, WA:AAAI Press.

[4] Akkus, A. & Güvenir, H.A. (1996). K Nearest Neighbor Classification on Feature Projections. *Proceedings of the 13th International Conference on Machine Learning ICML96* (pp. 12-19).

[5] Almuallin, H., & Dietterich, T.G. (1991). Efficient algorithms for identifying relevant features. *Proceedings of the Ninth Canadian Conference on Artificial Intelligence* (pp. 38-45). Vancouver, BC:Morgan Kaufmann.

[6] Apté, C., Damerau, F., & Weiss, S.M. (1994). Toward Language Independent Automated Learning of Text Categorization Models. *Proceedings of the 7th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* Dubline.

[7] Armstrong, R., Freitag, D., Joachims, T. & Mitchell, T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. *Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford.
http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/webagents-plus.ps.Z

[8] Bala, J., Huang, J., & Vafaie, H. (1995). Hybrid Learning Using Genetis Algorithms and Decision Trees for Pattern Classification. *Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI-95* (pp. 719-724), Montreal, Quebec.

[9] Balabanović, M., & Shoham, Y. (1995). Learning Information Retrieval Agents: Experiments with Automated Web Browsing. *Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.* Stanford. http://robotics.stanford.edu/people/marko/papers/lira.ps

[10] Balabanović, M. & Shoham, Y. (1997). Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM 3*, (Vol. 40), 66-70.

[11] Bartell, B.T., Cottrell, G.W. & Belew, R.K. (1992). Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling. *Proc. of the ACM SIG Information Retrieval*, Copenhage.

[12] Berry, M.W., Dumais, S.T., & OBrein, G.W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review, 4*, (Vol. 37), 573-595.

[13] Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, P.J. (1984). Classification and regression trees. Belmont, CA: Wadsworth International Group.

[14] Breiman, L. (1996). Pasting bites together for prediction in large data sets and online. Berkeley University, November.

[15] Burke, R., Hammond, K. & Kozlovsky, J. (1995). Knowledge-based Information Retrieval for Semi-Structured Text. *Working Notes from AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval* (pp. 19-24), American Association for Artificial Intelligence.

[16] Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N. & Schoenberg, S. (1997). Question Answering from Frequently Asked Question Files, *AI magazine 2*, (Vol. 18), 57-66.

[17] Cardie, C. (1993). Using Decision Trees to Improve Case-Based Learning. *Proceedings of the 10th International Conference on Machine Learning ICML93* (pp. 25-32).

[18] Caruana, R., & Freitag, D. (1994). Greedy Attribute Selection. *Proceedings of the 11th International Conference on Machine Learning ICML94* (pp. 28-26).

[19] Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. *Proceedings of the 9th European Conference on Aritficial Intelligence ECAI'90* (pp. 147-149), Stockholm, Sweden.

[20] Cherkauer, K.J., & Shavlik, J.W. (1996). Growing simpler decision trees to facilitate knowledge discovery, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD-96* (pp. 315-318). Portland, OR:AAAI Press.

[21] Cohen, W.W. (1995). Learning to Classify English Text with ILP Methods. *Proceedings of the Workshop on Inductive Logic Programming.* Leuven.

[22] Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning, 29*, 103-130. Kluwer Academic Publishers.

[23] Drummond,C., Ionescu, D. & Holte, R. (1995). A Learning Agent that Assists the Browsing of Software Libraries. *Technical Report TR-95-12*, Computer Science Dept., University of Ottawa.

[24] Dubrawski, A.W. & Moore, A.W. (1997). Scoring discriminative capabilities of data attributes using Bayesian classifier and Monte Carlo integration. *Proceedings of the ECML97 Workshop on Case-Based Learning: Beyond Classification of Feature Vectors.*

[25] Mc Elligott, M. & Sorensen, H. (1993). An emergent approach to information filtering. *Abakus. U.C.C. Computer Science Journal 4*, (Vol 1).

[26] Etizioni, O. & Weld, D. (1994). A Softbot-Based Interface to the Internet. *Communications of the ACM 7*, (Vol. 37), 72-79.

[27] Filo, D., & Yang, J. (1997). Yahoo! Inc. http://www.yahoo.com/docs/pr/

[28] Foltz, P. W. & Dumais, S. T. (1992). Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM 12*, (Vol. 35), 51-60.

[29] Frakes W.B., & Baeza-Yates R., (Eds.) (1992). Information Retrieval: Data Structures & Algorithms. Englewood Cliffs: Prentice Hall.

[30] Gams, M. & Grobelnik, M. (1997). Intelligent agents in information society. *Proceedings of the 6th Electrotechnical and Computer Science Conference ERK'97* (pp. 125-128). Ljubljana, Slovenia: Slovenia Section IEEE.

[31] Gams, M. & Hribovšek, B. (1996). Intelligent-personal-agent interface for operating systems. *Applied Artificial Intelligence 10*, 353-383.

[32] Goldman, C.V., Langer, A. & Rosenschein, J.S. (1997). Musag: an agent that learns what you mean. *Applied Artificial Intelligence 11*, 413-435.

[33] Grobelnik, M., & Mladenić, D. (1998). Learning Machine: design and implementation. *Technical Report IJS-DP-7824*. Department for Intelligent Systems, J.Stefan Institute, Slovenia.

[34] Grobelnik, M. & Mladenić, D. (1998). Efficient text categorization. *Proceedings of the ECML-98 Text Mining workshop.*

[35] Hammond, K., Burke, R. & Schmitt, K. (1994). A Case-Based Approach to Knowledge Navigation. *Proceedings of the AAAI Workshop on Indexing and Reuse in Multimedia Systems* (pp. 46-57), American Association for Artificial Intelligence.

[36] Hedberg, S. (1997). Agents for sale: first wave of intelligent agents go commercial. *IEEE Expert Intelligent systems & their applications 6*, (Vol. 11), 16-19.

[37] Kraut, R., Scherlis, W., Mukhopadhyay, T., Manning, J. & Kiesler, S. (1996). The HomeNet Field Trial of Residential Internet Services. *Communications of the ACM 12*, (Vol. 39), 55-63. http://homenet.andrew.cmu.edu/Progress/

[38] Holte, R.C. & Drummond, C. (1994). A Learning Apprentice For Browsing. *Proceedings of the AAAI Spring Symposium on Software Agents.*

[39] Joachims, T., Mitchell, T., Freitag, D. & Armstrong, R. (1995). WebWatcher: Machine Learning and Hypertext. *Fachgruppentreffen Maschinelles Lernen*, Dortmund.

[40] Joachims, T. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. *Proceedings of the 14th International Conference on Machine Learning ICML97* (pp. 143-151).

[41] Joachims, T., Freitag, D. & Mitchell, T. (1997). WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the 15th International Joint Conference on Artificial Intelligence IJCAI-97* (pp. 770-775).

[42] John, G.H., Kohavi, R., & Pfleger, K. (1994). Irrelevant Features and the Subset Selection Problem. *Proceedings of the 11th International Conference on Machine Learning ICML94* (pp. 121-129).

[43] Kamba, T., Sakagami, H. & Koseki, Y. (1997). ANATAGONOMY: a personalized newspaper on the World Wide Web. *International Journal Human-Computer Studies 46*, 789-803.

[44] Karalič, A. (1992). Employing linear regression in regression tree leaves. *Proceedings of the 4th European Conference on Machine Learning ECML92.*

[45] Kautz, H., Selman, B. & Shah, M. (1997). Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM 3*, (Vol. 40), 63-65.

[46] Kautz, H., Selman, B. & Shah, M. (1997). The Hidden Web. *AI magazine 2*, (Vol. 18), 27-36.

[47] Kindo, T., Yoshida, H., Morimoto, T., & Watanabe, T. (1997). Adaptive Personal Information Filtering System that Organizes Personal Profiles Automatically. *Proceedings of the 15th International Joint Conference on Artificial Intelligence IJCAI-97* (pp. 716-721).

[48] Kira, K., & Rendell, L.A. (1992). The feature selection problem: Traditional methods and a new algorithm. *Proceedings of the Ninth National Conference on Artificial Intelligence AAAI-92* (pp.129-134). AAAI Press/The MIT Press.

[49] Kononenko, I. (1995). On biases estimating multi-valued attributes. *Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI-95* (pp. 1034-1040), Montreal, Quebec.

[50] Kononenko, I. & Bratko, I. (1991). Information-Based Evaluation Criterion for Classifier's Performance. *Machine Learning 6*, 67-80. Kluwer Academic Publishers.

[51] Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. & Riedl, J. (1997). GroupLense: Applying Filtering to Usenet News. *Communications of the ACM 3*, (Vol. 40), 77-87.

[52] Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proceedings of the 13th International Conference on Machine Learning ICML96* (pp. 284-292).

[53] Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. *Proceedings of the 14th International Conference on Machine Learning ICML97* (pp. 170-178).

[54] de Kroon, H.C.M, Mitchell, T. & Kerckhoffs, E.J.H. (1996). Improving Learning Accuracy in Information Filtering. *Proceedings of the ICML-96 Workshop Machine learning meets human computer interaction.* http://www.ics.forth.gr/ moustaki/ICML96_HCI_ML/kroon.ps

[55] Krulwich, B. & Burkey, C. (1996). The ContactFinder agent: Answering buletin board questions with referrals. *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96* (pp.10-15).

[56] Krulwich, B. (1997). Lifestyle Finder. *AI magazine 2*, (Vol. 18), 37-46.

[57] Kubat, M., Holte & R., Matwin, S. (1997). Learning When Negative Examples Abound. *Proceedings of 9th European Conference on Machine Learning ECML97*, (pp. 146-153).

[58] Kubat, M. & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. *Proceedings of the 14th International Conference on Machine Learning ICML97* (pp. 179-186).

[59] Kukar, M., Kononenko, I. (1998). Cost-sensitive learning with neural networks. *Proceedings of the 13th European Conference on Aritficial Intelligence ECAI'98* (pp. 445-449).

[60] Lam, W., Low, K.F. & Ho, C.Y. (1997). Using Bayesian Network Induction Approach for Text Categorization. *Proceedings of the 15th International Joint Conference on Artificial Intelligence IJCAI97* (pp. 745-750).

[61] LaMacchia, B.A. (1996). Internet Fish, *A revised version of a thesis proposal at MIT.*

[62] Lang, K. (1995). News Weeder: Learning to Filter Netnews. *Proceedings of the 12th International Conference on Machine Learning ICML95.*

[63] Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and applications.* Ellis Horwood.

[64] Lewis, D.D. (1992). Feature Selection and Feature Extraction for Text Categorization. *Proceedings of Speech and Natural Language workshop* (pp. 212-217). Harriman, CA:Morgan Kaufmann.

[65] Lewis, D.D. (1995). Evaluating and optimizating autonomous text classification systems. *Proceedings of the 18th Annual International ACM-SIGIR Conference on Recsearch and Development in Information Retrieval* (pp.246-254).

[66] Lewis, D.,D. & Gale, W., A. (1994). A Sequential Algorithm for Training Text Classifiers. *Proceedings of the 7th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dubline.

[67] Lieberman, H. (1995). Letizia: An Agent that Assists Web Browsing. *14th International Joint Conference on Artificial Intelligence IJCAI95.*

[68] Liu, H., & Setiono, R. (1996). A probabilistic approach to feature selection - A filter solution. *Proceedings of the 13th International Conference on Machine Learning ICML'97* (pp. 319-327). Bari.

[69] Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM 7*, (Vol. 37), 30-40.

[70] Manly, B.F.J. (1994). Multivariate Statistical Methods - a primer, Chapman & Hall. Second edition.

[71] Mansuripur, M. (1987). *Introduction to Information Theory.* Prentice-Hall.

[72] Menczer, F. (1997). ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhood for Information Discovery. *Proc. of the Fourteenth International Conference on Machine Learning ICML '97* (pp. 227-235).

[73] Mitchell, T.M. (1997). Machine Learning. The McGraw-Hill Companies, Inc..

[74] Mitchell, T., Caruana, R., Freitag, D., McDermott, J. & Zabowski, D., Experience with a Learning Personal Assistant, *Communications of the ACM* Vol. 37, No. 7, pp.81—91, July 1994. http://www.cs.cmu.edu/afs/cs/user/mitchell/ftp/cacm.ps.Z

[75] Mladenić, D. (1995). Automated model selection. *Proceedings of the MLNet familiarisation workshop: Knowledge level modelling and machine learning, ECML 95*, Heraklion.

[76] Mladenić, D. (1995). Domain-Tailored Machine Learning. *M.Sc. Thesis*, Faculty of computer and information science, University of Ljubljan, Slovenia.

[77] Mladenić, D. (1996). Personal WebWatcher: Implementation and Design. *Technical Report IJS-DP-7472.* Department for Intelligent Systems, J.Stefan Institute, Slovenia.

[78] Mladenić, D. (1998). Feature subset selection in text-learning. *Proceedings of the 10th European Conference on Machine Learning ECML98.*

[79] Mladenic, D. (1998). Turning Yahoo into an Automatic Web-Page Classifier. *Proceedings of the 13th European Conference on Aritficial Intelligence ECAI'98* (pp. 473-474).

[80] Mladenić, D., & Grobelnik, M. (1998). Word sequences as features in text-learning. *Proceedings of the Seventh Electrotechnical and Computer Sc. Conference ERK'98.* Ljubljana, Slovenia: IEEE section.

[81] Muggletom, S. (1991). Inductive Logic Programming. *New Generation Computing 8*, (Vol. 4), 295-318.

[82] Muggletom, S. (1996). Learning from positive data. *Proceedings of the 6th International Workshop on Inductive Logic Programming ILP'96* (pp. 225-244).

[83] Pazzani, M., Muramatsu, J. & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, March 1996 and *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96* (pp.54-61).

[84] Pazzani, M., & Billsus, D. (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning, 27*, 313-331. Kluwer Academic Publishers.

[85] Pfahringer, B. (1995). Compression-Based Feature Subset Selection. In P. Turney (Ed.), *Proceedings of the IJCAI-95 Workshop on Data Engineering for Inductive Learning*, Workshop Program Working Notes, Montreal, Canada, 1995.

[86] Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning 1*, 81-106. Kluwer Academic Publishers.

[87] Quinlan, J.R. (1993). Constructing Decision Tree. In *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers. pp. 17-26.

[88] Rachlin, J., Kasif, S., Salzberg, S. & Aha, D.W. (19914). Torwards a Better Understanding of Memory-Based Reasoning Systems. *Proceedings of the 11th International Conference on Machine Learning ICML94* (pp. 242-250).

[89] van Rijsbergen, C.J,. Harper, D.J., & Porter, M.F. (1981). The selection of good search terms. *Information Processing & Management, 17*, 77-91.

[90] Rocchio, J. (1971). Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, Chapter 14, pp.313-323, Prentice-Hall Inc..

[91] ARTFL project, Roget thesaurus (1997).
http://humanities.uchicago.edu/forms_unrest/ROGET.html

[92] Rucker, J. & Marcos, J.P. (1997). Siteseer: Personalized Navigation for the Web. *Communications of the ACM 3*, (Vol. 40), 73-75.

[93] Salton, G. & Buckley, C. (1987). Term Weighting Approaches in Automatic Text Retrieval. *Technical report, COR-87-881*, Department of Computer Science, Cornell University, November.

[94] Shapiro, A. (1987). *Structured induction in expert systems*. Addison-Wesley.

[95] Shaw Jr, W.M. (1995). Term-relevance computations and perfect retrieval performance. *Information Processing & Management, 31(4)* 491-498.

[96] Skalak, D.B. (1994). Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. *Proceedings of the 11th International Conference on Machine Learning ICML94* (pp. 293-301).

[97] Sorensen, H. & McElligott, M. (1995). PSUN: A Profiling System for Usenet News. *Proceedings of CIKM'95 Intelligent Information Agents Workshop*, Baltimore.

[98] Terveen, L., Hill, W., Amento, B., McDonald, D. & Creter, J. (1997). PHOAKS: A System for Sharing Recommendations. *Communications of the ACM 3*, (Vol. 40), 59-62.

[99] Vafaie, H., & De Jong, K. (1993). Robust feature selection algorithms. *Proceedings of the 5th IEEE International Conference on Tools for Artificial Intelligence* (pp. 356-363). Boston, MA: IEEE Press.

[100] Yang, Y. (1994). Expert Network: Effective and Efficient Learning form Human Decisions in Text Categorization and Retrieval. *Proc. of the 7th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dubline.

[101] Yang, Y., & Pedersen, J.O. (1997). A Comparative Study on Feature Selection in Text Categorization. *Proceedings of the 14th International Conference on Machine Learning ICML-97* (pp. 412-420).