Universität des Saarlandes
Naturwissenschaftlich-Technische Fak. I
Fachrichtung 6.2 - Informatik

Max-Planck-Institut für Informatik
AG 5 - Datenbanken und Informationssysteme
Prof. Dr. Ing. Gerhard Weikum

Combination Methods for Automatic Document Organization

Stefan Siersdorfer

Dissertation
zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften
an der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

November 2005

Abstract

Automatic document classification and clustering are useful for a wide range of applications such as organizing Web, intranet, or portal pages into topic directories, filtering news feeds or mail, focused crawling on the Web or in intranets, and many more. This thesis presents ensemble-based meta methods for supervised learning (i.e., classification based on a small amount of hand-annotated training documents). In addition, we show how these techniques can be carried forward to clustering based on unsupervised learning (i.e., automatic structuring of document corpora without training data). The algorithms are applied in a restrictive manner, i.e., by leaving out some 'uncertain' documents (rather than assigning them to inappropriate topics or clusters with low confidence).

We show how restrictive meta methods can be used to combine different document representations in the context of Web document classification and author recognition. As another application for meta methods we study the combination of different information sources in distributed environments, such as peer-to-peer information systems. Furthermore we address the problem of semi-supervised classification on document collections using retraining. A possible application is focused Web crawling which may start with very few, manually selected, training documents but can be enhanced by automatically adding initially unlabeled, positively classified Web pages for retraining.

The results of our systematic evaluation on real world data show the viability of the proposed approaches.

Kurzfassung

Automatische Dokumentklassifikation und Clustering sind für eine Vielzahl von Anwendungen von Bedeutung, wie beispielsweise Organisation von Web-, Intranet- oder Portalseiten in thematische Verzeichnisse, Filterung von Nachrichtenmeldungen oder Emails, fokussiertes Crawling im Web oder in Intranets und vieles mehr. Diese Arbeit untersucht Ensemble-basierte Metamethoden für Supervised Learning (d.h. Klassifikation basierend auf einer kleinen Anzahl von manuell annotierten Trainingsdokumenten). Weiterhin zeigen wir, wie sich diese Techniken auf Clustering basierend auf Unsupervised Learning (d.h. die automatische Strukturierung von Dokumentkorpora ohne Trainingsdaten) übertragen lassen. Dabei wenden wir die Algorithmen in restriktiver Form an, d.h. wir treffen keine Aussage über eine Teilmenge von "unsicheren" Dokumenten (anstatt sie mit niedriger Konfidenz ungeeigneten Themen oder Clustern zuzuordnen).

Wir verwendenen restriktive Metamethoden um unterschiedliche Dokumentepräsentationen, im Kontext der Klassifikation von Webdokumentem und der Autorenerkennung, miteinander zu kombinieren. Als weitere Anwendung von Metamethoden untersuchen wir die Kombination von unterschiedlichen Informationsquellen in verteilten Umgebungen wie Peer-to-Peer Informationssystemen. Weiterhin betrachten wir das Problem der Semi-Supervised Klassifikation von Dokumentsammlungen durch Retraining. Eine mögliche Anwendung ist fokussiertes Web Crawling, wo wir mit sehr wenigen, manuell ausgewählten Trainingsdokumenten starten, die durch Hinzufügen von ursprünglich nicht klassifizierten Dokumenten ergänzt werden.

Die Resultate unserer systematischen Evaluation auf realen Daten zeigen das gute Leistungsverhalten unserer Methoden.

Summary

This thesis addresses the problem of automatically structuring heterogenous document collections into thematically coherent subsets. This issue is relevant for a variety of applications, such as organizing large personal email folders, dividing topics in large Web directories into subtopics, structuring large amounts of company and intranet data, focused crawling on the Web, and many more. The methods of choice for accomplishing this are either based on supervised classification, which requires explicit, manually labeled, training data, or unsupervised clustering.

In this thesis we investigate the engineering and, in particular, tuning issues of using automatic classification and clustering algorithms for text document organization. We study ensemble-based meta methods for supervised learning (i.e., classification based on a small amount of hand-annotated training documents). In addition, we show how these techniques can be carried forward to clustering based on unsupervised learning (i.e., automatic structuring of document corpora without training data). The algorithms are applied in a restrictive manner, i.e., by leaving out some 'uncertain' documents (rather than assigning them to inappropriate topics or clusters with low confidence).

We develop a constructive and practically efficient methodology for tuning a repertoire of classifiers and meta methods to the application's specific goals in terms of classification error and document loss (the fraction of documents where the restrictive classifier abstains). A key element in our approach is to devise analytic estimators that can predict the error and loss for a given parameter setting sufficiently accurately. Although our techniques are anchored upon empirical leave-one-out or cross-validation estimators on the underlying training data to some extent, we take great care to avoid computationally expensive steps that would involve repeated retraining.

In the classical scenario it is often assumed that all topic categories (classes) are known and that the training corpus provides example documents for all these categories. However in many real world applications these assumptions do not hold. As an example consider a focused crawler where we are interested just in a limited number of topics and subtopics. Here we have to deal with the problem that the Web covers such a large variety (and growing number) of other topics that it is impossible to build a training set that comprises all these topics. However a focused crawler will very likely see such "junk documents", although the underlying classifier has never seen (and never had a chance to see) any training data for the "junk" class, and will

have to make a decision about them. We show, by a probabilistic model as well as by experiments on various data sets, that restrictive classification methods can be used to eliminate junk documents.

In many situations explicit training data is unavailable, so that clustering is the only viable option. Conventional clustering methods partition the entire data set into clusters, but this may lead to poor results in terms of cluster impurity, for example, mixing thematically unrelated documents into the same cluster. We propose an approach for automatically clustering heterogenous document collections by using restrictive clustering methods. A key element in our approach is to construct restrictive meta methods that result in higher cluster purity. The introduced algorithms ensure better accuracy and make clustering results robust and accurate at the cost of a moderate loss of uncertain samples.

In the context of a distributed peer-to-peer (P2P) network that connects multiple users with shared topics of interest, it is natural to aggregate their knowledge and construct better machine learning models that could be used by every network member for his information demands. We address this task using meta methods. We combine multiple independently learned models from several peers and construct the advanced decision model that simultaneously takes the knowledge of multiple P2P users into account in a decentralized manner.

We describe classification with different document representations. In addition to well known features like document terms in the Bag-Of-Words model, part-of-speech tagging, etc., we consider alternative stylistic features like the depth or the structure of syntax trees. We combine the feature representations using two techniques: 1) combination vectors, where we construct a single vector from the different feature vectors, 2) meta methods combining the classification results based on the different representations into a meta result.

Our work on semi-supervised classification is motivated by the fact that the availability of training data is often a critical point in real applications. This has led us to a semi-supervised learning approach with iterative retraining using initially unlabeled data. An additional difficulty that real applications often pose is the imbalance between acceptable and unacceptable documents in the corpus that creates a mismatch with the ratio of positive and negative training samples and may result in a wrong bias of the classifier. In Web applications, but also for large-scale intranet corpora, this is a typical situation and creates a major impediment to the previously proposed state-of-the-art techniques for semi-supervised classification. Our method successfully addresses these practically relevant issues, which were largely disregarded by prior work, and significantly outperforms the other methods in terms of classification accuracy.

Zusammenfassung

Diese Arbeit untersucht die Problematik der automatischen Strukturierung von Dokumentkorpora in thematisch koheränte Teilmengen. Dies ist für eine Vielzahl von Anwendungen von Bedeutung, wie die Organisation von großen Email-Ordnern, Aufteilung von Themen innerhalb großer Webverzeichnisse in Unterthemen, Strukturierung großer Mengen von Unternehmens- und Intranetdaten, fokussiertes Crawling im Web und vieles mehr. Die Methoden der Wahl für solche Aufgaben basieren entweder auf supervised Klassifikation, die eine Menge von manuell kategorisierten Trainingsdokumenten erfordert, oder auf unsupervised Clustering.

In dieser Arbeit untersuchen wir die Ingenieurs- und insbesondere Tuning-Aspekte der Nutzung von automatischen Klassifikations- und Clusteringalgorithmen zur Organisation von Textdokumenten. Wir betrachten ensemble-basierte Metamethoden für das Supervised Learning (d.h. Klassifikation basierend auf einer kleinen Anzahl von manuell annotierten Trainingsdokumenten). Weiterhin zeigen wir, wie sich diese Techniken auf das Clustering basierend auf Unsupervised Learning (d.h. automatische Strukturierung von Dokumentkorpora ohne Verwendung von Trainingsdaten) übertragen lassen. Wir wenden diese Algorithmen auf 'restriktive' Weise an, d.h. durch Auslassen von 'unsicheren' Dokumenten (anstatt sie mit geringer Konfidenz zu unpassenden Themen oder Clustern zuzuordnen).

Wir entwickeln eine konstruktive und effiziente Methode zum Tuning einer Menge von Klassifikatoren und Metamethoden um anwendungsspezifische Ziele hinsichtlich des Errors der Klassifikation und des Dokument Loss (Anteil der Dokumente über die der restriktive Klassifikator keine Aussage trifft) zu erzielen. Ein Schlüsselelement unseres Ansatzes ist die Entwicklung analytischer Estimatoren, die Error und Loss für eine gegebene Wahl der Parameter hinreichend genau bestimmen können. Obwohl unsere Techniken zu einem gewissen Maß auf empirischen Leave-One-Out- oder Crossvalidierungs-Estimatoren auf den zugrunde liegenden Trainingsdaten basieren, vermeiden wir sorgfältig teure Berechnungen, die ein wiederholtes Neutrainieren von Klassifikatoren bedingen würden.

Im klassischen Szenario wird oft angenommen, dass alle thematischen Kategorien (Klassen) bekannt sind und dass der Trainingkorpus Beispiele für all diese Kategorien bereitstellt. Allerdings trifft dies für viele reale Anwendungen nicht zu. Man betrachte zum Beispiel einen fokussierten Crawler, bei dem wir nur an einer begrenzten Anzahl von Themen und Unterthemen interessiert sind. Hier tritt das Problem

auf, dass im Web eine solche Vielfalt (und wachsende Anzahl) von Themen existiert, dass es praktisch unmöglich ist, eine Trainingsmenge zu konstruieren, die alle diese Themen abdeckt. Ein fokussierter Crawler wird jedoch mit sehr hoher Wahrscheinlichkeit mit solchen 'Junk-Dokumenten' konfrontiert werden, obwohl der zugrunde liegende Klassifikator auf Trainingsdaten der 'Junk' Klasse nicht trainiert wurde. Trotzdem muß zu solchen Dokumenten eine Entscheidung getroffen werden. Wir zeigen, sowohl mit einem probabilistischen Modell als auch durch Experimente auf unterschiedlichen Datensätzen, dass man restriktive Klassifikationsmethoden zur Eliminierung von Junk-Dokumenten verwenden kann.

In vielen Situationen sind keine expliziten Trainingsdaten verfügbar, so dass Clustering die einzig realisierbare Option ist. Konventionelle Clusteringmethoden partitionieren die gesamte Datenmenge in Cluster, aber dies kann zu schlechten Resultaten in Form von unreinen Clustern führen, zum Beispiel bei Vermischung von thematisch nicht verwandten Dokumenten im selben Cluster. Wir schlagen einen Ansatz zum automatischen Clustering von heterogenen Dokumentsammlungen vor, der auf restriktiven Clusteringmethoden basiert. Ein Schlüsselelement bei diesem Ansatz ist die Konstruktion von restriktiven Metamethoden, die zu einer höheren Reinheit der Cluster führen. Die eingeführten Algorithmen gewährleisten eine bessere Clusterqualität und machten die Clusterresultate robust, auf Kosten eines moderaten Dokument-Loss durch unsichere Beispiele.

Im Kontext eines verteilten Peer-to-Peer (P2P) Netzwerks, das mehrere Benutzer mit gemeinsamen Interessengebieten miteinander verbindet, ist es natürlich, deren Wissen zu aggregieren und bessere maschinelle Lernmodelle zu konstruieren, die von jedem Mitglied des Netzwerkes für seine Informationsbedürfnisse genutzt werden können. Wir lösen diese Aufgabe mithilfe von Metaverfahren. Wir kombinieren mehrere unabhängig voneinander gelernete Modelle von unterschiedlichen Peers und konstruieren ein erweitertes Entscheidungsmodell, das gleichzeitig das Wissen mehrerer Peers auf dezentrale Weise einbezieht.

Wir beschreiben Klassifikation mit unterschiedlichen Dokumentrepräsentationen. Zusätzlich zu wohlbekannten Features, wie Dokumentterme im Bag-Of-Words-Modell, Part-of-Speech-Tags, usw., betrachten wir alternative stilistische Features wie die Tiefe und die Struktur von Syntaxbäumen. Wir kombinieren die Featurerepräsentationen durch zwei Techniken: 1) Kombinationsvektoren, wo wir einen einzelnen Featurevektor aus mehreren Fearturevektoren konstruieren und 2) Metamethoden, die die auf unterschiedlichen Repräsentationen basierenden Klassifikationsresultate zu einem Metaresultat kombinieren.

Unsere Arbeit über semi-supervised Klassifikation ist durch die Tatsache motiviert, dass die Verfügbarkeit von Trainingsdaten oft ein kritischer Punkt bei realen Anwendungen ist. Dies führt uns zu einem semi-supervised Ansatz, der auf iterativem Retraining mit initial nicht klassifizierten Daten basiert. Eine zusätzliche Schwierigkeit bei realen Anwendungen ist das Auftreten von Unbalanciertheiten von akzeptablen und unakzeptablen Dokumenten in einem Korpus, der zu einer Fehlanpassung des Verhältnisses von positiven und negativen Trainingsbeipielen, und damit zu einer Fehlausrichtung des Klassifikators, führen kann. Bei Web-Applikationen, aber auch für große Intranetkorpora ist dies eine typische Situation und stellt ein Hindernis für in der Vergangenheit vorgeschlagene Techniken zur semi-supervised Klassifikation dar. Unsere Methode behandelt diese praktisch relevanten Punkte erfolgreich und übertrifft die anderen Methoden signifikant hinsichtlich der Klassifikationsgüte.

Contents

1 Motivation and Overview

1.1 Motivation

This thesis addresses the problem of automatically structuring heterogenous document collections into thematically coherent subsets. This issue is relevant for a variety of applications, such as organizing large personal email folders, dividing topics in large Web directories into subtopics, structuring large amounts of company and intranet data, focused crawling on the Web, and many more [29]. The methods of choice for accomplishing this are either based on supervised classification, which requires explicit, manually labeled, training data, or unsupervised clustering.

Restrictive Meta Classification

There exists a great variety of classification methods such as Naive Bayes, kNN, Rocchio, linear SVM, etc. [88, 29, 43], all of which operate on a high-dimensional feature space usually constructed from word occurrence frequencies in documents (and possibly some additional input such as anchor texts in hyperlink neighbors, neighbor topics, etc.). All methods face inherent *tradeoffs* regarding the result quality metrics: precision, which is the fraction of documents that are automatically placed under some topic and do indeed belong there (as per a human expert's assessment), can be improved by making the classifier more conservative, but this way recall, which is the fraction of positively classified documents among all documents that the human expert would place under the given topic, usually becomes worse.

Whether precision or recall is more important is application dependent, and this raises the need for tuning classification methods towards the goals of the application. To this end we exploit the fact that many methods have certain calibration parameters anyway by which we can control their degrees of making more conservative or more speculative decisions. For example, a Bayesian classifier may accept a given document for some topic only if the probability of the document belonging there exceeds some specific threshold. Similarly, an SVM (support vector machine) classifier may choose to accept only documents whose positive distance from the separating hyperplane is above some threshold (SVM outputs can be also mapped to probabilities as described, e.g., in [97]). In this work we will also discuss families of “ensemble-based” meta methods (combining results from multiple classifiers) that come with explicitly designed parameters of this kind.

We consider classifiers for a given topic that make a ternary decision on a newly seen document: they can accept the document for the topic, reject it, or abstain if there is neither sufficiently strong evidence for acceptance nor for rejection. The third option is important as it makes a key difference for constructing meta classifiers that combine the results of different classifiers (e.g., in a quorum consensus manner). Now the quality metrics of interest are primarily the classification *error*, which is the fraction of erroneously accepted or erroneously rejected documents, and the document *loss*, which is the fraction of documents for which the classifier or meta *classifier* makes no decision at all (i.e., abstains).

Junk Elimination

In the classical scenario it is often assumed that all topic categories (classes) are known and that the training corpus provides example documents for all these categories. However in many real world applications these assumptions do not hold. As an example consider a focused crawler where we are interested just in a limited number of topics and, as the case may be, subtopics. Here we have to deal with the problem that the Web covers such a plethora (and growing number) of other topics that it is impossible to build a training set that comprises all these topics. However a focused crawler will very likely see such ”junk documents”, although the underlying classifier has never seen (and never had a chance to see) any training data for the “junk” class, and will have to make a decision about them. It is not clear how a classifier trained to discriminate topics based on training data about ”computer sciene”, ”mathematics”, and ”physics” will behave on documents about, say, ”esoterism”; there is a significant difference between negative examples and ”junk” documents.

We propose restrictive classification methods to tackle the ”junk problem”. In restrictive classification, we consider classifiers for a given topic that make a ternary decision on a newly seen document: they can accept the document for the topic, reject it for the topic, or abstain if there is neither sufficiently evidence for acceptance nor for rejection. With the abstention option we aim to achieve a lower error on the remaining documents and to eliminate the junk documents that would be spuriously assigned to one of the classes of interest.

Restrictive Meta Clustering

In many situations explicit training data is unavailable, so that clustering is the only viable option.

Conventional clustering methods partition the entire data set into clusters, but this may lead to poor results in terms of cluster impurity, for example, mixing thematically unrelated documents into the same cluster. The approach that we advocate and further develop is to cluster only a subset of the available data, but do so with a higher clustering quality. This is analogous to restrictive classification in the supervised scenario. The left-out data which is not assigned to any cluster is collapsed into an extra container with “miscellaneous” documents.

We call this approach restrictive clustering methods. With simple restrictive methods we provide restrictive modifications of conventional clustering methods; with meta-clustering methods we combine different clustering methods in a restrictive way. Additionally, by using supervised learning techniques (e.g., SVM-based text classification) on the results of the restrictive clustering methods and meta methods, we can generalize the clusters to a larger subset or even the entire dataset.

As a possible application scenario for our techniques consider a focused Web crawler [29, 116]. Such a crawler starts with a set of training documents for a given topic or an entire topic directory, e.g., for topic ”sports” with subtopics ”ball games”, ”track and fields”, and ”swimming”. The result of a large crawl populate these explicitly labeled classes, and we may obtain a huge number of documents in the ball games topics and a much smaller number of documents in the other two subclasses. Obviously this suggests that the originally given topic directory was not wisely chosen and should have foreseen additional subsubtopics under the ball games class. What we would like to achieve now is an automatic, but unsupervised, organization of the ball games documents by partitioning this class into appropriate subsubclasses. In doing this we strive for high accuracy in the sense that whatever subsubclasses we form should indeed be reasonably homogeneous, but it would be perfectly acceptable to completely leave out documents for which a cluster assignment can be made only with very low confidence. These left out documents would simply be considered as a subsubtopic ”/sports/ball games/miscellaneous”. Suppose we can create three new clusters that correspond to ”soccer”, ”basketball”, and ”handball” documents. Initially, these clusters would be unlabeled, but if each of them has very high thematic purity then the user could easily assign labels after inspecting a few samples from each class (or additional methods based on term statistics analysis could automatically suggest appropriate class labels). This is when the restrictive nature of our clustering methods and the resulting higher class purity, relative to traditional clustering, would pay off towards making personal data collections self-organizing.

Document Organization in P2P Systems

In the context of a distributed peer-to-peer (P2P) network that puts together multiple users with shared topics of interest, it is natural to aggregate their knowledge and construct better machine learning models that could be used by every network member for his information demands. The naive solution would be to share available data (training samples and/or results of the focused crawl) along a higher number of peers with others. However, the following reasons may prevent the peer from sharing all of its data with other members of the overlay network:

- significantly increased network costs for downloads of additional training data on every peer
- increased runtimes for the training of the decision models
- privacy, security, and copyright aspects of the user's personal information sources

We tackle this problem using *meta methods*. Our objective is to combine multiple independently learned models from several peers and to construct the advanced decision model that simultaneously takes the knowledge of multiple P2P users into account in a decentralized manner.

Document Representations and their Combination

Most of the text classification approaches deal with topic-oriented classification (e.g. classifying documents into classes like ”Sports”, ”Politics” or ”Computer Science”). Here the Bag-Of-Words model, taking just the occurrences of words into account (often using additional techniques like stemming, stopword elimination and different weighting schemes), has been shown to be very effective for this task [66, 130].

But there are also classification tasks like author recognition (e.g. classifying a document as either written by ”A.C. Doyle” or by ”A. Christie”), classifying different writing styles (e.g., adults vs. children, native vs. non-native speakers, etc.), or the classification of opinions (e.g. training a classifier to distinguish ”positive” from ”negative” movie reviews). Although a certain amount of topic and word correlation occurs here, too (in books written by Doyle we will typically find the names ”Holmes” and ”Watson”, in books written by Christie we have, e.g., ”Poirot” and ”Marple”), in addition, writing style and linguistic information become important issues.

In this thesis we study, in addition to some known approaches (like Bag-Of-Words, filtering based on Part-Of-Speech tagging, etc.), several new approaches for feature construction. These include writing style features using the syntax structure, considering term tuples according to the constituent structure, or syntax tree depth distributions. Some of these techniques even do not use any word information at all. As a result we obtain different and, to a certain degree, orthogonal document representations. We combine these representations using two different techniques: combination vectors and meta classification.

Another application domain is the classification of Web documents, where we can take specific features like anchor texts and the neighborhood of the documents into account.

Adaptive Retraining

In some applications, the availability of good training data for the classifier is the key bottleneck. As an example, consider a personalized or community information tool that uses thematically focused crawling [29] to build and maintain a directory or index for browsing, search, or recommendations. The quality of this approach stands or falls with the amount and quality of the training data that the underlying classifier has available. But this training data has to be provided by the user, and it is unrealistic to expect humans to spend extensive time on this task. Thus, focused crawling is often bound to start with a few dozen or at most a few hundred training documents (e.g., taken from bookmark collections).

To overcome the training bottleneck, semi-supervised learning techniques could be applied. In our given setting, the classifier could be bootstrapped by training it with whatever explicitly class-labeled training data are available and used for making decisions about the classes of previously unseen, unlabeled test documents retrieved by the crawler. These decisions would have a certain degree of uncertainty, depending on the classifier's statistical learning model. However, some of the test documents are usually accepted for their corresponding classes with high statistical confidence, and these could then be selected for re-training the classifier, now with considerably more training documents. Obviously, this simple idea does not provide a robust solution, for the automatically selected, additional training data may also increase the classifier's uncertainty and may eventually lead to an unintended topic drift. We address the issue of how to make such a semi-supervised classifier robust and practically viable.

1.2 Contribution

This thesis makes the following contributions:

- We develop a *methodology for tuning* a classifier or committee-based meta classifier to the *error and loss goals* of a given application. As base methods we consider linear SVM and a simple but robust centroid separation method, but our approach to meta classification, estimation, and tuning would apply to other base classifiers (e.g., Naive Bayes) as well. We show how to leverage *split-based meta methods* for the purpose of goal-oriented tuning. We investigate under which conditions it is beneficial to split a larger set of training documents into subsets for independent training of multiple classifiers whose decisions for previously unseen document are then combined in a quorum consensus manner. We develop *estimators* for predicting the error and loss of a specific meta classifier setup (with specific parameter settings). We use special care in ensuring that the estimations and the training phase for the classifiers under consideration are efficient, so that interactive exploration of document collections with repeated re-training is feasible.

- We develop decision procedures for *junk elimination* based on restrictive classifiers and meta classifiers. We develop a probabilistic explanation model which analytically shows that the elimination ratio of junk documents is larger than the loss of potentially interesting documents. We present comprehensive experiments, using four different data sets, including a Web document collection, that demonstrate the benefits of meta classification for junk elimination.

- We show that, analogous to restrictive meta classification, restrictive meta methods can be applied to combine different *clustering* results. We identify a particularly beneficial technique, coined metamapping, where we combine the clusters found under different simple clustering methods. We provide a simple probabilistic model that explains why the meta technique improves accuracy at the expense of "losing" some fraction of documents (which will then be organized into the "miscellaneous" container); the model could even be used for approximately predicting the achievable accuracy and loss of our techniques. We show how to combine the restrictive clustering methods and meta methods with standard classification such as SVM. We provide a comprehensive experimental study of the pros and cons of a variety of methods, including our metamapping technique and also transductive SVMs.

- For *collaborative peer-to-peer (P2P) systems*, we combine multiple independently learned models from several peers and construct an advanced decision model that takes simultaneously the knowledge of multiple P2P users into account in a decentralized and restrictive manner.
- We describe alternative ways to construct *feature spaces* for the domains of *Web document classification* and *author recognition* and show how the different document representations can be combined using meta classification and combination vectors.
- We develop a robust, practically viable procedure for *automated retraining* of classifiers with careful selection of initially unlabeled documents. We perform comprehensive experiments that evaluate our retraining procedure against state-of-the-art semi-supervised classification methods like EM-iterated Bayesian classifiers, Transductive SVMs, and Spectral Graph Transduction.

Preliminary results from this thesis have been published in [111, 113, 112, 115, 110, 114].

1.3 Overview

This thesis is organized as follows.

We describe the technical basics for this thesis in Chapter 2. These include basic classification and clustering algorithms, quality measures for classification and clustering and apriori quality estimators for classification. Furthermore we discuss feature selection and, specific to the application domain of document classification, the Bag-Of-Words model and some linguistic basics.

Related work for this thesis is discussed in Chapter 3.

Chapter 4 deals with restrictive classification and meta classification in more detail. We provide a probabilistic model for the tradeoff between *loss*, a measure for the restrictivity of a classifier, and the classification accuracy. We use this model to tune the parameter of a special instance of restrictive classifier: the ensemble based k-split meta classifier. In Chapter 5 we show how restrictive classification can be applied to eliminate "junk" documents.

In Chapter 6 we carry the concepts of restrictive meta classification, described in Chapter 4, forward to clustering. Here we introduce a technique to reduce the problem of combining cluster labels to combining class labels: the meta mapping.

Chapter 7 applies the concepts of meta classification and clustering in the context of P2P information systems.

In Chapter 8 we describe the construction of alternative feature spaces for classification and combination for two different domains: classification of Web documents and author classification. Here we apply two combination techniques: 1) restrictive meta classification, and 2) combination vectors.

We tackle the problem of an insufficient number of labeled documents in Chapter 9. We consider the automatic parameter tuning of "retraining", a semisupervised learning method that iteratively considers automatically assigned class labels.

2 Technical Basics

This chapter describes the technical basics necessary for the understanding of this thesis.[1] These include foundations of document processing and indexing, classification and clustering, feature selection and some basics from linguistics.

2.1 Preprocessing of Documents and Indexing

2.1.1 Preprocessing

Documents, which typically are strings of characters, need to be transformed into a representation suitable for automatic processing. The text is usually transformed as follows:

1. Remove HTML (or other) tags
2. Remove stopwords
3. Perform word stemming

The *stopwords* are frequent words that carry no information (i.e. pronouns, prepositions, conjunctions, etc.). By *word stemming* we mean the process of suffix removal and other translations to generate word stems. This is done to normalize morphological variants, such as "connect", "connected", "connection", and "connecting". The Porter stemmer [99] is a well-known algorithm for this task.

2.1.2 Indexing

The most commonly used document representation is the so called vector space model [104]. In the vector space model, documents are represented by vectors of words. Usually, one has a collection of documents, which is represented by a word-by-document matrix A, where each entry represents the occurrences of a word in a

[1] It is mostly based on [58, 29, 8, 84].

document, i.e.,

$$A = (a_{ik}) \quad , \tag{2.1}$$

where a_{ik} is the weight of word i in document k. This representation is called *Bag-of-Words* representation.

There are several ways of determining the weight a_{ik} of word i in document k. Most of the approaches are based on two empirical observations regarding text:

- The more times a word occurs in a document, the more relevant it is to the topic of the document.
- The more times a word occurs throughout all documents in the collection, the more poorly it discriminates between documents.

Let f_{ik} be the frequency of word i in document k, N the number of documents in the collection, M the number of words in the collection after stopword removal and word stemming, and n_i the total number of documents in the whole collection in which word i occurs. We describe 3 examples for weighting schemes that are based on these quantities.

Boolean weighting The simplest approach is to let the weight be 1 if the word occurs in the document and 0 otherwise:

$$a_{ik} = \begin{cases} 1 & \text{if } f_{ik} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Word frequency weighting Another simple approach is to use the frequency of the word in the document:

$$a_{ik} = f_{ik} \tag{2.3}$$

Alternatively the frequency can be normalized by the document length $|d_k|$:

$$a_{ik} = \frac{f_{ik}}{|d_k|} \tag{2.4}$$

tf×idf-weighting The previous two schemes do not take into account the frequency of the word throughout all documents in the collection. A well-known approach for computing word weights is the tf×idf-weighting [104], which assigns the weight to word i in document k in proportion to the number of occurrences of the word in the document, and in inverse proportion to the number of documents in the collection for which the word occurs at least once.

$$a_{ik} = f_{ik} \cdot log\left(\frac{N}{n_i}\right) \tag{2.5}$$

There are other enhanced weighting methods based, e.g., on Statistical Language Models [98, 21].

2.2 Classification

Classification (also called *Supervised Learning*) is the process of finding a set of models (or functions) which describe and distinguish data classes or concepts, for the purpose of predicting the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known). In the context of document classification our objects are the documents and we aim to automatically assign thematic labels like "Sports", "Music", or "Computer Science" to these documents.

In what follows we describe some of the algorithms for text categorization that have been proposed and evaluated in the past, but first some general notion is given: Let $\vec{d} = (d_1, \ldots, d_M)$ be the document vector to be classified and $c_1, \ldots, c_K$ the possible topics. Further assume that we have a training set consisting of N document vectors $\vec{d_1}, \ldots, \vec{d_N}$ with true class labels $y_1, \ldots, y_N$. N_j is the number of training documents for which the true class is c_j.

2.2.1 Naive Bayes

The Naive Bayes classifier is constructed by using the training data to estimate the probability of each class given the document feature values of a new instance. We use Bayes theorem to estimate the probabilities:

$$P(c_j|\vec{d}) = \frac{P(c_j)P(\vec{d}|c_j)}{P(\vec{d})} \tag{2.6}$$

The denominator in the above equation does not differ between categories and can be left out. Moreover, the naive part of such a model is the assumption of word independence, i.e., we assume that the features are conditionally independent, given the class variable. This simplifies the computations yielding

$$P(c_j|\vec{d}) \sim P(c_j) \prod_{i=1}^{M} P(d_i|c_j) \tag{2.7}$$

An estimate $\hat{P}(c_j)$ for $P(c_j)$ can be calculated from the fraction of training documents that is assigned to class c_j:

$$\hat{P}(c_j) = \frac{N_j}{N} \tag{2.8}$$

Moreover, an estimate $\hat{P}(d_i|c_j)$ for $P(d_i|c_j)$ is given by:

$$\hat{P}(d_i|c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^{M} N_{kj}} \tag{2.9}$$

where N_{ij} is the number of times word i occurred within documents from class c_j in the training set. (Here 1 is added to the numerator and M to the denominator to avoid null values. This is a variant of the so called *Laplace Smoothing*.)

Despite the fact that the assumption of conditional independence is generally not true for word appearance in documents, the Naive Bayes classifier is surprisingly effective.

2.2.2 k-Nearest-Neighbors

To classify an unknown document vector $\vec{d}$, the k-Nearest-Neighbors (kNN) algorithm ranks the document's neighbors among the training document vectors, and uses the class labels of the k most similar neighbors to predict the class of the input document. The classes of these neighbors are weighted using the similarity of each neighbor to $\vec{d}$, where similarity may be measured by, for example, the cosine similarity. The cosine similarity between two vectors $\vec{a}$ and $\vec{b}$ is defined as follows:

$$cosine(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \cdot ||\vec{b}||} \tag{2.10}$$

2.2.3 Support Vector Machines

With the usual assumption that the training and test samples are drawn from the same distribution, a hyperplane that is close to many training data points has a higher risk of misclassifying test instances compared to a hyperplane that passes through a no-man's land clear of any training instances. This is the basic intuition behind

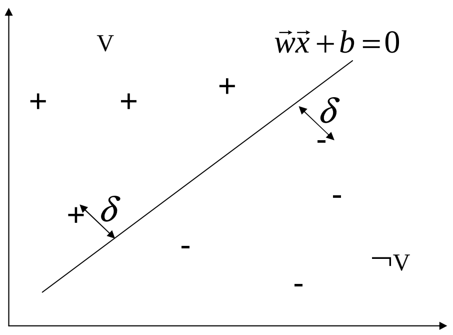

Figure 2.1: Linear SVM Classifier

support vector machines (SVMs), which are currently the most accurate classifiers for text. Linear SVMs make a binary decision by thresholding a function $\vec{w}\vec{d}+b$ (the estimated class is +1 or -1 according to whether the quantity is greater or less than 0) for a suitable vector $\vec{w}$ and constant b.

Initially, let us consider the case that the N training documents (represented as vectors) from the two classes are linearly separable by a hyperplane perpendicular to a suitable $\vec{w}$. SVM seeks a $\vec{w}$ that maximizes the distance of any training point from the hyperplane; this can be written as

$$\begin{aligned} &\text{Minimize} && \frac{1}{2}\vec{w}\cdot\vec{w} \\ &\text{subject to} && c_i(\vec{w}\cdot\vec{d_i}+b) \geq 1 \ \forall i = 1,\dots,N \end{aligned} \tag{2.11}$$

where $\vec{d_1},\dots,\vec{d_N}$ are the training document vectors and $c_1,\dots,c_N$ their corresponding classes. The optimal separator maximizes the margin δ to the nearest training points (see Figure 2.1). Since all $c_i \in \{-1,1\}$, $\vec{w}$ and b can be scaled so that for all training documents $\vec{d_i}$ we obtain:

$$c_i(\vec{w}\cdot\vec{d_i}+b) \geq 1 \tag{2.12}$$

If $\vec{d_1}$ and $\vec{d_2}$ are points touching the separator slab on opposite sides, it follows that

$$\vec{w}\cdot(\vec{d_1}-\vec{d_2}) = 2 \tag{2.13}$$

and therefore

$$\frac{\vec{w}}{\|\vec{w}\|}\cdot(\vec{d_1}-\vec{d_2}) = \frac{2}{\|\vec{w}\|} \tag{2.14}$$

The distance of any training point from the optimized hyperplane (called the *margin*) will be at least $\frac{1}{\|\vec{w}\|}$.

In real life, the classes in the training data are sometimes, but not always, separable. To handle the general case where a single hyperplane may not be able to correctly separate *all* training points, *slack* variables $\{\xi_1, \ldots, \xi_N\}$ are introduced, and Equation 2.11 is extended into

$$\begin{aligned}
\text{Minimize} \quad & \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum_i \xi_i \\
\text{subject to} \quad & c_i(\vec{w} \cdot \vec{d_i} + b) \geq 1 - \xi_i \ \forall i = 1, \ldots, N \\
& \xi_i \geq 0 \ \forall i = 1, \ldots, N
\end{aligned} \tag{2.15}$$

If $\vec{d_i}$ is misclassified, then $\xi_i \geq 0$, so $\sum_i \xi_i$ bounds from above the number of training errors, which is traded off against the margin using the tuned constant C. SVM packages [63, 7] solve the dual of Equation 2.15, involving scalars $\lambda_1, \ldots, \lambda_N$, given by

$$\begin{aligned}
\text{Maximize} \quad & \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j c_i c_j (\vec{d_i}\vec{d_j}) \\
\text{subject to} \quad & \sum_i c_i \lambda_i = 0 \\
& 0 \leq \lambda_i \leq C \ \forall i = 1, \ldots, N
\end{aligned} \tag{2.16}$$

Formula 2.16 represents a quadratic optimization problem.

2.2.4 Transductive Support Vector Machines

Unlike the inductive SVM setting, for Transductive SVM (TSVM) [64, 125], a hyperplane is computed that separates *both* training data $\vec{d_1}, \ldots, \vec{d_N}$ and (unlabeled) test data $\vec{d'_1}, \ldots, \vec{d'_k}$ with maximum margin, as illustrated in Figure 2.2 (with unlabeled data shown as plain points). For the linearly separable case this leads to the following optimization problem:

$$\begin{aligned}
\text{Minimize} \quad & \frac{1}{2}\vec{w} \cdot \vec{w} \\
\text{subject to} \quad & c_i(\vec{w} \cdot \vec{d_i} + b) \geq 1 \ \forall i = 1, \ldots, N \\
& c'_j(\vec{w} \cdot \vec{d'_j} + b) \geq 1 \ \forall j = 1, \ldots, k
\end{aligned} \tag{2.17}$$

The minimization is carried out over $\vec{w}$, b, and the $c'_j \in \{+1, -1\}$.

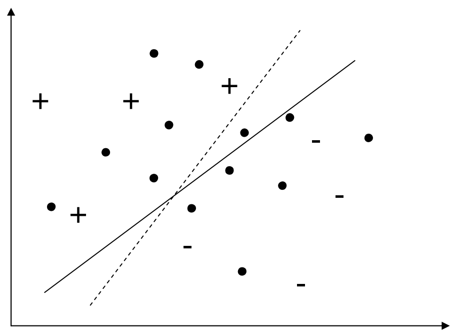

Figure 2.2: Linear SVM Classifier and Linear TSVM Classifier (dashed hyperplane)

Analogously to inductive SVM, this optimization problem can be generalized to the non-separable case, and can be formulated as a dual optimization problem.

2.2.5 The Centroid Method

We consider also a much simpler *centroid classifier* that separates the centroids of positive and negative training sets with maximum margin. The positive centroid $\vec{c}_{pos}$ and the negative centroid $\vec{c}_{neg}$ are defined as follows:

$$\vec{c}_{pos} = \frac{1}{\left|\{i|c_i \in \{+1\}\}\right|} \sum_{\{i|c_i \in \{+1\}\}} \vec{d_i} \tag{2.18}$$

$$\vec{c}_{neg} = \frac{1}{\left|\{i|c_i \in \{-1\}\}\right|} \sum_{\{i|c_i \in \{-1\}\}} \vec{d_i} \tag{2.19}$$

Obviously this method, which can be regarded as a variant of the Rocchio family of classifiers [62], is much faster than SVMs in the training phase, as its centroid and hyperplane computation are linear in the number of training documents. As for the decision phase for test documents, there is no difference between an SVM and a centroid classifier.

2.2.6 Multi-Class Classification

For multi-class classification we are given a set $\Omega = \{c_1, \ldots, c_K\}$ of $K > 2$ classes and aim to assign one of these classes to a previously unknown document. Some machine learning algorithms, such as Naive Bayes or kNN, described in Section 2.2.1

	$\Omega_1 = \{c_1\}$	$\Omega_1 = \{c_2\}$	$\Omega_1 = \{c_3\}$	$\Omega_1 = \{c_4\}$	$\Omega_1 = \{c_1, c_2\}$	$\Omega_1 = \{c_1, c_3\}$	$\Omega_1 = \{c_1, c_4\}$
c_1	1	0	0	0	1	1	1
c_2	0	1	0	0	1	0	0
c_3	0	0	1	0	0	1	0
c_4	0	0	0	1	0	0	1

Figure 2.3: Code Matrix (C_{ij}) for four Classes $\{c_1, \dots, c_4\}$

and Section 2.2.2, can handle the multiclass case. For other algorithms, such as SVM (Section 2.2.3), a direct extension to the multiclass case may be problematic. Typically, in such cases, the multiclass problem is reduced to multiple binary classification problems that can be solved separately [11, 86]. Below we describe two methods to reduce multiclass classification to binary classification.

One vs. All

One of the simplest multiclass classification schemes built on top of real-valued binary classifiers is to train K different binary classifiers, each one trained to distinguish the examples in a single class c_i from the examples in all remaining classes $\Omega \backslash \{c_i\}$. To classify a new example, the K classifiers are run, and the classifier that outputs the largest (most positive) value is chosen. For instance, for SVM, we can choose the distance of the test example from the hyperplane as an output value.

Error Correcting Output Codes (ECOC)

For *error correcting output codes* (ECOC) each classifier discriminates between two possible compound classes $\Omega_1, \Omega_2 \subset \Omega$ with $\Omega_1, \Omega_2 \neq \emptyset$, $\Omega_1 \cap \Omega_2 = \emptyset$, and $\Omega_1 \cup \Omega_2 = \Omega$. The number of possible different splits $\{\Omega_1, \Omega_2\}$ of a set Ω of K classes is

$$S = 2^{(K-1)} - 1 \ . \tag{2.20}$$

For example a set $\{c_1, \dots, c_4\}$ of four classes can be split in $S = 7$ ways into $\Omega_1 = \{c_1\}$, $\{c_2\}$, $\{c_3\}$, $\{c_4\}$, $\{c_1, c_2\}$ $\{c_1, c_3\}$, or $\{c_1, c_4\}$, and corresponding sets $\Omega_2 = \Omega \backslash \Omega_1$.

The (ideal) classifier assignments can be represented as a binary code matrix C_{ij} with $C_{ij} = 1$ if $c_i \in \Omega_1$ for split number j, $C_{ij} = 0$ otherwise. Figure 2.3 shows the code matrix for four classes. Suppose that the classifiers output binary labels $(l_1, \dots, l_S)$ for a given test example. The class with the shortest Hamming distance between the classifier outputs and the codewords for the classes is chosen as the label for the test example. For the above example, let $(l_1, \ldots, l_S) = (0, 1, 1, 0, 0, 1, 0)$. The Hamming distances are 5, 4, 1 and 5 respectively; hence label c_3 is assigned to the test example.

To avoid a combinatorial explosion of the code matrix, different approaches that choose an appropriate subset of splittings are suggested [40, 105]; one of the simplest methods is a random choice of splittings.

2.3 Performance Measures and Estimators for Classification

2.3.1 Performance Measures

In this section we consider the issue of measuring the performance of the classifiers. Many measures have been used, each of which has been designed to evaluate some aspect of the categorization performance of a system. We describe some of the measures that are widely used in the literature.

A common approach for multi-class categorization is to break the task into disjoint binary categorization problems. For each category and each document one determines whether the document belongs to the category or not. When evaluating the performance of the classifiers, four quantities are of interest for each category:

- a - the number of documents *correctly assigned* to this category.
- b - the number of documents *incorrectly assigned* to this category.
- c - the number of documents *incorrectly rejected* from this category.
- d - the number of documents *correctly rejected* from this category.

From these quantities, we define the following performance measures:

$$recall = \frac{a}{a+c} \tag{2.21}$$

$$precision = \frac{a}{a+b} \tag{2.22}$$

$$accuracy = \frac{a+d}{a+b+c+d} \tag{2.23}$$

$$error = \frac{b+c}{a+b+c+d} \tag{2.24}$$

An evaluation criterion that combines recall and precision is the *F-measure* (weighted harmonic mean):

$$F_\beta = \frac{(\beta^2+1) * precision * recall}{\beta^2 + precision + recall} \tag{2.25}$$

where β is a parameter allowing different weighting of recall and precision (usually $\beta = 1$).

Micro- and Macro-Averaging For evaluating average performance across categories, there are two conventional methods, namely *macro-averaging* and *micro-averaging*. Macro-averaged performance scores are determined by first computing the performance measures per category and then averaging these to compute the global mean. Micro-average performance scores are determined by first computing the totals a, b, c, and d for all categories and then use these totals to compute the performance measures. Micro-averaging gives equal weight to every document, while macro-averaging gives equal weight to each category.

Sometimes micro- and macro-averaging is accomplished on a set of class *pairs*.

2.3.2 Estimators for Classification

Estimating classifier accuracy or other performance measures is important in that it allows to evaluate how accurately a given classifier will label future data, that is, data on which the classifier has not been trained. Accuracy estimates can help in the comparison of different classifiers.

Using training data to derive a classifier and then to estimate the accuracy of the classifier can result in misleading overoptimistic estimates due to over-fitting of the learning algorithm (or model) to the training data.

The most widely used technique for empirically estimating the classifier quality is *cross-validation* [84] on a set of independent data samples with known topic memberships (aka. class labels). The partitioning is systematically varied by dividing the overall training data into k groups and investigating each of the k choices for using one group as test data and the other $k-1$ groups for training; the empirical results are finally averaged over all choices. An important special case is *leave-one-out validation* [84]. Here the N documents of a data collection are divided by the ratio $(N-1):1$. Leave-one-out prediction is more accurate than prediction based on cross-validation with smaller k but requires training the classifier N times, unless special properties of the classifier's underlying model could be exploited.

Another method of estimating classifier accuracy is *bootstrapping*, which samples the given training instances uniformly *with replacement*.

The use of such techniques to estimate classifier accuracy increases the overall computation time, yet is useful for selecting among several classifiers.

2.4 Clustering

Unlike classification, which analyzes class-labeled data objects, clustering analyzes data objects without consulting a known class label (*Unsupervised Learning*); i.e. for class labels are not known and training data are not available. Clustering can be used to generate such labels. The objects are clustered or grouped based on the principle of maximizing the intracluster similarity and minimizing the intercluster similarity. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Each cluster that is formed can be viewed as a class of objects, from which rules can be derived. Clustering can also facilitate *taxonomy formation*, that is, the organization of observations into a hierarchy of classes that group similar subclasses together. In our context, we aim to cluster documents into groups of thematically related documents.

2.4.1 Families of Clustering Methods

There exist a large number of clustering algorithms in the literature. The choice of clustering algorithm depends both on the type of data available and on the particular purpose and application.

In general, major clustering methods can be classified into the following categories.

Partitioning methods Given a database of n objects or data tuples, a partitioning method such as k-means [83] constructs k partitions of the data, where each partition represents a cluster and $k \leq n$. That is, it classifies the data into k groups, which together satisfy the following requirements: 1) each group must contain at least one object, and 2) each object must belong to exactly one group. Notice that the second requirement can be relaxed in some fuzzy partitioning techniques.

Hierarchical methods A hierarchical method such as AGNES, or DIANA [70] creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*, based on how the hierarchical decomposition is formed. The *agglomerative approach*, also called *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds. The *divisive approach*, also called the *top-down* approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

Density-based methods The general idea of density-based methods such as DB-SCAN, OPTICS, or DENCLUE [46, 14, 60] is to continue growing the given cluster as long as the density (number of object or data points) in the "neighborhood" exceeds some threshold; that is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

Grid-based methods Grid-based methods such as STING, or CLIQUE [127, 9] quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure (i.e., on the quantized space). The main advantage is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

Model-based methods Model-based methods such as COBWEB, or CLASSIT [49, 54] hypothesize a model for each of the clusters and find the best fit of the data to the given model. A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking "noise" or outliers into account and thus yielding robust clustering methods.

In what follows we will present an example of a partitioning clustering method.

2.4.2 Partitioning Clustering: k-Means

The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured with regard to the mean value of the objects in a cluster.

The k-means algorithm proceeds as follows. First, it randomly selects k objects, each of which initially represents a cluster centroid. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster centroid. It then computes the new mean for each cluster. This process is iterated until the objective function converges. Typically, the *squared error criterion* is used as an objective function, defined as

$$E = \sum_{i=1}^{k} \sum_{\vec{p} \in C_i} \|\vec{p} - \vec{m}_i\|^2 \qquad (2.26)$$

where E is the sum of squared errors for all objects, $\vec{p}$ is the point in space representing a given object, and $\vec{m}_i$ is the centroid of cluster C_i. This criterion tries to make the resulting k clusters as compact as possible.

2.5 Dimensionality Reduction

A central problem in statistical text classification is the high dimensionality of the feature space. There exists one dimension for each unique word found in the collection of documents, typically many thousands. Some classification and clustering techniques cannot deal with such a large feature set. Hence there is a need for a reduction of the original feature set, which is commonly known as dimensionality reduction in the pattern recognition literature. Most of the dimensionality reduction approaches can be classified into one of two categories: feature selection or re-parameterization.

2.5.1 Feature Selection

Feature selection attempts to remove non-informative words from documents in order to improve categorization effectiveness and reduce computational complexity. In [131] a thorough evaluation of the five feature selection methods, Document Frequency Thresholding, Information Gain, χ^2-statistic, Mutual Information and Term Strength is given. Below we give a short description of four methods:

Document Frequency Thresholding The document frequency *df* for a word is the number of documents in which the word occurs. In Document Frequency Thresholding one computes the document frequency for each word in the training corpus and removes those words whose document frequency is less than som predetermined threshold. The basic assumption is that rare words are either non-informative for category prediction, or not influential in global performance.

Information Gain Information Gain (IG) measures the number of bits of information obtained for category prediction by knowing the presence or absence of a word in a document. Let $\{c_1, \dots, c_K\}$ denote the set of possible categories. The information gain of a word w is defined to be:

$$IG(w) = -\sum_{j=1}^{K} P(c_j) log P(c_j) + P(w) \sum_{j=1}^{K} P(c_j|w) log P(c_j|w) + P(\bar{w}) \sum_{j=1}^{K} P(c_j|\bar{w}) log P(c_j|\bar{w}) \tag{2.27}$$

Here $P(c_j)$ can be estimated from the fraction of documents in the total collection that belongs to class c_j and $P(w)$ from the fraction of documents in which the word w occurs. Moreover, $P(c_j|w)$ can be computed as the fraction of documents from class c_j among the documents that have at least one occurence of word w, and $P(c_j|\bar{w})$ as the fraction of documents from class c_j among the documents that do not contain word w.

The Information Gain is computed for each word of the training set, and the words whose information gain is less than some predetermined threshold are removed or the m words with the highest IG value are chosen.

χ^2-Statistics The χ^2-statistics measures the lack of independence between word w and class c_j. It is given by:

$$\chi^2(w, c_j) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \tag{2.28}$$

Here A is the number of documents from class c_j that contains word w, B is the number of documents that contains w but does not belong to class c_j, C is the number of documents from class c_j that does not contain word w, and D is the number of documents that neither belongs to class c_j nor contains word w. N is still the total number of documents.

Two different measures can be computed based on the χ^2-statistics:

$$\chi^2(w) = \sum_{j=1}^{K} P(c_j) \chi^2(w, c_j) \tag{2.29}$$

or

$$\chi^2_{max}(w) = \max_j \chi^2(w, c_j) \tag{2.30}$$

Mutual Information Mutual Information (MI) is a criterion commonly used in statistical language modeling of word associations and related applications. The mutual information criterion between a word w and a class c_j is defined to be:

$$MI(w, c_j) = \log \frac{P(w \wedge c_j)}{P(w)P(c_j)} \tag{2.31}$$

With A, B, C, D, and N, as defined above, it is estimated using:

$$MI(w, c_j) \approx \log \frac{A \times N}{(A + C) \times (A + B)} \tag{2.32}$$

$MI(w, c_j)$ has a natural value of zero if w and c_j are independent.

2.5.2 Re-Parameterization

Re-parameterization is the process of constructing new features as combinations or transformations of the original features. In this section we describe one such approach: *Latent Semantic Indexing* (LSI) [22].

LSI is based on the assumption that there is some underlying or latent structure in the pattern of word usage across documents, and that statistical techniques can be used to estimate this structure. LSI uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis. In what follows we describe the mathematics underlying the particular model of the latent structure; the singular value decomposition.

Assume that we have an $M \times N$ word-by-documents matrix $\mathbf{A}$, where M is the number of words, and N the number of documents. The singular value decomposition of $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.33}$$

where $\mathbf{U}$ $(M \times R)$ and $\mathbf{V}$ $(N \times R)$ have orthonormal columns and $\mathbf{\Sigma}$ $(R \times R)$ is the diagonal matrix of singular values (i.e., eigenvalues of $\mathbf{A}\mathbf{A}^T$). $R \leq min(M, N)$ is the rank of $\mathbf{A}$. If the singular values of $\mathbf{\Sigma}$ are ordered by size, the K largest may be kept and the remaining smaller ones set to zero (with their corresponding eigenvectors in $\mathbf{U}$ and $\mathbf{V}^T$ set to zero, too). The product of the resulting matrices is a matrix $\mathbf{A}_K$ which is an approximation to $\mathbf{A}$ with rank K.

$$\mathbf{A}_K = \mathbf{U}_K\mathbf{\Sigma}_K{\mathbf{V}_K}^T \tag{2.34}$$

where $\mathbf{\Sigma}_K$ $(K \times K)$ is obtained by deleting the zero columns of $\mathbf{\Sigma}$, and $\mathbf{U}_K$ $(M \times K)$ and $\mathbf{V}_K$ $(N \times K)$ are obtained by deleting the corresponding rows and columns of $\mathbf{U}$ and $\mathbf{V}$. $\mathbf{A}_K$ is the rank-K matrix which minimizes $||\mathbf{A} - \mathbf{A}_K||_2$.

$\mathbf{A}_K$ in some sense captures most of the underlying structure in $\mathbf{A}$, yet at the same time removes the noise or variability in word usage. Since the number of dimensions K is much smaller than the number of unique words M, minor differences in terminology will be ignored. Words which occur in similar documents may be near each other in the K-dimensional space even if they never co-occur in the same document. Moreover, documents that do not share any words with each other may turn out to be similar. Using LSI, a document vector $\vec{d}$ can be represented in a K-dimensional space using the following transformation:

$$\vec{\hat{d}} = \vec{d}^T \mathbf{U}_K \mathbf{\Sigma}_K^{-1} \tag{2.35}$$

2.6 Linguistic Basics

2.6.1 Parts of Speech and Morphology

Linguists group the words of a language into classes (sets) which show similar syntactic behavior, and often a typical semantic type. These word classes are called *parts of speech* (POS). Three important parts of speech are *noun*, *verb*, and *adjective*. *Nouns* typically refer to people, animals, concepts and things. The prototypical verb is used to express the action in a sentence. *Adjectives* describe properties of nouns.

Word classes are normally divided into two types. The *open* or *lexical categories* are the ones like nouns, verbs, and adjectives which have a large number of members, and to which new words are commonly added. The *closed* or *functional categories* are categories such as prepositions and determiners (containing words like *of*, *on*, *the*, *a*) which have only a few members, and the members of which normally have a clear grammatical use.

Traditional systems of parts of speech distinguish about 8 categories, but corpus linguists normally want to use more fine-grained classifications of word classes. There are well-established sets of abbreviations for naming these classes, usually referred to as POS *tags*. A particularly widely known tag set is, e.g., the Penn-Treebank-Tagset [85].

Word categories are systematically related by morphological processes such as a formation of the *plural* form (e.g., *dog-s*) to the *singular* form of the noun (*dog*).

2.6.2 Phrase Structure

Words do not occur in just any order. Languages have constraints on *word order*. But is also the case that the words in a sentence are not simply combined as a sequence of parts of speech. Instead, words are organized into *phrases*, groupings of words that are combined into one unit. *Syntax* is the study of the regularities and constraints of word order and phrase structure.

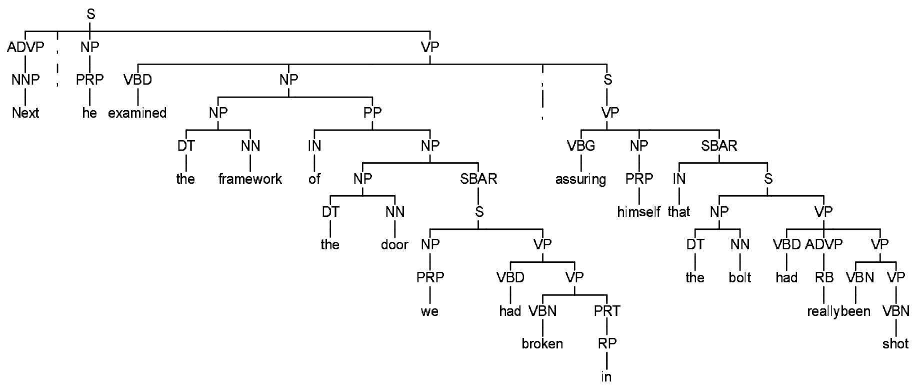
S
ADVP , NP VP
NNP , PRP VBD NP , S
Next he examined NP PP , VP
DT NN IN NP VBG NP SBAR
the framework of NP SBAR assuring PRP IN S
DT NN S himself that NP VP
the door NP VP DT NN VBD ADVP VP
PRP VBD VP the bolt had RB VBN VP
we had VBN PRT really been VBN
broken RP shot
in

Figure 2.4: PCFG-Tree

One fundamental idea is that certain groupings of words behave as *constituents*. Constituents can be detected by their being able to occur in various positions, and showing uniform syntactic possibilities for expansion.

As an example consider the sentence *Next, he examined the framework of the door we had broken in, assuring himself that the bolt had really been shot.* and its syntax tree representation in Figure 2.4. In particular, consider the part *he examined the framework*. This part is a constituent of the sentence with sub-constituents, e.g. *"the door"*. The sub-constituents can change their positions inside the bigger constituent. Just considering that specific part, *he examined the framework* has the same meaning as *the framework he examined*.

Formally syntax trees are generated by a probabilistic context-free grammar (PCFG) as follows: Consider a set Σ of tags for linguistic corpus annotation (e.g., the Penn-Treebank-Tagset [85]). Let s be a sentence and $T_s := (V, E, \sigma)$ an ordered tree with a set of nodes V, a set of edges E and a labeling function $\sigma : V \to \Sigma$, that assigns a label $l \in \Sigma$ to each node of the tree. We call T_s the syntax tree of sentence s. A PCFG is a contextfree grammar enriched by transition probabilities for each rewriting rule ([84]). For example, consider Figure 2.4. There, the sentence *Next, he examined the framework of the door we had broken in, assuring himself that the bolt had really been shot.* is represented as a syntax tree. The leaves of the tree represent the words themselves, i.e. terminal symbols, where the higher nodes represent the PCFG Tags, i.e., non terminal symbols. Non-terminals can be subdivided into other non-terminals or terminals, e.g. NP (a noun phrase) into DT (determiner, an article) and NN (a noun in singular case) and NN into "framework".

In the following we briefly mention some of the major phrase types:

Noun phrases A noun is usually embedded in a *noun phrase* (NP), a syntactic unit of the sentence in which information about the noun is gathered. The noun is the *head* of the noun phrase, the central constituent that determines the syntactic character of the phrase. Noun phrases are usually the *arguments* of verbs, the participants of the action, activity or state described by the verb. Noun phrases normally consist of an optional determiner, zero or more adjective phrases, a noun head and the modifiers, with the constituents appearing in that order. Here is an example of a large noun phrase: *The homeless old man in the park that I tried to help yesterday.*

Prepositional phrases *Prepositional phrases* (PPs) are headed by a preposition and contain a noun phrase complement. They can appear within all the other major phrase types. They are particular common in noun phrases and verb phrases where they usually express spatial and temporal locations and other attributes.

Verb phrases Analogous to the way nouns head noun phrases, the verb is the head of the *verb phrase* (VP). In general, the verb phrase organizes all elements of the sentence that depend syntactically of the verb (except that in most syntactic theories the verb phrase does not contain the subject noun phrase). Here is an example for a verb phrase: "*Getting in school on time* was a struggle."

Adjective phrases Complex *adjective phrases* (APs) are less common, but encompass examples like "She is *very sure of herself.*" or, "He seemed a man who was *quite certain to succeed.*".

3 Related Work

3.1 Meta Classification and Clustering

There is a plethora of work on text document classification using a variety of probabilistic and discriminative models [29]. The emphasis of this body of work has been on the mathematical and algorithmic aspects, and the engineering aspects of how to cope with tradeoffs and how to tune a classifier with regard to properties of the training data and, most importantly, specific application goals have been largely neglected (exceptions being, e.g., [23, 35, 126], which address different settings and are only marginally related to our work, however).

The machine learning literature has studied a variety of ensemble based meta methods such as bagging, stacking, or boosting [26, 129, 82, 52, 77], and also combinations of heterogeneous learners (e.g., [132]). For bagging, an ensemble consists of classifiers built on bootstrap replicates of the training set. The classifiers outputs are combined by the plurality vote. For stacking, multiple classifiers are trained on parts of the training set and evaluated on the remaining training documents. The outputs of the classifiers are used as feature values for training a new classifier (stacked generalization). Boosting can be viewed as a model averaging method. Here a succession of models is built, each one trained on a data set in which the points misclassified by the previous model are given more weight.

Our notion of a meta method is closest to bagging (see, e.g., [26]). To our knowledge none of the prior work on bagging and related techniques has considered the parameter tuning of such methods towards application-specific quality goals.

The approach of intentionally splitting a training set for meta learning has been investigated by [31]. However, that work has focused on the efficiency versus accuracy tradeoff; so the improvements in efficiency were achieved at the expense of reduced accuracy. In contrast, our approach preserves and even improves high accuracy, and the measure that we are trading this for is document loss. The notion of loss in a ternary decision model, on the other hand, has not received wide attention. The recent paper [106] studied the accuracy-loss tradeoff in a ROC curve model (for a recommender system), but has not looked at how to systematically engineer and tune methods for judicious application choices regarding this tradeoff.

For SVM classifiers some isolated tuning issues have been considered in the literature. The popular SVM Light software package [63] provides various kinds of thresholds and variations of SVM training (e.g., SVM regression, transductive SVMs, etc.), but there is no systematic discussion of how to adjust these tuning knobs for a given application. [25] have proposed to introduce a bias for the separating hyperplane towards negative training samples, and advocated that this is beneficial when the number of positive training samples is very low. To our knowledge, these techniques were, up to now, not considered in the context of restrictive classification and junk reduction.

There is recent work on combining multiple clustering methods in an ensemble learning manner, using consensus functions for clusterings based on information theoretic measures [118], constructing a co-association matrix and performing hierarchical clustering on this matrix [51], combining clusterings pair-wise and iteratively [41], using graph partitioning methods [48], or combining clusterings on different subspaces of a given feature space [121]. Neither of these papers considers restrictive methods where documents may be completely left out and are not assigned to any cluster; we believe that this is crucial for aiming at very high precision. Also, none of the prior work provides analytical estimation models, which is crucial for understanding why such methods work. Finally, our application context is broader and combines meta clustering with other techniques like supervised classification, and we present much more comprehensive application-oriented experimental results with real-life datasets.

3.2 Alternative Document Representations

There is considerable prior work about alternatives to the Bag-Of-Words approach for document classification [89]. These include: using Part-Of-Speech (POS) tags (”verbs”, ”nouns”, ”adjectives”, etc.) [95] either for disambiguation or for feature selection, using a thesaurus like Wordnet [47] for feature construction [107, 102], and feature selection based on statistical measures like Mutual Information or Information Gain [131]. N-grams of characters are popular for distinguishing different languages [28, 18]; also word based n-grams and phrases were examined for the text classification task [119, 79].

In [117] sentiment classification is performed by a rule based approach using manually chosen features for a very specific classification task. In [122] reviews are classified as "positive" or "negative" using a semantic orientation measure for the phrases in the documents based on an information theoretic measure between these phrases and the manually chosen key words "excellent" vs. "poor". This measure was computed on a set of documents obtained by querying a search engine. More general manually chosen linguistic structures are used in [128] to recognize opinions in newspaper articles. In [95] various approaches using POS tags, unigrams and bigrams are studied for the classification of movie reviews. Here the best performance was obtained by a simple unigram approach. The identification of unique users among a set of online pseudonyms using features such as simple words, misspellings, punctuation etc., is described in [92]. Various techniques exist to tackle the problem of spam recognition: besides manually engineered methods like keyword filters, source blacklists, signature blacklists, etc., also machine learning methods, using mostly Bag-Of-Words features, are applied [42, 13, 56].

The problem of authorship attribution is also different from the classical topic based classification task. Here, stylometric features may become important [61]. Baayen et. al. [15] show the occurrence of some kind of "stylistic fingerprint" for authors by considering a text corpus produced by student writers of different age and education level. They use the most frequent function words and apply principal component analysis (PCA) as well as linear discriminant analysis (LDA).

Diederich et al. [39] present a study on authorship attribution with Support Vector Machines. Their feature set consists of "full word forms" (in fact Bag-Of-Words) and so called tagwords, a combination of function words and grammatical information. Here, simple Bag-Of-Words outperforms their combination techniques with more enhanced linguistic features, in contrast to our combination vectors and meta methods.

In [16], Baayen et al. present a methodological study on the usefulness of stylometry-based features. They investigate features related to our writing style technique, taking grammatical rewriting rules derived from syntax trees into account. De Vel's work [36] deals with the exploration of style based features for identification of email authors. They use features such as style markers (average sentence or word length, total number of function words, vocabulary richness, etc.) and structural attributes (availability of signatures, number of attachments, etc.).

There are also several alternative learning paradigms for authorship attribution, e.g., Khmelev and Tweedie [71] considering learning models for authorship attribution tasks using Markov chains of characters, or Oakes [93] using a kind of swarm intelligence simulation technique called Ant Colony Optimization.

Combination vectors are used for authorship attribution (e.g. [124, 73, 39]), but neither explicit component weighting nor normalization are considered. The machine learning literature has studied a variety of meta methods such as bagging, stacking, or boosting [26, 129, 82, 52], and also combinations of heterogeneous learners (e.g., [132]). But, to our knowledge, meta classification was not applied in the context of authorship recognition.

There is a variety of work on Web document classification based on different document representations. These representations are based on text from neighbors in the Web graph [30], classes from neighbors (using an iterative labeling process for neighbors with apriori unknown classes) [30], link similarity [68, 33, 50], anchor texts of neighbors [55], HTML structures (such as headings and paragraphs) [53], etc. Typically one enhanced representation is combined with a simple Bag-of-Words model, see e.g. [68, 33, 27], but no restrictive meta classification has been applied.

3.3 Distributed Learning

Algorithms for distributed clustering are described in [69, 81], but here data samples (i.e., in our context, documents) must be provided to a central server, making these solutions inconsistent with our requirements. The distributed execution of k-means was discussed in [38]. However, this method requires multiple iterations that must be synchronized among the peers and causes a considerable amount of coordination overhead. Privacy-preserving distributed classification and clustering were also addressed in the prior literature: in [123] a distributed Naive Bayes classifier is computed; in [87] the parameters of local generative models are transmitted to a central site and combined, but not in a decentralized and restrictive peer-to-peer manner.

3.4 Semi-Supervised Classification

There is a considerable prior of work on classification using unlabeled data (also called semi-supervised learning), see [108] for an overview. Naive Retraining where new documents with highest classification confidence are iteratively added to the training set, is, e.g., described in [29]; but these methods perform often worse than the underlying base learning method. A more advanced EM (Expectation Maximization) based variant for Bayesian Classifiers is proposed in [91] and applied to text classification. For Transductive SVM [64, 125] and Semi-Supervised SVM [19] unlabeled samples are taken into account (opposite to standard SVM) in a modified optimization problem (standard SVM cannot use unlabeled samples at all). Co-training [24] splits the feature space into conditionally independent dimensions and performs retraining on the corresponding classifiers. Recent graph-based semi-supervised learning algorithms work by formulating the assumption that ”nearby” points, and points in the same structure should have similar labels [74, 133, 67]. In [20] semi-supervised learning is combined with ensemble classification methods. An approach for the case that only positive (and no negative) training data plus unlabeled data are available is described in [78]. In [12] semi-supervised learning is used for text summarization; in [134] a retraining method with user feedback as a stopping criterion is used for image retrieval. However, to our knowledge, none of these methods deals with the problem of automatically tuning their parameters.

The issue of asymmetric distribution of documents among different classes is addressed, e.g., in [75, 25, 57], and the problem of automated parameter tuning has been considered in the field of machine learning, e.g., in [72], but, to our knowledge, not in the context of retraining.

4 Restrictive Meta Classification

In this chapter we investigate restrictive forms of classifiers and we develop meta methods that combine the results of different classifiers. These techniques tend to improve accuracy at the expense of document loss. We develop estimators that help to predict the accuracy and loss for a given setting of the methods' tuning parameters, and a methodology for efficiently deriving a setting that meets the application's goals.

4.1 Making Classifiers Restrictive and Tunable

The idea of restrictive classification is to avoid making a decision about a test document at all if that decision can be made only with relatively low confidence. So out of a given set of unlabeled data U, our method chooses a subset S of documents that are either accepted or rejected for the given topic label, and abstains on the documents in $U - S$. The quality measures precision, recall, F1, accuracy, and error are computed on the subset S, and we call the ratio $|U - S|/|U|$ the document *loss*.

We can use confidence measures to make simple methods restrictive. For SVMs or the Centroid method a natural confidence measure is the distance of a test document vector from the separating hyperplane. So we can tune these methods by requiring that accepted or rejected documents have a distance above some threshold, and abstain otherwise. The threshold is our tuning parameter. This approach is illustrated in Figure 4.1: all test documents that fall in the shaded region are dismissed.

Given an application-acceptable loss of L percent, we can make a classifier restrictive by dismissing the L percent of the test documents with the lowest confidence values.

4.2 Restrictive Meta Classifiers

For meta classification we are given a set $V = \{v_1, \ldots, v_k\}$ of k binary classifiers with results $R(v_i, d)$ in $\{+1, -1\}$ for a document d, namely, +1 if d is accepted for the given topic by v_i, and -1 if d is rejected. We can combine these results into a meta result: $Meta(d) = Meta(R(v_1, d), \ldots, R(v_k, d))$ in $\{+1, -1, 0\}$ where 0 means

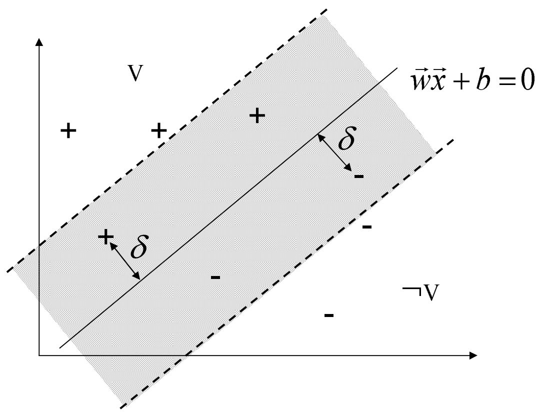

Figure 4.1: Restrictive SVM Classifier

abstention. A family of such meta methods is the linear classifier combination with thresholding [111]. Given thresholds t_1 and t_2, with $t_1 > t_2$, and weights $w(v_i)$ for the k underlying classifiers we compute $Meta(d)$ as follows:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

This meta classifier family has some important special cases, depending on the choice of the weights and thresholds:

1. voting [26]: Meta returns the result of the majority of the classifiers.
2. unanimous decision: if all classifier give us the same result (either +1 or -1), Meta returns this result, 0 otherwise.
3. weighted averaging [126]: Meta weighs the classifiers by using some predetermined quality estimator, e.g., a leave-one-out estimator for each v_i.

The restrictive and tunable behavior is achieved by the choice of the thresholds: we dismiss the documents where the linear result combination lies between t_1 and t_2. In the rest of the work we will consider only the unanimous-decision meta classifier as the simplest of the above cases in order to demonstrate the feasibility of our approach. The approach itself carries over to more sophisticated instantiations of the meta classifier framework.

4.3 k-split Meta Classifier

One possibility to obtain a set V of k different classifiers is to split the training set T_0 into k disjoint subsets $T_1, \dots, T_k$ and build one classifier v_i for each set T_i. Then we can easily construct a meta classifier, coined the k-split meta classifier.

Why would such a partitioning of the training data be useful at all? There are two aspects to consider:

- First, it may help to make the overall classification procedure to become more robust and reduce its error. The rationale for this is that subsets of T_0 may be good enough for effectively training a classifier and that the consensus or averaging step over all classifiers then helps to counteract possible overfitting effects and thus makes the meta classifier more robust. Obviously, there are limitations to this desirable but not always achievable effect; we would expect some optimal choice of k beyond which further splitting becomes detrimental (and with very small training sets the optimal k would often be 1, i.e., not splitting at all).
- Second, the time for training a classifier often depends in some super-linear way on the cardinality of the training set T (e.g., more than quadratic with SVM). So we can improve the training efficiency by learning with subsets of T_0. This is particularly intriguing for applications that require interactive re-training such as personalized focused crawlers where user feedback after some initial results can help to calibrate the focus.

A similar approach has been studied by Chan [31], but his classifiers were not restrictive and tunable, so that he obtained efficiency gains at the expense of significantly increasing the classification error. In contrast, our method trades efficiency for loss, but keeps accuracy high or even improves it.

The naturally arising next question is how to search the tuning parameter space for the most appropriate value of k. Before we turn to this issue in Section 4.4.4, we first discuss, in Section 4.4, how to estimate the accuracy and loss for a fixed setting of k and the other tuning parameters.

A natural alternative to splitting the training set into disjoint partitions would be to allow overlapping partitions and not necessarily use all training documents. This could be easily implemented using *random sampling* to create a training partition, where the number k of partitions and their size m in terms of training documents are tuning parameters. Note that the sampling procedure is allowed to select duplicates in different partitions, hence the name random resampling. This method has the same need for parameter tuning that the k-split approach faces. We will study random resampling as a competitor to k-split meta classification in our experiments.

4.4 Estimators for Accuracy and Loss

4.4.1 Estimators for Single Classifiers

For tuning the use of a classifier in an application it is desirable to have apriori estimators for the accuracy and loss of the classification method given its training data. While such estimators entirely based on analytical models are conceivable, our experience has led us to believe that some degree of empirical estimation is inevitable for sufficiently accurate predictions, at least as a baseline upon which analytical reasoning can build. We will see that good estimators for the k-split meta classifiers can be constructed without (repeatedly) performing time-consuming leave-one-out estimation on the full training set. Likewise, for restrictive variants of the base classifiers SVM and Centroid we can avoid having to fully recompute (with repeated invocation of the training procedure) the leave-one-out estimator for every setting of the threshold parameters.

SVM requires an initial full-fledged leave-one-out estimator which involves n times retraining, where n is the cardinality of T_0, or at least an initial cross-validation with m partitions of size n/m and m times retraining. For the Centroid method a more efficient technique is feasible: we compute the two centroids that result from all documents in T_0. For each leave-one-out choice we incrementally "subtract" the left-out test document yielding the adjusted centroids and the adjusted hyperplane in time $O(1)$ and classify the document. So just like the entire training procedure for Centroid has run-time $O(n)$, the complete leave-one-out estimator (over all n choices) can be carried out in the same $O(n)$ time.

For the k-split meta methods we will need to estimate the accuracy of each of the underlying classifiers trained with subsets $T_1, \ldots, T_k$. We can exploit this situation by estimating the accuracy $acc(T_i)$ via cross-validation on the complementary subsets $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_k$. In the experiments we refer to this estimator as the *CV estimator*.

We notice that this step does not require expensive leave-one-out estimations, and it is carried out over training sets that are significantly smaller than T_0. However, depending on the application situation we may like to avoid having to re-run the estimations on all candidate choices of k, as this would still require the training and cross-validation of k classifiers for every value of k. Rather we would prefer analytically reasoning, with very low computational cost, about the quality degradation that we are likely to see when training with some subset T_i instead of T_0. Our rationale is the following: the accuracy for training with T_i, $acc(T_i)$, should really be the same as for T_0 itself, $acc(T_0)$, if T_i is still a representative sample for the same stochastic feature distribution for which T_0 is our baseline sample; conversely, if the distribution in T_i deviates significantly from that of T_0 we should see some degradation in accuracy. Deviations $diff$ between probability distributions can be measured by the Kullback-Leibler divergence (relative entropy) $KL(T_i, T_0)$.

We performed extensive experiments to study the viability of such heuristic estimators, and also looked at variations and other entropy- or χ^2-based measures for deviation. It turned out that the correlation coefficient between our $diff$ metric and the resulting accuracy was consistently above 0.9 for a number of experiments with k-splits on $Newsgroups$ and $Reuters$ datasets; so $acc(T_i)$ does indeed deteriorate approximately linearly with increasing $KL(T_i, T_0)$. Our estimator for $acc(T_i)$ is based on a simple linear regression using two (or possibly more) data points for acc and KL derived from two (or more) k-split partitionings. This way we fit the coefficients a and b in the following equation:

$$acc(T_i) = a \cdot KL(T_i, T_0) + b \tag{4.2}$$

Note that this procedure requires $acc(T_i)$ estimators, using the cross-validation technique outlined above, for only two choices of k, and we can choose relatively large values of k (e.g., 10 and 20) so that the training and cross-validation of k classifiers on relatively small training sets is fairly inexpensive. Further note that this computation is carried out outside the actual search procedure for the best possible k, and that we do not need to be particularly judicious about choosing our two sample points for k. We refer to this alternative estimator as the *KL estimator* in our experiments.

The KL divergence itself is estimated by

$$KL(T_i, T_0) = \sum_j f_j(T_i) \cdot log_2 \frac{f_j(T_i)}{f_j(T_0)} \tag{4.3}$$

where $f_j(S)$ is the relative frequency of documents in document set S that contain feature j (i.e., a word stem). This computation is linear in the cardinality of the training set T_i. For a robust estimate we actually average the KL values over all subsets T_i of a given k-split partitioning.

One complication that arises with this approach is that positive and negative training examples follow radically different distributions, and we have to make sure that our KL-based distance measure captures this. To this end we actually compute the KL divergence for positive and negative samples separately and take their maximum for the scaling factor in the accuracy prediction:

$$diff(T_i, T_0) = max\{KL_{pos}(T_i, T_0), KL_{neg}(T_i, T_0)\} \tag{4.4}$$

$$acc(T_i) = a \cdot diff(T_i, T_0) + b \tag{4.5}$$

Figures 4.2 and 4.3 illustrate the viability of this analytic approximation technique.

Test	pos. samples	neg. samples	k	correlation KL - acc
Reuters:				
money-fx vs. acq	700	700	1..20	-0.93
earn vs. trade	500	500	1..10	-0.92
Newsgroups:				
rec.autos vs. rec.motorcycles	700	700	1..20	-0.97
talk.politics.guns vs. talk.politics.mideast	700	700	1..20	-0.98

Figure 4.2: Correlation between $diff(T_i, T_0)$ and $accuracy(T_i)$

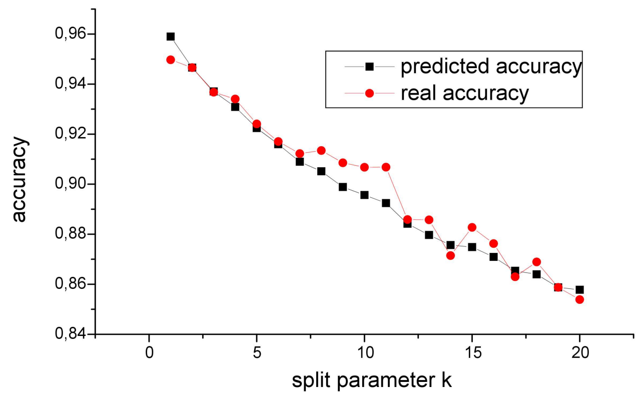

Figure 4.3: Accuracy Prediction by KL for Newsgroups Topics "rec.cars" and "rec.motorcycles"

The run-time cost of constructing the accuracy estimator for a basic SVM classifier, using leave-one-out estimation, is between $O(n^3)$ and $O(n^4)$ (n times retraining each with complexity of typically between $O(n^2)$ and $O(n^3)$) with n denoting the cardinality of the complete training set T_0. With $m - fold$ cross-validation instead of leave-one-out the cost is between $O(mn^2)$ and $O(mn^3)$. The $\xi\alpha$ estimator of [65] would require only a single training procedure (thus typically running in time $O(n^2)$ to $O(n^3)$), but is way too crude to be useful in our framework. For the Centroid method, training and leave-one-out estimation are combined and have only $O(n)$ run-time cost. Finally, for estimating the accuracy of an SVM classifier trained on some subset T_i we need time $O(n)$ for the KL computation (needed only in the KL estimator) plus the a priori probing cost, with training, cross-validation, and regression (needed in both the CV and the KL estimators), which in total is between $O((n/k)^2)$ and $O((n/k)^3)$ where k is the smaller one of our two probing points. With k being 10 or larger, this is a substantial saving compared to the basic SVM estimator.

4.4.2 Estimators for Restrictive Classifiers

For a basic classifier like SVM or Centroid, the leave-one-out decision step gives us a set of tuples $(d, confidence, isCorrect)$ where d is the left-out-document, $confidence$ is the classification confidence (i.e., hyperplane distance or probability), and $isCorrect$ is a Boolean value that tells us if the classifier trained with the $n-1$ remaining documents correctly classifies d (value 1) or not (value 0). Note that the confidence value may not be exported by the built-in estimators of existing software packages such as SVM light; in this case we either need to re-implement the leave-one-out estimator on top or modify the software or switch to a package that makes these values easily accessible.

Given a threshold for the confidence, it is now easy to compute the adjusted accuracy. We only need to restrict the summation over the $isCorrect$ values that form the basis of the average accuracy estimation to those $isCorrect$ values for which confidence exceeds the threshold, and the denominator for accuracy then is the count of all tuples with confidence higher than the threshold. Likewise, loss simply is the count of the tuples with confidence below the threshold divided by n.

4.4.3 Estimators for k-split Meta Classifiers

Now we explain how to construct an estimator for loss and accuracy (or equivalently error) of a k-split meta classifier with unanimous decision, given the estimators for the underlying classifiers $\{v_1, \ldots, v_k\}$.

Let $T_0 = T_1 \cup \ldots \cup T_k$ be our partitioning of the overall training data and let v_i be the classifier trained on T_i. We associate a Bernoulli random variable X_i with each v_i, where $X_i = 1$ if v_i classifies a document correctly, 0 otherwise.

A simple, but unfortunately oversimplified, approach would be to estimate the accuracy, $P[X_i = 1]$, or equivalently error, $P[X_i = 0]$, for every v_i by 1) assuming that X_i and X_j are pairwise independent for different v_i and v_j, and 2) using the KL-based prediction model of Section 4.4.1. Unfortunately, the independence assumption leads to fairly inaccurate estimations for both accuracy and document loss. Instead we pursue a more sophisticated approach that takes the correlations between classifiers into account.

To this end we run an additional cross-validation upfront. We consider three mutually disjoint subsets T_1, T_2, T_3 of T_0 where T_1 and T_2 serve to train classifiers v_1 and v_2 and T_3 is held-back test data to assess v_1 and v_2 such that neither v_1 nor v_2 has seen this test data before. Note that the three subsets may be easily derived by a random partitioning of a medium-sized random subset of T_0; we do not need to take all of T_0 into consideration thus reducing the computational cost. The training procedure gives us data points (x_1, x_2) for the joint distribution of (X_1, X_2). Thus we obtain an estimator for the covariance

$$cov(X_1, X_2) = \frac{1}{n-1} \cdot \sum (x_1 - \overline{x}_1)(x_2 - \overline{x}_2) \tag{4.6}$$

where n is the number of data points in T_3 and $\overline{x}_1$, $\overline{x}_2$ are the means of the marginal distributions of X_1 and X_2. From basic probability theory it follows that

$$P(X_1 = 1 \wedge X_2 = 1) = cov(X_1, X_2) + P(X_1 = 1) * P(X_2 = 1) \tag{4.7}$$

This procedure requires training the classifiers v_1 and v_2, but we do not need any further expensive steps for $k > 3$ by making two assumptions:

1. For any two subsets T_i, T_j in any possible k-split partitioning, the covariance is the same as $cov(X_1, X_2)$ computed above. So the covariance estimator for $k = 3$ can be reused without additional computations. Below we therefore refer to the covariance estimator simply as *cov* without any subscripts or arguments.

2. In a k-split partitioning we consider only the dependencies between v_i and v_{i+1} and postulate that all other pairs v_i and v_j can be considered as independent.

Assumption 1 is justified as long as all subsets T_i in a k-split partitioning are reasonably representative samples for the original data T_0.

We can justify Assumption 2 by using a tree dependence model, which is a well known approximation method in probabilistic IR ([100]): We define a *Dependence Graph*

$G = (V, E)$ where V consists of the Bernoulli Variables X_i, and which contains for all X_i, X_j $(i \neq j)$ an undirected edge $e(X_i, X_j)$ with weight $w(e(X_i, X_j)) = cov(X_i, X_j)$. We approximate the Dependence Graph by a Maximum Spanning Tree $G' = (V, E')$ which maximizes the sum of the edge weights. The nodes in G' with no edges in between are considered as independent. So we obtain:

$$P(X_1 = x_1, \ldots, X_k = x_k) = P(X_{root} = 1) \prod_{(i,j) \in E'} \frac{P(X_i = x_i, X_j = x_j)}{P(X_i = x_j)} \tag{4.8}$$

where X_{root} is the root node of the tree G' and $x_i \in \{0, 1\}$. Because $w(e(X_i, X_j)) = cov$ (where cov is a constant according to Assumption 1) we can w.l.o.g. choose X_1 as the root node and the edges (X_i, X_{i+1}) as tree edges. This corresponds to Assumption 2.

Now we have:

$$\begin{aligned} P(X_1 = 1, \ldots, X_k = 1) &= P(X_1 = 1) \prod_{i=1}^{k-1} P(X_{i+1} | X_i) \\ &= P(X_1 = 1) \prod_{i=1}^{k-1} \frac{P(X_i = 1, X_{i+1} = 1)}{P(X_i = 1)} \end{aligned} \tag{4.9}$$

By considering equation 4.7 and Assumption 1 we obtain:

$$P(X_1 = 1, \ldots, X_k = 1) = P(X_1 = 1) \prod_{i=1}^{k-1} \frac{P(X_i = 1) P(X_{i+1} = 1) + cov}{P(X_i = 1)} \tag{4.10}$$

Analogously we obtain $P(X_1 = 0, \ldots, X_k = 0)$.

Estimators for $P(X_i = 1)$ and $P(X_i = 0)$ (i.e., for accuracy and error of the single classifiers v_i) can be determined by either the CV or the KL estimator explained in Section 4.4.1.

Finally we can substitute these results in the following formulas for the loss estimator

$$\begin{aligned} & loss(Meta(v_1, \ldots, v_k)) = \\ & \quad 1 - P(X_1 = \ldots = X_k) = \\ & \quad 1 - (P(X_1 = 1, \ldots, X_k = 1) + P(X_1 = 0, \ldots, X_k = 0)) \end{aligned} \tag{4.11}$$

and the error and accuracy estimator

$$\begin{aligned} error(Meta(v_1, \ldots, v_k)) &= P(X_1 = 0, \ldots, X_k = 0 | X_1 = \ldots = X_k) \\ &= \frac{P(X_1 = 0, \ldots, X_k = 0)}{P(X_1 = 1, \ldots, X_k = 1) + P(X_1 = 0, \ldots, X_k = 0)} \end{aligned} \tag{4.12}$$

$$accuracy(Meta(v_1, \dots, v_k)) = 1 - error(Meta(v_1, \dots, v_k)) \tag{4.13}$$

The point of these analytic derivations is that we can start with a limited set of empirically determined quality measures (based on cross-validation techniques) and can assess candidates for a k-split partitioning meta classifier in an efficiently computable, solely analytic manner without further retraining. This capability is crucial for an efficient traversal of the space of possible k-split partitionings, as discussed in the next section.

In principle, the same estimation procedure can be carried over to a meta classifier based on k-fold random resampling. The only potential caveat is that the independence assumption for more than two partitions could be more critical in the case of overlapping partitions. We will discuss the estimators' accuracy in our experiments in Section 4.5.

4.4.4 Parameter Search Heuristics

Our goal is to minimize the error subject to the constraint that the loss is bounded by some application specific threshold.

For the basic classifiers, SVM, Centroid, and Naive Bayes, we simply perform a binary search over their control parameters, which are either hyperplane distance or probability threshold. This way we can easily find their best parameter settings. Varying these parameters does not require retraining the classifier with the above range of methods. Our estimation techniques presented in Section 4.4 allow us to predict loss and accuracy from the leave-one-out predictions for the non-restrictive baseline cases (with all thresholds set to 0).

For the k-split meta classifier, the parameter search space is much larger. We need to decide

- into how many partitions we split T_0,
- how we divide the positive training samples among the resulting partitions $T_1, \dots, T_k$, and
- how we divide the negative training samples (note that we may consider treating positive and negative samples differently for they may vary significantly in cardinality).

Obviously, there is some danger of combinatorial explosion here (e.g., exponentially many options in the cardinality of T_0); so we restrict ourselves to a fairly simple greedy heuristics. We only consider splitting T_0 into k equally sized subsets (plus/minus one if exactly equal sizes were impossible) and we always preserve the original ratio of positive to negative samples. For dividing the training samples themselves, we only consider random partitioning, which is both efficient and guarantees that the resulting training subsets reasonably preserve the statistical properties (e.g., feature distributions) of the original data. More sophisticated alternatives (e.g., dividing based on information-theoretic models) are subject to future research.

The meta classifiers that we tentatively construct consider a 1-split, a 2-split, a 3-split, and so on, and this leads to the fairly simple search procedure described in Figure 4.4.

```
Input: lower bound threshold for loss
Output: best choice of k,
         training subsets T1, ..., Tk
Algorithm:
k=1; T1 = T0; best := 1;
estimate loss(1) for
  the meta classifier with k=1; // in our case: 0
while (loss(k) < threshold) {
  randomly partition T0 into k subsets
  estimate loss(k+1);
  estimate error(k+1);
  if (loss(k+1) < threshold)
  {
    if (error(k+1) < best) best = k+1;
  };
  k = k+1;
};
```

Figure 4.4: Pseudocode: Search Procedure for the best Split Parameter

This procedure exploits that loss is almost certainly monotonically increasing with increasing k, not only with the unanimous-decision variant that we are using but also with most other variants of our meta classifier framework. For unanimous decision our experiments even showed that accuracy is monotonically increasing with increasing k; so in this particular case our procedure returns the maximum k such that the estimated loss is still acceptable.

4.5 Experiments

4.5.1 Setup

We performed a series of experiments with real-life data from:

1. The Newsgroups collection at [1]. This collection contains 17,847 postings collected from 20 Usenet newsgroups from topics like 'rec.autos', 'sci.space', etc.
2. The Internet Movie Database (IMDB) at [4]. Documents of this collection are short movie descriptions that include the storyboard, cast overview, and user comments. This collection contains 20 topics according to particular movie genres ('drama', 'horror' etc.). Only movies were considered that have a unique genre (avoiding movies that belong to, say, both 'drama' and 'science fiction'). Thereof it holds a total of 34,681 documents.

For our experiments we selected subsets of 250 and 500 documents for training (i.e., a small and a medium-sized training set), and tested our various methods with the remaining held-out data. We performed two kinds of experiments:

- *baseline experiments* investigated the classification error as a function of the document loss, and compared the measured results to the predictions by our various estimators, and
- *use-case experiments* investigated the error and loss as functions of the application goal for the maximum tolerable loss, and studied to what extent we could indeed automatically tune the methods to a given loss threshold.

All experiments were based on software written in Java, except for the base classifiers SVM, where we used SVM light, and Centroid, which we implemented in C++. We compared the following methods and meta methods:

- Restrictive variants of the base classifiers SVM and Centroid. (We also studied restrictive variants of a Bayesian classifier, but it was consistently outperformed by the other base methods, both as a classifier and as a base method in meta methods. Therefore we do not include these results here.)

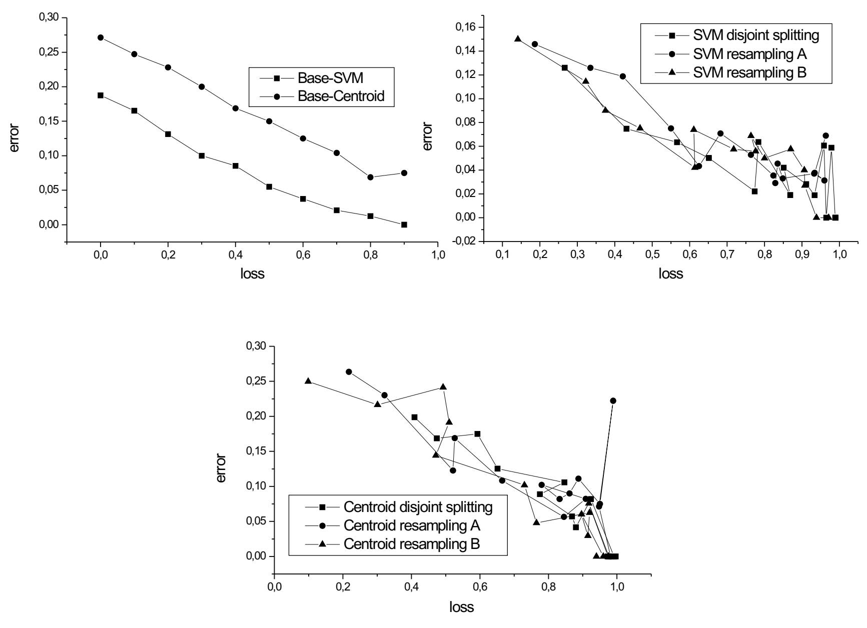

Figure 4.5: Loss-Error Tradeoff for "Drama vs. Horror"

- The k-split meta method, based on (non-restrictive variants of) either SVM or Centroid, where k was varied from 1 to 16.
- Two variants of the k-fold random resampling meta method, one with replacement of drawn samples (variant A) and one without replacement for the same training partition (variant B). Each one of these was based on either SVM or Centroid. k was varied from 1 to 16, and the number m of samples per training partition was set to $1.5 * (\#training\ docs)/k$ (e.g., 375 for a total training set of 500 documents and $k = 2$ partitions).

Our systematic experiments capture the behavior of classifiers and meta classifiers for pairs of topics such as "Drama vs. Horror" for IMDB data or "rec.autos vs. rec.motorcycles" for the Newsgroups data. For each data set we identified all topics with sufficiently many documents (> 900 for Newsgroups, > 550 for IMDB) for training and testing. These were 17 topics for Newsgroups and 5 for the genres of IMDB documents. We randomly choose 100 topic pairs from Newsgroups and 10 from IMDB. We computed micro-averaged results for these topic pairs.

4.5.2 Results

Baseline Experiments

Figure 4.5 illustrates the loss-error tradeoff for the two base methods SVM and Centroid, the k-split meta method and the two variants of random resampling. A typical observation is that being willing to lose up to 40 percent of the documents by abstaining on low-confidence decisions could reduce the classification error from about 20 percent down to less than 10 percent. This behavior was consistent across all methods, with variation of the quantitative results.

When comparing k-split vs. random resampling (Figure 4.6), we see that resampling (both variants) usually led to lower loss but often to significantly higher error for the same number of partitions. But the affordable loss can be effectively controlled by our tuning procedure; so for the same tolerable loss, k-split may simply use a slightly smaller number of partitions and would still usually be at least as good as resampling in terms of error. The analytical estimators for document loss turned out to be fairly accurate and slightly conservative. The estimator for error, on the other hand, turned out to be optimistically biased and moderately accurate at best. Figure 4.9 illustrates this on the example of two IMDB topics.

It is much easier to construct accurate and computationally inexpensive estimators for the k-split meta method than for the resampling approach. Therefore, we will disregard resampling in our discussion of use-case results in the next subsection.

The simple Centroid method as a basis for the k-split and resampling meta methods performed amazingly well relative to the theoretically much superior SVM classifiers. However, the Centroid-based meta methods did exhibit a non-negligible penalty in terms of classification error.

Use-Case Experiments

In a second line of experiments we studied how well our procedure for goal-oriented tuning was indeed able to determine practically viable parameter settings (Figure 4.7). Here we gave ourselves a goal for the acceptable loss threshold, ran our automatic tuning procedure for the various methods, and finally evaluated the tuned methods on the held-out validation data. The base methods can exactly control the loss, and thus inherently performed much better in terms of the actual loss. In terms of error, the base Centroid method was also superior to the Centroid-based k-split meta method, but lost against the SVM-based k-split meta method (for the same loss goal). In situations where the low error rate is critical (e.g., when automatically selecting new, initially unlabeled, training documents for a focused crawler), this gain may be important. The SVM-based k-split meta method was still outperformed by the base SVM method (albeit sometimes only by a small margin), but the key point here is that base SVM requires expensive cross-validation or even leave-one-out validation for building estimators.

Figure 4.8 compares the training times for the various meta methods and base methods that are necessary to construct the estimators. Base SVM without any partitioning would often be considered prohibitively expensive for interactive applications (i.e., when the training and estimation procedures themselves are part of user-perceived response times).

4.5.3 Summary and Lessons Learned

Summary

We have gained new insights into the fundamental tradeoffs between *classification accuracy*, *loss in ternary classification*, *amount of necessary training data*, and *training efficiency*.

The experiments clearly show that our analytical estimators are useful for driving the tuning procedure. In terms of absolute quality and meeting the error and loss goals of the application, the k-split meta methods are very competitive. The restrictive variant of the basic SVM classifier still is usually the best classifier, but its training and tuning time is an order of magnitude higher than the corresponding cost of the k-split meta method.

Lessons Learned

Our findings suggest the following guidelines for selecting the most appropriate method and tuning it towards a given application setting:

- When only few training documents (say < 100) are available and the time needed for training (incl. error estimation) is uncritical, then the *restrictive variant of SVM* is the method of choice. Its accuracy is usually the best among all competitors, and it can be easily tuned, as shown in this chapter, for a specified loss tolerance. A typical situation where these properties are important is for classifying query results in a personalized search engine. Such a system would be trained with few documents that reflect an individual user's interest profile, but this is performed offline, i.e., not within the response time of a query, so that training efficiency is not critical. At query time, it may be desirable for an advanced user to tolerate a certain loss and see only query results that clearly fall into the given scope of interest.

- When only few training documents (say < 100) are available and the efficiency of training (incl. error estimation) is critical, then the *restrictive Centroid variant* is the method of choice. Its accuracy is worse than for the other methods, but may still be acceptable to the application. Moreover, it can be easily tuned with regard to the accuracy-loss tradeoff. Last but not least, its training time is drastically shorter than that of an SVM classifier on the same training data. A situation where these arguments in favor of restrictive Centroid apply is within an expert user's focused crawler with online re-training [116]. Here a classifier may start with very few, manually selected, training documents that capture a user's interest profile. To incrementally improve the focused crawler, semisupervised learning techniques can be used to automatically select additional training data among the automatically classified documents and to dynamically re-train the classifier. As this additional training data comes with a non-negligible error probability, it may be important to accept only the highest-confidence documents and tolerate a certain loss. As the re-training is performed within an interactive session with an expert user, its efficiency is highly critical.
- When a medium or large number (say ≥ 100) of training documents are available and training efficiency (incl. error estimation) is a critical issue, then the *k-split meta method with SVM as base method* is most appropriate (with a reasonably chosen value of k, e.g. $k = 4$, so that the training partitions are sufficiently large). Its training time is substantially shorter than for restrictive SVM, and its accuracy is competitive to SVM, albeit not quite as good, and significantly better than that of restrictive Centroid. With regard to the accuracy-loss tradeoff, the k-split meta method can be automatically tuned using the estimators developed in this paper. A situation where this case arises is in generating, maintaining, and organizing Web information portals or digital libraries on special topics (e.g., for hiking and climbing, for music styles, for cultural heritage projects, etc.). Here a classifier can be used to aid a human portal administrator or digital-library curator to organize information more easily, quickly, and accurately. In such an environment, there should be a sufficient number of initial training documents, but the administrator may occasionally select additional high-quality documents for re-training, e.g., when she decides that a topic with (too) many documents should be subdivided into more specialized topics. If such reorganizations are to

be carried out interactively, allowing the administrator to intellectually validate the resulting classification quality, then efficient re-training is a must and flexible control over accuracy versus loss is important.

Newsgroups							
	Disjoint k-Split		**Resampling A**		**Resampling B**		**Method**
Partitions	avg(error)	avg(loss)	avg(error)	avg(loss)	avg(error)	avg(loss)	**#TrainDocs**
2	0.03	0.052	0.036	0.038	0.037	0.024	
4	0.018	0.131	0.023	0.101	0.024	0.09	SVM
8	0.008	0.26	0.013	0.2	0.013	0.188	250
16	0.003	0.456	0.005	0.368	0.005	0.351	
2	0.024	0.04	0.027	0.03	0.029	0.02	
4	0.016	0.097	0.019	0.076	0.02	0.066	SVM
8	0.008	0.187	0.011	0.15	0.013	0.141	500
16	0.004	0.331	0.006	0.262	0.006	0.256	
2	0.057	0.05	0.062	0.04	0.067	0.02	
4	0.038	0.135	0.045	0.111	0.049	0.084	Centroid
8	0.023	0.314	0.029	0.248	0.03	0.206	250
16	0.014	0.59	0.017	0.49	0.017	0.445	
2	0.057	0.032	0.059	0.03	0.062	0.016	
4	0.041	0.09	0.048	0.073	0.048	0.058	Centroid
8	0.025	0.194	0.031	0.153	0.033	0.132	500
16	0.014	0.397	0.019	0.296	0.019	0.276	

IMDB							
	Disjoint k-Split		**Resampling A**		**Resampling B**		**Method**
#Partitions	avg(error)	avg(loss)	avg(error)	avg(loss)	avg(error)	avg(loss)	**#TrainDocs**
2	0.113	0.193	0.139	0.133	0.155	0.078	
4	0.068	0.399	0.103	0.315	0.092	0.288	SVM
8	0.026	0.629	0.046	0.533	0.048	0.51	250
16	0.005	0.791	0.017	0.739	0.016	0.729	
2	0.176	0.171	0.142	0.105	0.166	0.067	
4	0.14	0.359	0.119	0.246	0.117	0.228	SVM
8	0.066	0.6	0.068	0.409	0.072	0.413	500
16	0.018	0.789	0.016	0.629	0.017	0.612	
2	0.105	0.149	0.119	0.141	0.13	0.063	
4	0.067	0.376	0.088	0.291	0.085	0.25	Centroid
8	0.027	0.623	0.045	0.545	0.051	0.47	250
16	0.008	0.895	0.026	0.816	0.024	0.771	
2	0.125	0.098	0.126	0.071	0.151	0.032	
4	0.085	0.264	0.091	0.215	0.095	0.176	Centroid
8	0.027	0.467	0.05	0.395	0.067	0.345	500
16	0.012	0.66	0.018	0.631	0.016	0.583	

Figure 4.6: Micro-Averaged Results for Different Restrictive Splitting and Resampling Methods for the Newsgroups and the IMDB Data Set

Newsgroups			
Loss Threshold	**Disjoint k-Split** Avg(error)	**BaseMethod** Avg(error)	**Method #TrainDocs**
0.1	0.033	0.021	
0.3	0.017	0.009	SVM
0.5	0.011	0.005	250
0.7	0.007	0.003	
0.9	0.004	0.002	
0.1	0.024	0.014	
0.3	0.013	0.005	SVM
0.5	0.009	0.003	500
0.7	0.006	0.002	
0.9	0.005	0.001	
0.1	0.059	0.074	
0.3	0.0403	0.045	Centroid
0.5	0.027	0.018	250
0.7	0.017	0.01	
0.9	0.014	0.006	
0.1	0.052	0.068	
0.3	0.032	0.039	Centroid
0.5	0.021	0.015	500
0.7	0.016	0.008	
0.9	0.015	0.005	

IMDB			
Loss Threshold	**Disjoint k-Split** Avg(error)	**BaseMethod** Avg(error)	**Method #TrainDocs**
0.1	0.176	0.15	
0.3	0.137	0.114	SVM
0.5	0.095	0.076	250
0.7	0.069	0.046	
0.9	0.028	0.042	
0.1	0.176	0.154	
0.3	0.149	0.108	SVM
0.5	0.109	0.075	500
0.7	0.086	0.042	
0.9	0.047	0.025	
0.1	0.136	0.121	
0.3	0.122	0.088	Centroid
0.5	0.08	0.073	250
0.7	0.074	0.053	
0.9	0.038	0.042	
0.1	0.153	0.128	
0.3	0.128	0.092	Centroid
0.5	0.097	0.065	500
0.7	0.07	0.039	
0.9	0.045	0.033	

Figure 4.7: Micro-Averaged Tuning Results for the Newsgroups and the IMDB Data Set

partitions	k-Split SVM	k-Split Centroid
1	19.64	0.67
2	8.44	0.44
4	4.12	0.32
8	2.0	0.2
16	1.12	0.19

L-fold	SVM	Centroid
2	8.22	0.45
3	25.65	1.05
4	52.88	1.64
5	69.55	2.3

L-1-O SVM	L-1-0 Centroid
670064.64	2.64

Figure 4.8: Training and Estimation Times for k-Split, L-fold Cross Validation, and Leave-one-out (in Seconds), on 3 GHz Intel Pentium 4 PC

#par-titions	**Disjoint k-Split**						**Resampling A**		**Resampling B**		Method **#Train-Docs**
	Loss	estLossCV	estLossKL	Error	estErrorCV	estErrorKL	Loss	Error	Loss	Error	
2	0.266	0.247	0.462	0.126	0.11	0.043	0.205	0.157	0.113	0.156	
4	0.566	0.581	0.688	0.063	0.046	0.009	0.473	0.088	0.399	0.102	SVM
8	0.869	0.91	0.922	0.019	0.008	0.001	0.809	0.046	0.726	0.05	250
16	0.99	0.999	0.999	0	0	0.002	0.989	0	0.959	0.06	
2	0.179	0.215	0.311	0.082	0.099	0.041	0.141	0.096	0.115	0.096	
4	0.418	0.54	0.514	0.061	0.028	0.011	0.291	0.075	0.312	0.056	SVM
8	0.744	0.871	0.813	0.023	0.002	0.001	0.609	0.045	0.541	0.026	500
16	0.947	0.995	0.996	0	0	0	0.85	0.039	0.862	0.043	
2	0.409	0.29	0.635	0.199	0.311	0.133	0.141	0.255	0.09	0.234	
4	0.593	0.778	0.849	0.175	0.1	0.05	0.686	0.092	0.411	0.168	Centroid
8	0.869	0.948	0.976	0.057	0.03	0.01	0.84	0.063	0.79	0.077	250
16	0.996	0.999	0.1	0	0.003	0.004	0.959	0	0.983	0.071	
2	0.494	0.192	0.655	0.18	0.368	0.18	0.1	0.225	0.056	0.24	
4	0.647	0.867	0.865	0.075	0.059	0.08	0.309	0.243	0.321	0.182	Centroid
8	0.809	0.737	0.976	0.0462	0.012	0.015	0.676	0.1	0.65	0.118	500
16	0.95	0.998	0.999	0	0.004	0.001	0.898	0	0.856	0.041	

Figure 4.9: Different Restrictive Splitting and Resampling Methods for ”Drama vs. Horror”

5 Using Restrictive Meta Classification for Junk Elimination

Up to now it was assumed that all underlying classifiers had sufficient training data: both positive and negative samples of every thematic that might occur among the test documents. In this chapter we drop this assumption and make a major step forward to cope with corpora that are not necessarily "in tune" with the thematic classes that were defined apriori. This is a very significant case with "open" corpora like the Web with a huge amount of topics and documents for which comprehensive training is absolutely impossible. It is not clear how a classifier trained to discriminate topics based on training data about "computer sciene", "mathematics", and "physics" will behave on documents about, say, "esoterism"; there is a significant difference between negative examples and "junk" documents.

5.1 Tradeoffs for Restrictive Classification in the Junk Elimination Scenario

In this section we describe the tradeoffs that occur in restrictive classification if the test set contains junk documents (beside documents of interest). Consider a training set T consisting of documents from two classes *pos* and *neg*, and a set of unlabeled documents U containing documents from *pos* and *neg*, and *junk* documents that are not in these classes. (The scenario can be easily generalized to a set of l classes $C = \{c_1, \dots, c_l\}$ instead of two classes.) Given a document $d \in U$, a restrictive classifier gives us the result $+1$ if it classifies the document into *pos*, -1 if it classifies the document into *neg*, 0 if the classifier abstains. The possible combinations between the real classes and the possible results of a classifier are shown in the contingency table in Figure 5.1. In this notation $N+$ is the set of documents in *neg* which are assigned to class *pos* by the classifier, $J0$ is the set of junk documents from U where the classifier abstains, etc.

		classification		
		+	-	0
real class	pos	P+	P-	P0
	neg	N+	N-	N0
	junk	J+	J-	J0

Figure 5.1: Contingency Table for Restrictive Classification with Junk Reduction

An appropriate restrictive classifier should optimize the following quality measures:

1. Maximize *junk reduction* (fraction of junk documents dismissed by the classifier):
$$junkRed := \frac{|J0|}{|J+| + |J-| + |J0|} \tag{5.1}$$
2. Minimize *loss* (fraction of dismissed documents from the classes of interest *pos* and *neg*):
$$loss := \frac{|P0| + |N0|}{|P+| + |P-| + |N+| + |N-| + |P0| + |N0|} \tag{5.2}$$
3. Minimize *error* (fraction of non-dismissed documents classified into the wrong class):
$$error := \frac{|P-| + |N+| + |J+| + |J-|}{|P+| + |P-| + |N+| + |N-| + |J+| + |J-|} \tag{5.3}$$

As *document reduction* (not to confuse with the loss), we define the fraction of documents in U, where the classifier abstains:
$$docRed := \frac{|P0| + |N0| + |J0|}{|U|} \tag{5.4}$$
The document reduction can be observed directly from the classifier output without knowing the real class labels of the documents in U. The document reduction has an implicit influence on *junkRed*, *loss* and *error*.

In practice we observe a tradeoff between the loss on one hand and junk-reduction and error on the other hand.

5.2 Making Simple Methods Restrictive

We can use confidence measures to make simple methods restrictive, as described in Section 4.1.

5.3 Restrictive Meta Methods

For meta classification with a given set $V = \{v_1, \dots, v_k\}$ of k binary classifiers we can perform meta classification using a linear classifier combination with thresholding, as described in Section 4.2:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

5.4 A Probabilistic Model for Restrictive Meta Methods in a Junk Reduction Scenario

In this section we extend the probabilistic model of Section 4.4.3 to the case that test documents may contain junk documents and we provide approximations for $loss$, $error$ and $junkRed$. This leads to a better understanding of why meta classification can be used for junk reduction.

Consider the unanimous-decision meta method. We associate a Bernoulli random variable X_i with each classification method v_i, where $X_i = 1$ if v_i classifies a document into class *pos* and $X_i = 0$ if v_i classifies a document into class *neg*. We want to compute the probability $P(X_1 = \ldots = X_k | Junk)$ that the classifiers v_i provide a unanimous decision if they are presented a junk document. From basic probability theory it follows that

$$\begin{aligned} &P(X_1 = 1 \wedge X_2 = 1 | Junk) = \\ &\quad cov(X_1, X_2 | Junk) + P(X_1 = 1 | Junk) \cdot P(X_2 = 1 | Junk) \end{aligned} \tag{5.6}$$

Where

$$cov(X_1, X_2 | Junk) = \frac{1}{n-1} \sum_j (x_1 - \overline{x_1})(x_2 - \overline{x_2}) \tag{5.7}$$

is the covariance for the data points (x_1, x_2) of the joint distribution of (X_1, X_2) on the set of junk documents.

To model the most important correlations among $l > 2$ classification methods we use a tree dependence model, which is a well known approximation method in probabilistic IR ([100]). We define a *Dependence Graph* $G = (V, E)$ where V consists of the Bernoulli variables X_i and which contains for all X_i,X_j $(i \neq j)$ an undirected edge $e(X_i, X_j)$ with weight $w(e(X_i, X_j))) = cov(X_i, X_j)$. We approximate the Dependence Graph by a maximum spanning tree $G' = (V, E')$ which maximizes the sum of the edge weights. The nodes in G' with no edges in between are considered as independent. So we obtain:

$$\begin{aligned}&P(X_1 = x_1, \ldots, X_k = x_k|Junk) = \\ &\quad P(X_{root} = 1|Junk) \prod_{(i,j)\in E'} \frac{P(X_i = x_i, X_j = x_j|Junk)}{P(X_i = x_j|Junk)}\end{aligned} \tag{5.8}$$

where X_{root} is the root node of the tree G' and $x_i \in \{0, 1\}$. Now we introduce the following special case: For any two classification methods v_i,v_j the covariance has approximately the same value cov. With $w(e(X_i, X_j)) = cov$ we can (without loss of generality) choose X_1 as the root node and the edges (X_i, X_{i+1}) as tree edges.

Now we have:

$$\begin{aligned}&P(X_1 = 1, \ldots, X_k = 1|Junk) = \\ &\quad P(X_1 = 1|Junk) \prod_{i=1}^{k-1} P(X_{i+1} = 1|X_i = 1|Junk) = \\ &\quad P(X_1 = 1|Junk) \prod_{i=1}^{k-1} \frac{P(X_i = 1, X_{i+1} = 1|Junk)}{P(X_i = 1|Junk)}\end{aligned} \tag{5.9}$$

By considering equation 5.6 and the above assumption about the covariance we obtain

$$\begin{aligned}&P(X_1 = 1, \ldots, X_k = 1|Junk) = \\ &\quad P(X_1 = 1|Junk) \prod_{i=1}^{k-1} \frac{P(X_i = 1|Junk)P(X_{i+1} = 1|Junk) + cov}{P(X_i = 1|Junk)}\end{aligned} \tag{5.10}$$

Analogously we obtain $P(X_1 = 0, \ldots, X_k = 0|Junk)$.

If we assume that for *junk* documents the classes *pos* and *neg* are equally likely, we can substitute in the above formulas:

$$P(X_i = 1|Junk) = P(X_i = 0|Junk) = \frac{1}{2} \tag{5.11}$$

For the junk reduction we substitute the above formulas into:

$$\begin{aligned}&junkRed = 1 - P(X_1 = \ldots = X_k|Junk) = \\ &\quad 1 - (P(X_1 = 0, \ldots, X_k = 0|Junk) + P(X_1 = 1, \ldots, X_k = 1|Junk))\end{aligned} \tag{5.12}$$

To compute the probabilities that all classifiers v_i classify a document into the same class, if the document belongs to one of the classes in $C = \{pos, neg\}$, we associate a Bernoulli variable X'_i with each classification method v_i, where $X'_i = 1$ if v_i classifies a document correctly, 0 otherwise. We want to compute the probabilities $P(X'_1 = 1, \ldots, X'_k = 1|C)$ and $P(X'_1 = 0, \ldots, X'_k = 0|C)$ that all classifiers classify a document correctly / incorrectly if the document belongs to one of the classes in C.

With analogous arguments as above we obtain the following approximation:

$$P(X'_1 = 1, \ldots, X'_k = 1|C) = \\ P(X'_1 = 1|C) \prod_{i=1}^{k-1} \frac{P(X'_i = 1|C)P(X'_{i+1} = 1|C) + cov'}{P(X'_i = 1|C)} \tag{5.13}$$

where cov' is the covariance on the documents in C. Analogously we obtain $P(X'_1 = 0, \ldots, X'_k = 0|C)$.

Let $P(C)$ be the probability that a document belongs to a class in C and $P(Junk)$ be the probability that a document is a junk document. Then we obtain approximations for *junkRed*, *loss*, *error*, and *docRed* by inserting the above expressions into:

$$junkRed = 1 - (P(X_1 = 1, \ldots, X_k = 1|Junk) + P(X_1 = 0, \ldots, X_k = 0|Junk)) \tag{5.14}$$

$$loss = 1 - (P(X'_1 = 1, \ldots, X'_k = 1|C) + P(X'_1 = 0, \ldots, X'_k = 0|C)) \tag{5.15}$$

$$error = \frac{P(C)P(X'_1 = 0, \ldots, X'_k = 0|C) + P(Junk)P(X_1 = \ldots = X_k|Junk)}{1 - junkRed \cdot P(Junk) - loss \cdot P(C)} \tag{5.16}$$

$$docRed = junkRed \cdot P(Junk) + loss \cdot P(C) \tag{5.17}$$

As an illustrative example we consider the case that the $k > 2$ classification methods have the same probability $p < 0.5$ to misassign a document from C (i.e. the classification methods perform better than random), that in the case of a junk document the assignment of the classes *pos* or *neg* are equally likely, that we have in all cases a covariance $c < p(1-p)$ (i.e. the classification methods are not perfectly correlated.), and that our document corpus contains 50 percent junk documents. In this case we would obtain for *junkRed*, *loss* and *error*:

$$junkRed = 1 - \left(\frac{c + 1/4}{1/2}\right)^{k-1} \tag{5.18}$$

$$loss = 1 - \left((1-p) \left(\frac{c + (1-p)^2}{1-p} \right)^{k-1} + p \left(\frac{c+p^2}{p} \right)^{k-1} \right) \tag{5.19}$$

$$error = \frac{p \left(\frac{c+p^2}{p} \right)^{k-1} + \left(\frac{c+1/4}{1/2} \right)^{k-1}}{\left(\frac{c+1/4}{1/2} \right)^{k-1} + (1-p) \left(\frac{c+(1-p)^2}{1-p} \right)^{k-1} + p \left(\frac{c+p^2}{p} \right)^{k-1}} \tag{5.20}$$

It is easy to show that for $k \to \infty$ the loss converges monotonically to 1, and the error to 0 (i.e. with more classification methods we can obtain a lower error but pay the price of a higher loss). Furthermore also *junkRed* converges to 1 and the salient invariant $loss > junkRed$ holds. Even $\frac{1-loss}{1-junkRed}$ converges to ∞; this means, that with increasing k we dismiss much more junk documents than documents of interest. The covariance plays the role of a "smoothing constant": with higher correlated classification methods the convergence of both loss and error is slowed down.

5.5 Experiments

5.5.1 Setup

We performed a series of experiments with real-life data from

1. The Newsgroups collection as described in Section 4.5
2. The Internet Movie Database (IMDB) described in Section 4.5.
3. The Reuters articles [80]. This is the most widely used test collection for text categorization research. The collection contains 21,578 Reuters newswire articles; 12,904 of them are subdivided into categories ('earn', 'grain', 'trade', etc.).

For each data set we identified all topics with sufficiently many documents. These were 20 topics for Newsgroups, 7 for Reuters and 9 from IMDB. Among these topics we randomly chose 100 topic pairs from Newsgroups, 30 from IMDB and 20 from Reuters. For each topic pair we randomly chose 25, 50 or 100 training documents per class and kept 500 documents per class for Newsgroups, 200 documents per class for IMDB and 400 documents per class from Reuters (distinct from the training set and also randomly chosen) for the validation of the classifiers for each pair. Additionally we "spoiled" the validation set for each pair by increasing this set by 50 percent by adding randomly chosen "junk documents" from different topics. Finally, we computed macro-averaged results for these topic pairs.

5.5.2 Results

# TrainDocs	Meta		restrictive Base			Base			Dataset
			base1	base2	base3	base1	base2	base3	
	avg (docRed)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	
25	0.159	0.489	0.493	0.489	0.489	0.52	0.515	0.518	IMDB
50	0.208	0.457	0.463	0.457	0.457	0.506	0.499	0.499	
100	0.188	0.432	0.439	0.433	0.433	0.483	0.475	0.479	
25	0.165	0.344	0.358	0.358	0.358	0.419	0.416	0.417	Newsg.
50	0.166	0.316	0.327	0.328	0.329	0.398	0.396	0.397	
100	0.143	0.31	0.318	0.315	0.315	0.385	0.381	0.381	
25	0.099	0.326	0.335	0.334	0.331	0.378	0.374	0.375	Reuters
50	0.086	0.318	0.323	0.319	0.318	0.366	0.362	0.362	
100	0.078	0.314	0.321	0.316	0.315	0.360	0.357	0.356	
79	0.074	0.301	0.282	0.319	0.327	0.323	0.348	0.351	Web

Figure 5.2: Error of Meta Classification on Reuters, Newsgroups, and IMDB

# TrainDocs	Meta			restrictive Base						Dataset
				base1	base2	base3	base1	base2	base3	
	avg (docRed)	avg (loss)	avg (jRed)	avg (loss)	avg (loss)	avg (loss)	avg (jRed)	avg (jRed)	avg (jRed)	
25	0.159	0.147	0.183	0.149	0.146	0.148	0.181	0.186	0.182	IMDB
50	0.208	0.192	0.239	0.188	0.186	0.187	0.246	0.252	0.248	
100	0.188	0.167	0.231	0.165	0.162	0.163	0.234	0.24	0.238	
25	0.165	0.109	0.276	0.118	0.122	0.12	0.259	0.251	0.254	Newsg.
50	0.166	0.098	0.301	0.103	0.108	0.108	0.29	0.281	0.282	
100	0.143	0.077	0.275	0.078	0.079	0.078	0.272	0.271	0.273	
25	0.099	0.047	0.202	0.055	0.057	0.055	0.186	0.184	0.187	Reuters
50	0.086	0.034	0.188	0.038	0.037	0.037	0.181	0.182	0.184	
100	0.078	0.024	0.186	0.034	0.029	0.028	0.167	0.178	0.179	
79	0.074	0.044	0.144	0.032	0.055	0.06	0.173	0.118	0.143	Web

Figure 5.3: Loss and JunkReduction of Meta Classification on Reuters, Newsgroups and IMDB

In all discussed experiments, the standard bag-of-words model (using tf-values to build L1-normalized feature vectors) with different feature selection methods was used for document representation and we used SVM as learning algorithm

In our experiments we considered the following base methods:

- *base1*: Feature selection by Mutual Information (top 200 terms); learning by linear SVM
- *base2*: Feature selection by Information Gain(top 200 terms); learning by linear SVM
- *base3*: Feature selection by Chi Squared Statistics (top 200 terms); learning by linear SVM

There are many alternative ways to build the base classifiers, e.g. using Naive Bayes, Decision Trees, etc. Here we chose linear SVM because it has been shown to often outperform other methods in text classification tasks - see e.g. [44].

In the first experimental serial we compared the meta results with the results of the underlying base methods and the restrictive base methods (inducing the same document reduction as the meta method). (Figures 5.2 and 5.3)

In the second experimental serial we compared each base method for different degrees of restrictivity [1] (inducing different document reductions). (Figure 5.4).

The main observations are:

- For all experiments the average error of the meta method was lower than the error of the *best* underlying base method.
- The junk reduction is (for restrictive base methods as well as for meta methods) always significantly higher than the loss (i.e. we dismiss a higher percentage of junk than of documents of interest).
- For the IMDB data set the ratio $junkRed : loss$ is best for the best base method, for the Reuters and Newsgroups data sets this ratio is best for the meta method.
- We can clearly observe the tradeoffs between $loss$ on the one hand and $error$ and $junkRed$ on the other hand described in Section 5.1 and analyzed in Section 5.4.

[1]We randomly chose the training and test documents once more for these experiments, causing minimal differences in the results for *docRed* = 0 compared to the base methods of the first experimental serial.

As an application example we tested junk reduction for a Web crawl. We obtained our training set from a bookmark file containing 79 documents of the categories "Movies" and "Computer Science" and started the crawl on the portals shown in Figure 5.5. By this crawl we obtained an overall number of 1061 documents consisting of 400 documents about computer science, 348 about movies, and 313 junk documents. We evaluated the techniques described above on this data set. The results are shown in Figures 5.2 through 5.4 (data set "Web"). As in the previous experiments, the junk reduction was much higher than the loss for all restrictive methods. In terms of loss, error and junkReduction the meta method performed better than two of the 3 underlying base methods but the best restrictive base method outperformed the meta method in this experiment.

5.5.3 Discussion

The experiments show that all restrictive methods (i.e. meta methods as well as restrictive base methods) dismiss a significantly higher percentage of junk than of documents of interest, and additionally decrease the classification error on all data sets.

Comparing meta classifiers and restrictive base classifiers there is no clear winner: For the IMDB and Web data, the best base classifier outperformed the meta classifier; for Newsgroups and Reuters, the meta classifier outperformed the base classifiers.

	base1	base2	base3	base1	base2	base3	base1	base2	base3	
docRed	avg (error)	avg (error)	avg (error)	avg (loss)	avg (loss)	avg (loss)	avg (jRed)	avg (jRed)	avg (jRed)	**Dataset**
0	0.517	0.509	0.509	0	0	0	0	0	0	
0.1	0.498	0.492	0.489	0.094	0.091	0.092	0.111	0.117	0.116	
0.2	0.48	0.472	0.47	0.183	0.182	0.183	0.234	0.237	0.233	
0.3	0.461	0.449	0.451	0.278	0.273	0.277	0.345	0.354	0.346	
0.4	0.441	0.431	0.433	0.372	0.369	0.374	0.456	0.462	0.452	IMDB
0.5	0.416	0.407	0.412	0.466	0.464	0.47	0.568	0.572	0.561	
0.6	0.397	0.389	0.391	0.567	0.565	0.567	0.666	0.669	0.666	
0.7	0.375	0.369	0.372	0.67	0.667	0.67	0.76	0.765	0.759	
0.8	0.351	0.345	0.348	0.777	0.776	0.777	0.847	0.849	0.847	
0.9	0.309	0.313	0.307	0.884	0.885	0.885	0.932	0.93	0.93	
0	0.42	0.417	0.417	0	0	0	0	0	0	
0.1	0.386	0.384	0.383	0.073	0.075	0.074	0.154	0.15	0.153	
0.2	0.348	0.346	0.345	0.145	0.147	0.146	0.31	0.307	0.309	
0.3	0.307	0.305	0.304	0.218	0.22	0.219	0.463	0.461	0.462	
0.4	0.261	0.259	0.26	0.297	0.298	0.298	0.605	0.605	0.604	Newsg.
0.5	0.216	0.215	0.216	0.385	0.387	0.387	0.729	0.727	0.726	
0.6	0.176	0.172	0.173	0.488	0.487	0.487	0.825	0.827	0.825	
0.7	0.139	0.135	0.137	0.602	0.6	0.601	0.897	0.899	0.897	
0.8	0.108	0.102	0.105	0.727	0.725	0.726	0.947	0.95	0.948	
0.9	0.081	0.076	0.078	0.86	0.859	0.859	0.98	0.982	0.981	
0	0.38	0.377	0.377	0	0	0	0	0	0	
0.1	0.336	0.336	0.334	0.057	0.06	0.058	0.185	0.181	0.183	
0.2	0.291	0.292	0.29	0.119	0.121	0.119	0.362	0.359	0.361	
0.3	0.247	0.247	0.245	0.188	0.19	0.189	0.523	0.52	0.522	
0.4	0.209	0.209	0.208	0.272	0.274	0.275	0.656	0.652	0.65	Reuters
0.5	0.174	0.172	0.172	0.369	0.369	0.37	0.763	0.762	0.761	
0.6	0.142	0.144	0.144	0.477	0.479	0.48	0.847	0.842	0.841	
0.7	0.113	0.111	0.109	0.596	0.595	0.595	0.908	0.909	0.91	
0.8	0.087	0.083	0.085	0.724	0.723	0.723	0.952	0.955	0.954	
0.9	0.068	0.064	0.068	0.86	0.859	0.859	0.981	0.983	0.981	
0	0.323	0.348	0.351	0	0	0	0	0	0	
0.1	0.266	0.311	0.316	0.041	0.074	0.079	0.24	0.163	0.15	
0.2	0.214	0.265	0.284	0.095	0.143	0.164	0.45	0.335	0.284	
0.3	0.168	0.215	0.229	0.166	0.214	0.225	0.62	0.505	0.479	
0.4	0.152	0.198	0.206	0.27	0.31	0.317	0.709	0.613	0.597	Web
0.5	0.134	0.162	0.177	0.377	0.4	0.409	0.792	0.738	0.716	
0.6	0.127	0.146	0.167	0.497	0.509	0.521	0.843	0.815	0.786	
0.7	0.116	0.119	0.15	0.62	0.62	0.634	0.888	0.888	0.856	
0.8	0.113	0.117	0.122	0.745	0.746	0.747	0.93	0.927	0.923	
0.9	0.131	0.056	0.093	0.873	0.862	0.868	0.962	0.987	0.974	

Figure 5.4: Error, Loss and Junk Reduction for Restrictive Base Methods on Reuters, Newsgroups, and IMDB and T = 25 TrainDocs per Class and for Web Documents with T = 79 TrainDocs per Class

Computer Science:
`http://dir.yahoo.com/Science/Computer_Science/`
`http://www.developer.com/`
`http://www.techweb.com/`
`http://directory.google.com/Top/Computers/Computer_Science/`
`http://library.albany.edu/subject/csci.htm`

Movies:
`http://www.allmovieportal.com/`
`http://www.galatta.com/`
`http://adutopia.subportal.com/cgi-bin/apollo/apollo.cgi`
`http://dir.yahoo.com/Entertainment/Movies_and_Film/Genres/`
`http://www.badmovies.org/`

Figure 5.5: Starting Points for the Web Crawl

6 Restrictive Meta Clustering

This chapter addresses the problem of automatically structuring heterogenous document collections by using clustering methods. We adapt and extend the concept of restrictive (supervised) classification, described in the previous chapter to (unsupervised) clustering. So, in contrast to traditional clustering, we study restrictive methods and ensemble-based meta methods that may decide to leave out some documents rather than assigning them to inappropriate clusters with low confidence. These techniques result in higher cluster purity, and better overall accuracy, and make unsupervised self-organization more robust.

6.1 Making Simple Methods Restrictive

Analogously to the idea of restrictive classification, as described in the previous chapters, the idea of restrictive clustering is to avoid making a decision about a document at all if that decision can be made only with low confidence. So out of a given set of unlabeled data U, our method chooses a subset S of documents that are assigned to clusters, and abstains on the documents in $U - S$. We call the ratio $|U - S|/|U|$ of dismissed documents the document *loss*.

We can use confidence measures to make simple methods restrictive. For the different variants of the k-means method a natural confidence measure is the inverse distance of a document vector from the nearest centroid (or some other similarity measure). So we can tune these methods by requiring that the documents accepted for one of the clusters have a distance below some threshold, and abstain otherwise. The threshold is our tuning parameter.

Given an application-acceptable loss of L percent, we can make a clustering method restrictive by dismissing the L percent of the documents with the lowest confidence values. Although this is a fairly straightforward idea, we are not aware of prior literature that has explicitly considered such restrictive clustering methods.

6.2 Restrictive Meta Methods

For meta clustering we are given a set $C = \{c_1, \ldots, c_l\}$ of different clustering methods. A document d is assigned to one of k clusters with labels $\{1, \ldots, k\}$: $c_i(d) \in \{1, \ldots, k\}$. The idea of meta clustering is now to combine the different clustering results in an appropriate way.

6.2.1 Metamapping

To combine the $c_i(d)$ into a metaresult, the first problem is to determine which cluster labels of different methods c_i correspond to each other. Note that cluster label 2 of method c_i does not necessarily correspond to the same cluster label 2 of method c_j, but could correspond to, say, cluster label 5. With perfect clustering methods the solution would be trivial: the documents labeled by c_i as a would be exactly the documents labeled by c_j as b, and we could easily test this with one representative per cluster. This assumption is, of course, unrealistic; rather clustering results exhibit a certain fuzziness so that some documents end up in clusters other than their perfectly suitable cluster. Informally, for different clustering methods we would like to associate the clusters whith each other which are “most correlated”

Formally, for every method c_i we want to determine a bijective function $map_i : \{1, \ldots, k\} \to \{1, \ldots, k\}$ which assigns each label $a \in \{1, \ldots, k\}$ assigned by c_i to a meta label $map_i(a)$. By this mapping the clustering labels of the different methods are associated with each other and we can define the clustering result for document d using method c_i as:

$$res_i(d) := map_i(c_i(d)) \tag{6.1}$$

We now describe different ways to obtain the map_i functions.

Method A: Correlation-based Approach We want to maximize the correlation between the cluster labels. For sets $A_1, \ldots, A_x$, we can define their *overlap* as

$$overlap(A_1, \ldots, A_x) := \frac{|A_1 \cap \ldots \cap A_x|}{|A_1| + \ldots + |A_x| - |A_1 \cap \ldots \cap A_x|} \tag{6.2}$$

Now using

$$A_{ij} := \{d \in U | res_i(d) = j\} \tag{6.3}$$

we can define the *average overlap* for a document set U and the set of clustering methods C as

$$\frac{1}{k}\sum_{j=1}^{k}\frac{1}{\binom{l}{2}}\sum_{(i,m)\in\{1,\ldots,k\}^2,i<m} overlap(A_{ij},A_{mj}) \tag{6.4}$$

We are interested in the mappings map_i which maximize the average overlap. However there is a combinatory explosion: there are $k!^{l-1}$ possibilities to build mappings. This problem can be transformed into a multi-dimensional assignment problem (MAP) [96] which has been shown to be NP-complete; thus this approach is only viable for small values of k and l.

A greedy approach is to maximize the overlap between pairs of clustering methods, e.g. c_1 and c_2, c_2 and c_3, ..., c_{l-1} and c_l, and to use transitivity to compute an overall mapping. Each of these subproblems can be formulated as an assignment problem that can be solved by the Hungarian Algorithm [76] which has been shown to have a runtime complexity of $O(k^3)$. Since we have to solve $l-1$ of such subproblems, the complexity of the greedy approach is $O(k^3\cdot l)$.

An even greedier approach is to find for all c_{i-1} and c_i the highest, second highest, third highest, etc. $overlap(A_{i-1,j},A_{ij})$, to derive the mapping for c_{i-1} and c_i and to compute the overall mapping using again transitivity. This can be done by first sorting the k^2 overlap measures in time $O(k^2\log k^2)=O(k^2\log k)$, and at most one scan of the obtained list. Thus the overall complexity is $O(k^2\log k\cdot l)$.

Method B: Purity-based Approach Instead of maximizing the average overlap for all mappings, one may consider a ranked list of meta clusters, ordered by their *"purity"*. The underlying idea of this approach is to prioritize the clusters that produce a high overlap only in one particular (potentially proper) combination and low overlaps otherwise. A detailed description of this approach can be found in our paper [112].

Method C: Association-Rule-based Approach Another approach is to use association rules mining [10, 58], which was popularized for market basket analysis. Our (shopping) items here would be tuples $(c_i, c_i(d))$ of clustering methods and cluster labels. To every document $d\in U$ we can now assign an itemset:

$$\{(c_1,c_1(d)),\ldots,(c_k,c_k(d))\} \tag{6.5}$$

For these itemsets we can now apply a data mining algorithm like the well known Apriori algorithm [10] to compute a ranked list of association rules. As an example we could obtain the following rule:

$$\{(c_5, 3)\} \Longrightarrow \{(c_2, 4), (c_3, 3)\} \tag{6.6}$$

This can be interpreted as: "Cluster label 3 of method c_5 corresponds to cluster label 4 of c_2 and cluster label 3 of c_3."

Now starting with the highest ranked rule we can process the list of rules by successively deducing our mapings this way until we have obtained all mappings (ignoring all mapping proposals from lower ranked rules which contradict previous, higher ranked mappings).

More precisely we iteratively construct an (undirected) *association graph* $G = (V, E)$, with nodes consisting of all $k \cdot l$ method-label pairs. To the initially empty set of edges E, we successively add edges which do not violate the consistency constraint that the label of one method must not be connected to more than one label of another method, as described in the pseudo code in Figure 6.1.

```
initialize graph G = (V,E) with k*l vertices (c_i,label_j)
, i=1,..,l  j=1,..,k  and E empty

while (G contains < k*(l-1) edges) do
  select next association rule :r_i: antecedent -> consequent
  for all (c_i,a) in antecedent, (c_j,b) in consequent do
     if (     ! Exists Label L with: Exists path from (c_i,L) to (c_j,b)
          and ! Exists Label L with: Exists path from (c_i,a) to (c_j,L)
        )
     then
        add {(c_i,a),(c_j,b)} to E
```

Figure 6.1: Construction of the Association Graph for Meta Mapping

Figure 6.2 shows an example of an association graph. We assign the same meta label to all cluster labels which lie on a path in the graph.

Like for method A we can apply here also a greedier approach by computing the mapping for pairs of clustering methods and using transitivity to find the overall mapping.

The introduced approaches were designed for methods with constant pre-defined number of resulting clusters (e.g. partitioning-based clustering algorithms, such as k-means). The generalization for methods with a variable number of clusters, or

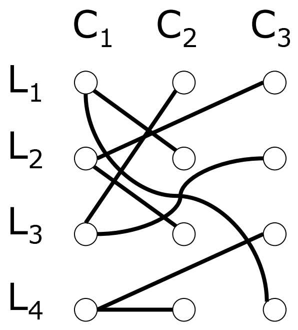

Figure 6.2: Association Graph for $l = 3$ Clustering Methods and $k = 4$ Clusters

hierarchical clustering approaches is subject of our current work.

6.2.2 Metafunctions

After having computed the mapping we are given a set $C = \{c_1, \dots, c_l\}$ of l clustering methods with results $res_i(d)$. For simplicity we consider here the case of $k = 2$ clusters and choose $res_i(d) \in \{+1, -1\}$ for a document d, namely, $+1$ if d is assigned to cluster 1, and -1 if d is assigned to cluster 2. We can combine these results into a meta result: $Meta(d) = Meta(res_1(d), \dots, res_l(d))$ in $\{+1, -1, 0\}$ where 0 means abstention. This is analogous to combination of results in meta classification, as described in Section 4.2.

Given thresholds t_1 and t_2, with $t_1 > t_2$, and weights $w(c_i)$ for the l underlying clustering methods we compute $Meta(d)$ as follows:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{l} res_i(d) \cdot w(c_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{l} res_i(d) \cdot w(c_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

These considerations can be easily extended to the case of $k >= 2$ possible clusters, e.g., by computing the linear combination for each cluster label separately, and by assigning the label with the maximum value to the document d if this value is above some threshold.

Analogously to meta classification, the restrictive and tunable behavior is achieved by the choice of the thresholds as t_1 and t_2. For this family of meta methods we obtain the same special cases, as described in Section 4.2 for classification.

6.2.3 A Probabilistic Model for Metaclustering

In this subsection we develop a simplified probabilistic model (for $k = 2$) to a better understanding of why metaclustering works. Consider the unanimous-decision meta clustering method. We assume that we have found appropriate mappings map_i as described above. We associate a Bernoulli random variable X_i with each clustering method c_i, where $X_i = 1$ if c_i clusters a document correctly, 0 otherwise.

With these settings the analysis becomes completely analogous to the derivations in Section 4.4.3. We obtain:

$$P(X_1 = 1, \ldots, X_l = 1) = P(X_1 = 1) \prod_{i=1}^{l-1} \frac{P(X_i = 1)P(X_{i+1} = 1) + cov(X_i, X_{i+1})}{P(X_i = 1)} \tag{6.8}$$

and the analog expression for $P(X_1 = 0, \ldots, X_l = 0)$.

Finally we can substitute these results in the following formulas for the loss probability

$$\begin{aligned} loss(Meta) &= 1 - P(X_1 = \ldots = X_l) \\ &= 1 - (P(X_1 = 1, \ldots, X_l = 1) + P(X_1 = 0, \ldots, X_l = 0)) \end{aligned} \tag{6.9}$$

and the error and accuracy probability

$$\begin{aligned} error(Meta) &= P(X_1 = 0, \ldots, X_l = 0 | X_1 = \ldots = X_l) \\ &= \frac{P(X_1 = 0, \ldots, X_l = 0)}{P(X_1 = 1, \ldots, X_l = 1) + P(X_1 = 0, \ldots, X_l = 0)} \end{aligned} \tag{6.10}$$

$$accuracy(Meta) = 1 - error(Meta) \tag{6.11}$$

As an illustrative example we consider the case that the l clustering methods have the same probability $p < 0.5$ (i.e. the clustering method performs better than random) to mis-assign a document, and a covariance $c < p(1-p)$ (i.e. the clustering methods are not perfectly correlated). In this case we would obtain for *loss* and *error*:

$$loss = 1 - \left((1-p) \left(\frac{c + (1-p)^2}{1-p} \right)^{l-1} + p \left(\frac{c + p^2}{p} \right)^{l-1} \right) \tag{6.12}$$

$$error = \frac{p \left(\frac{c+p^2}{p} \right)^{l-1}}{(1-p) \left(\frac{c+(1-p)^2}{1-p} \right)^{l-1} + p \left(\frac{c+p^2}{p} \right)^{l-1}} \tag{6.13}$$

It is easy to show that for $l \to \infty$ the loss converges monotonically to 1, and the error to 0 (i.e. with more clustering methods we can obtain a lower error but pay the price of a higher loss). The covariance plays the role of a "smoothing constant": with higher correlated clustering methods the convergence of both loss and error is slowed down.

6.3 Combination of Restrictive Clustering with Supervised Learning

The partitioning produced by a restrictive meta algorithm is expected to have a higher accuracy than the results of the underlying (non-restrictive) base methods. However, the higher clustering quality is connected with the loss of some data. This situation is acceptable in precision-oriented applications (e.g. data filtering) but may cause problems in recall-sensitive cases. To overcome this limitation of the restrictive clustering, the filtered output of the meta algorithm can be considered as training input for supervised or semisupervised classification methods (e.g. Naive Bayes, SVM, transductive SVM, etc.). The latter method allows customizable partitioning of unlabeled data (usually proportional to sizes of labeled data sets). The collection of documents that were dismissed by meta clustering can be assigned to clusters using this new decision model. It is obvious that the combination of restrictive meta-clustering and classification acts as a non-restrictive clustering approach (with zero loss).

More formally, let $U = L \cup \bar{L}$ be the set of unlabeled data, where L is the set of documents which were assigned a cluster label by the restrictive meta clustering method, and $\bar{L}$ is the set of documents where the restrictive meta method abstained. Now we train a classifier C (e.g., an SVM classifier) on the training samples $\{(d, res(d)) | d \in L\}$, where $res(d)$ is the meta label for document d, i.e., we treat the meta labels like class labels in conventional classification. We use the obtained classifier C to assign cluster labels to the documents in $\bar{L}$, and thus obtain an overall clustering for all documents in U.

In Section 6.4, we show results of preliminary evaluations for meta clustering in connection with linear SVM and transductive SVM algorithms.

6.4 Experiments

6.4.1 Quality Metrics for Clustering

Our quality measure describes the correlation between the actual topics of our datasets and the clusters found by the algorithm. Consider that the clusterlabels can be permutated: Given two classes $class_1$ and $class_2$, it does not matter for example whether a clustering algorithm assigns label a to all documents contained in $class_1$ and label b to the documents contained in $class_2$ or vice versa; the documents belonging together are correctly put together and the quality should reach its maximum value 1 and the error should be 0.

Let k be the number of classes and clusters, N_i the total number of clustered documents in $class_i$, N_{ij} the number of documents contained in $class_i$ and having cluster label j. We define:

$$accuracy = \max_{(j_1,\ldots,j_k)\in perm((1,\ldots,k))} \frac{\sum_{i=1}^{k} N_{i,j_i}}{\sum_{i=1}^{k} N_i} \tag{6.14}$$

and

$$error = 1 - accuracy \tag{6.15}$$

As $loss$ we simply define the fraction of documents dismissed over the whole document set. We use the macro-average of loss and error as an aggregation measure for a larger number of experiments.

6.4.2 Setup

We performed a series of experiments with real-life data from

1. The Newsgroups collection described in Section 4.5.
2. The Reuters articles described in Section 5.5.
3. The Internet Movie Database (IMDB) described in Section 4.5.

In all discussed experiments, the standard bag-of-words model [17] (using term frequencies to build L1-normalized feature vectors) was used for document representation.

Our experiments capture the behavior of (restrictive) base clustering methods and meta clustering, for tuples of topics such as "Drama vs. Horror vs. Western" for

IMDB data or ”rec.autos vs. rec.motorcycles vs. rec.sport.hockey” for the Newsgroups data. For each data set we identified all topics with sufficiently many documents. These were 20 topics for Newsgroups, 15 for Reuters and 15 for the genres of IMDB documents. We randomly chose 50 topic tuples from Newsgroups, from IMDB, and from Reuters for every set of k-tuples ($k = 2, 3, 5$). Finally, we computed macro-averaged results for these topic tuples.

6.4.3 Results

In our experiments we considered the following base methods:

1. *base1*: k-means, no feature selection, preclustering [45] with $k * 20$ documents - a standard heuristics for this initialization phase: before starting the actual clustering algorithm, a clustering is computed on a much smaller subset
2. *base2*: iterative feature selection applied on k-means - Mutual Information, when applied to a document corpus, requires an apriori partitioning of documents into thematically coherent subsets. If this partitioning is not already given, a clustering algorithm can provide us with an initial approximation. This initial step may use either all features or a *df* based feature selection. Once we have clusters, we can compute MI values and identify the most discriminative features, and then we can iterate this procedure, alternating between feature selection and clustering. We developed an iterative clustering algorithm based on this approach, using k-means as the underlying base method. - , preclustering with $k*20$ documents on a preselected feature space (*df*), after each iteration: feature selection (step 1: top-2000 according to *df*, step 2: top-500 according to MI), number of iterations: 5
3. *base3*: transforming feature vectors using SVD - see Section 2.5.2 - (SVD rank = 2), application of k-means on the transformed vectors (We found that a higher SVD rank results in a lower clustering accuracy in consistence with observations made by [59].)

Of course, the introduced meta approach can be used with any other clustering methods as well.

Figure 6.3 shows the loss-error tradeoff for the base methods for $k = 3$ and $k = 5$: By inducing a loss as described in Section 6.1 we can obtain a significant reduction of the error.

	k = 3			**k = 5**			
	base1	base2	base2	base1	base2	base3	
loss	avg(error)	avg(error)	avg(error)	avg(error)	avg(error)	avg(error)	**Dataset**
0.0	0.301	0.324	0.336	0.489	0.502	0.556	
0.1	0.289	0.316	0.319	0.486	0.487	0.540	
0.2	0.284	0.308	0.304	0.487	0.486	0.535	
0.3	0.279	0.302	0.284	0.485	0.488	0.528	
0.4	0.274	0.302	0.277	0.475	0.491	0.503	IMDB
0.5	0.260	0.295	0.260	0.465	0.478	0.466	
0.6	0.256	0.279	0.227	0.439	0.465	0.434	
0.7	0.244	0.269	0.213	0.416	0.447	0.378	
0.8	0.204	0.254	0.165	0.382	0.398	0.303	
0.9	0.178	0.199	0.098	0.329	0.365	0.213	
0.0	0.346	0.332	0.317	0.430	0.403	0.572	
0.1	0.337	0.321	0.303	0.416	0.390	0.551	
0.2	0.325	0.311	0.293	0.403	0.380	0.521	
0.3	0.315	0.302	0.282	0.386	0.369	0.486	
0.4	0.302	0.294	0.272	0.371	0.363	0.451	Newsg.
0.5	0.286	0.289	0.260	0.351	0.356	0.413	
0.6	0.266	0.278	0.229	0.328	0.347	0.365	
0.7	0.251	0.270	0.186	0.302	0.327	0.303	
0.8	0.237	0.259	0.142	0.270	0.294	0.234	
0.9	0.209	0.341	0.088	0.224	0.233	0.156	
0.0	0.219	0.221	0.292	0.211	0.205	0.348	
0.1	0.256	0.221	0.315	0.220	0.202	0.359	
0.2	0.246	0.242	0.317	0.210	0.191	0.358	
0.3	0.224	0.233	0.267	0.194	0.176	0.362	
0.4	0.207	0.218	0.258	0.170	0.171	0.342	Reuters
0.5	0.193	0.234	0.207	0.162	0.144	0.314	
0.6	0.175	0.214	0.191	0.171	0.163	0.291	
0.7	0.144	0.193	0.204	0.166	0.144	0.307	
0.8	0.176	0.162	0.181	0.138	0.132	0.269	
0.9	0.359	0.149	0.191	0.094	0.080	0.150	

Figure 6.3: Restrictive Base Methods for k = 3, k = 5 on Reuters, Newsgroups and IMDB

With the three base methods we built a restrictive meta clustering model based on the "unanimous decision" function and the 3 different meta-mappings described in Section 6.2.1 namely:

1. *MapA*: correlation-based mapping
2. *MapB*: purity-based mapping
3. *MapC*: mapping using association rules

k = 3									
	Meta		**restrictive Base**			**Base**			
			base1	base2	base3	base1	base2	base3	
Map	avg (loss)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	**Dataset**
MapA	0.497	0.234	0.275	0.277	0.250				
MapB	0.542	0.229	0.276	0.274	0.232	0.339	0.312	0.337	IMDB
MapC	0.458	0.236	0.287	0.281	0.263				
MapA	0.420	0.242	0.269	0.304	0.255				
MapB	0.479	0.199	0.255	0.291	0.242	0.341	0.326	0.317	Newsg.
MapC	0.413	0.240	0.268	0.300	0.257				
MapA	0.408	0.133	0.193	0.242	0.243				
MapB	0.638	0.130	0.170	0.233	0.290	0.179	0.215	0.300	Reuters
MapC	0.365	0.190	0.209	0.255	0.268				

k = 5									
	Meta		**restrictive Base**			**Base**			
			base1	base2	base3	base1	base2	base3	
Map	avg (loss)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	avg (error)	**Dataset**
MapA	0.704	0.346	0.400	0.441	0.376				
MapB	0.800	0.361	0.375	0.413	0.292	0.506	0.470	0.559	IMDB
MapC	0.636	0.427	0.438	0.467	0.433				
MapA	0.622	0.320	0.316	0.330	0.348				
MapB	0.758	0.264	0.286	0.281	0.264	0.439	0.403	0.578	Newsg.
MapC	0.567	0.341	0.329	0.339	0.378				
MapA	0.623	0.103	0.142	0.140	0.333				
MapB	0.735	0.111	0.136	0.111	0.290	0.222	0.194	0.351	Reuters
MapC	0.571	0.150	0.155	0.163	0.351				

Figure 6.4: Metaclustering Results for k=3 and k=5 on Reuters, Newsgroups and IMDB

We compared the meta results with the results of the underlying base methods and the restrictive base methods (inducing the same loss as the meta method in each experiment). The results are shown in Figure 6.4. The results clearly show that the meta approach provides a lower error than its underlying base methods at the cost of moderate loss. More important, the meta method performs typically better than the restrictive version of each base method for the same loss. Figure 6.4 also shows that mappings produced by particular meta algorithms are not always identical; this leads to different losses on the same dataset.

Although all proposed meta-mapping algorithms perform fairly well and predict the same mapping in most of the cases, there are marginal differences for experiments on topics that are very different in size (especially Reuters). In such cases the overlap mapping (MapA) tends to produce slightly better results. The purity-based mapping (MapB) is more stable in experiments with fairly comparable class sizes (Newsgroups

Base Methods			**Meta-Method**		**Comb. Supervised Learning**		**Dataset**
base1	base2	base2			svm	tsvm	
avg(error)	avg(error)	avg(error)	avg(loss)	avg(error)	avg(error)	avg(error)	
0.246	0.255	0.231	0.409	0.159	0.228	0.204	IMDB
0.260	0.252	0.299	0.347	0.243	0.281	0.313	Newsg.
0.121	0.209	0.291	0.328	0.076	0.188	0.207	Reuters

Figure 6.5: Combination of Restrictive Clustering and Supervised Learning in Comparison with underlying Base Methods for k=2

and IMDB). The underlying Apriori algorithm of the association rule-based mapping (MapC) has calibration parameters (such as the delta value by which the minimum support is decreased in each iteration) that, depending on properties of the current data, may lead to slightly less accurate results. The optimal, task-dependent, parameterization of this algorithm is subject of our future work.

To test the combination of clustering and supervised or semisupervised learning we performed a restrictive metaclustering for $k = 2$ (using *MapB*) as described in Section 6.2.1. The obtained new partitioning for the whole document set (i.e. loss = 0) was compared to the clusterings provided by the underlying non-restrictive base methods. The evaluation is shown in Figure 6.5. Although the partitioning provided by restrictive meta-clustering has high accuracy and results in many cases in good training sets and accurate classifiers, the high loss (and small training sets) causes, in particular experiments, reduced generalization performance leading to moderate average accuracy.

6.4.4 Discussion

The restrictive meta methods result in higher cluster purity. The introduced algorithms of meta mapping ensure better accuracy and make clustering results robust and accurate at the cost of moderate loss of uncertain data.

We have experimentally shown that meta clustering has higher accuracy than particular clustering methods, and, more importantly, in almost all cases performs better than the restrictive version of each underlying base method with the same loss.

6.5 Meta Clustering using Confidence Values for Clustering

Up to now, we have assumed that each document is assigned to exactly one cluster. We drop this assumption now, and consider "soft" assignments of cluster labels. For example EM clustering [84, 58] assigns to a document the probabilities of membership to the different clusters. Confidence values for cluster memberships can be also assigned to the results of other clustering algorithms, e.g., k-means.

6.5.1 Confidence Values of Cluster Membership for k-Means

The k-means algorithm provides us with a set of k centroids $\{z_1, \dots, z_k\}$. Centroid z_i is a representation of cluster i. Each document d is assigned to the cluster i for which the similarity $sim(d, z_i)$ between z_i and the document d is maximized (e.g., in terms of cosine similarity).

But the similarities $sim(d, z_i)$ can also be used to make soft assignments. The simplest way to do this is to assign to each document d a confidence value $c(i, d)$ for the membership to cluster i proportional to $sim(d, z_i)$:

$$c(i, d) = \frac{sim(d, z_i)}{\sum_{j=1}^{k} sim(d, z_j)} \tag{6.16}$$

We have chosen the normalization constant such that:

$$\sum_{j=1}^{k} c(j, d) = 1 \tag{6.17}$$

6.5.2 Generalized Meta Clustering Problem

For meta clustering we are given a set $C = \{c_1, \dots, c_l\}$ of clustering methods, and a set U of unlabeled documents. Each method c_i assigns to a cluster label j in $\{1, \dots, k\}$ and to a document d in U a cluster confidence:

$$c_i(j, d) = c_{ij}(d) \tag{6.18}$$

with

$$\sum_{j=1}^{k} c_{ij}(d) = 1 \text{ for all } i = 1, \dots, l \tag{6.19}$$

Our objective is to find l meta mappings

$$map_i : \{1, \dots, k\} \rightarrow \{1, \dots, k\} \tag{6.20}$$

that map the cluster labels of each method c_i to a meta label. Furthermore we aim to determine a meta function

$$meta(d) = meta(c_{11}(d), \dots, c_{lk}(d), map_i, \dots, map_k) \tag{6.21}$$

that assigns to each document d a meta label in $\{1, \dots, l\} \cup \{0\}$ ("0" means abstention, as described above).

6.5.3 Meta Mapping

Correlation-based Mapping Now we generalize the correlation-based mapping, described in Section 6.2.1, for the occurrence of soft assignments. We define:

$$|A_{ij}| = \sum_{d \in U} c_{ix}(d) \tag{6.22}$$

with

$$x = map_i^{-1}(j) \tag{6.23}$$

We define the intersection, $|A_{ij} \cap A_{mj}|$, as follows:

$$|A_{ij} \cap A_{mj}| = \sum_{d \in U} c_{ix}(d) \cdot c_{my}(d) \tag{6.24}$$

with

$$x = map_i^{-1}(j) \text{ and } y = map_m^{-1}(j) \tag{6.25}$$

Note that for the special case of a "hard" assignment of documents to clusters, i.e,

$$\begin{aligned} c_{ij}(d) = 1 \text{ for exactly one } j \in \{1, \ldots, k\}, 0 \text{ otherwise} \\ \text{for all } i \in \{1, \ldots, l\}, d \in U \ , \end{aligned} \tag{6.26}$$

these definitions are equivalent to those given in Section 6.2.1.

Now we can define the optimization problem exactly as in Section 6.2.1. For the overlap we have:

$$overlap(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{6.27}$$

We obtain the mapping by solving the following optimization problem:

$$\begin{gathered} \text{Maximize over } (map_1, \ldots, map_l) : \\ \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\binom{l}{2}} \sum_{(i,m) \in \{1,\ldots,k\}^2, i<m} overlap(A_{ij}, A_{mj}) \end{gathered} \tag{6.28}$$

Mapping with Association rules For the case of hard cluster assignment (see Section 6.2.1), the item sets used for finding association rules for cluster labels were in the following form:

$$\{(c_1, c_1(d)), (c_2, c_2(d)), \ldots, (c_l, c_l(d))\} \text{ with } c_i(d) \in \{1, \ldots, k\} \tag{6.29}$$

In case of soft cluster assignment, we obtain for each document a "pseudo item set" with $l \cdot k$ elements:

$$\{(c_1, c_{11}(d)), (c_1, c_{12}(d)), \ldots, (c_l, c_{lk}(d))\} \text{ with } c_{ij}(d) \in [0,1] \tag{6.30}$$

For these kinds of item sets we cannot perform a classical association mining algorithm. To transform these sets into a standard form we 1) perform a discretization of the values $c_{ij}(d)$ and 2) perform a duplication of itemsets according to the discretization. For the discretization we divide the interval $[0,1]$ into q equidistant subintervals. The values $c_{ij}(d)$ are mapped to $\{0,\ldots,q\}$ by

$$discr(c_{ij}(d)) = round(c_{ij}(d) \cdot q) \ , \tag{6.31}$$

where the *round*-function rounds a number to an integer value. The new pseudo item sets are:

$$\{(c_1, discr(c_{11}(d))), (c_1, discr(c_{12}(d))), \ldots, (c_l, discr(c_{lk}(d)))\} \\ \text{with } discr(c_{ij}(d)) \in \{0,\ldots q\} \tag{6.32}$$

Using this discretization, we construct standard item sets as follows:

$$\begin{aligned} &\text{for all } (j_1,\ldots,j_l) \in \{1,\ldots,k\}^l \\ &\quad \text{construct } \prod_{i=1}^{l} discr(c_{ij_i}(d)) \text{ association rules of the form:} \\ &\quad \{(c_1, j_1), \ldots, (c_k, j_l)\} \end{aligned} \tag{6.33}$$

Since we obtain an exponential number of item sets this way, for efficiency reasons, the proposed method is only suitable for small numbers k of clusters and l of clustering methods.

6.5.4 Meta Functions

Having obtained the mapping map_i as described above, we can compute the meta labels for clustering method c_i applied to document d as:

$$res_i(d) = map_i(\arg\max_x c_{ix}(d)) \text{ for all } i \in \{1,\ldots,l\} \tag{6.34}$$

and compute the meta result by a linear combination of these results as described in Section 6.2.2.

Alternatively, we can take the confidence values not just for the mapping, but also for the meta function into account, for instance, by replacing the $res_i(d)$ with the confidence values $\max_x c_{ix}(d)$.

7 Restrictive Methods and Meta Methods in Peer-to-Peer Systems

In this chapter we apply the meta classification and clustering approach, described in the previous chapters, in the context of peer-to-peer (P2P) systems. We consider this approach for distributed Web exploration applications like focused crawling. Typical applications are user-specific classification of retrieved Web contents into personalized topic hierarchies as well as automatic refinements of such taxonomies using unsupervised machine learning methods (e.g. clustering). Our approach is to combine models from multiple peers and to construct an advanced decision model that takes the knowledge of multiple P2P users into account.

7.1 System Architecture

The implementation of a peer in our distributed system consists of two layers. The *lower (network) layer* determines the communication among the peers. The peers form an autonomous agent environment: the exact way one particular peer solves its Web retrieval problem (e.g. crawling the Web, sending queries to 'Deep Web' portals, analyzing recent newsgroup discussions or publications in electronic journals, etc.) is not restricted in any way. We assume that all peers share the same thematic taxonomy such as *dmoz.org* [2]. The *upper (application) layer* is the distributed algorithm that utilizes results from particular peers to construct improved learning models (e.g. classification and/or clustering models) that can be used to continue the focused crawl with higher accuracy and to adjust the topics of a user-specific personalized ontology.

7.2 Properties of the Network Layer

In our model, the peers use the epidemic-style communication [37]. Every peer maintains an incomplete database about the rest of the network. This database contains entries (e.g. addresses) on some other peers (neighbors) together with timestamps of the last successful contact to that neighbors. The neighbor list is refreshed using a push-pull epidemic algorithm.

To connect a new peer to the network one needs only one living address. The database of the new peer is initialized with the entry containing this living address only, and the rest is taken care of by the epidemic algorithm. Removal of a peer does not require any administration at all.

When new data becomes available, the peer initiates the building of a new meta learning method together with its direct neighbors as described in Section 7.3.1. With the next epidemic messages, it is broadcast to all neighboring peers.

7.3 Properties of the Application Layer

In this section we first describe a general framework for aggregating information from k peers in meta models, and then consider two typical applications for such a framework: classification and clustering of document collections.

7.3.1 Exchanging Data Models among Peers

In our framework we are given a set of k peers $P = \{p_1, \ldots, p_k\}$. Each peer p_i maintains its collection of documents D_i. The idea is to build concise individual models on each peer and then combine the models into a meta model. More formally, in the first step each peer p_i builds a model $m_i(D_i)$ using its own document set D_i. In the second step, the models m_i are propagated among the k peers as described in Section 7.2. To avoid high network load, it is crucial for this step that the models m_i are a very compressed representation of the document sets D_i. In the next step, each peer p_i uses the set of received models $M = \{m_1, \ldots, m_k\}$ to construct a meta model $Meta_i(m_1, \ldots, m_k)$. From now on, p_i can use the new meta model $Meta_i$ (instead of the 'local' model m_i) to analyze its own data D_i.

7.3.2 Application to Supervised and Unsupervised Document Labeling

Meta Classifiers on k Peers

In the context of classification algorithms, the introduced general approach 7.3.1 can be substantiated as follows. Each peer p_i contains a document collection D_i, consisting of a set of labeled training documents T_i and unlabeled documents U_i.

Every peer wants to automatically classify the documents in U_i. In the first step, every peer p_i builds its own feature vectors of topic labeled text documents T_i (e.g., capturing $tf \cdot idf$ weights of terms). The model m_i corresponds to the classifier obtained by running a supervised learning algorithm on the training set T_i.

Now, instead of transferring the whole training sets T_i, only the models m_i need to be exchanged among the peers. For instance, linear support vector machines (SVMs), as described in Section 2.2.3, construct a hyperplane $\vec{w} \cdot \vec{x} + b = 0$ that separates the set of positive training examples from a set of negative examples with maximum margin. For a new, previously unseen, document $\vec{d}$ the SVM merely needs to test whether the document lies on the "positive" side or the "negative" side of the separating hyperplane. The classifiers m_i can be represented in a very compressed way: as tuples $(\vec{w}, \vec{l}, b)$ of the normal vector $\vec{w}$ and bias b of the hyperplane and $\vec{l}$, a vector consisting of the encodings of the terms (e.g., some hashcode) corresponding to the dimensions of $\vec{w}$. $\vec{l}$ provides us with a synchronization between the feature spaces of the different peers.

Similar space saving representations are possible for other learning methods (e.g., Bayesian Learners). In addition, building the classifiers this way is much more efficient than building one 'global' classifier based on $T = \bigcup T_i$ because the computation is distributed among the peers, and for classifiers with highly nonlinear training time (e.g. SVM) the splitting can save a lot of time (see Chapter 4).

In the next step, every peer p_j considers the set $M = \{m_1, \ldots, m_k\}$ of k binary classifiers with results $R(m_i, d)$ in $\{+1, -1\}$ for a document $d \in U_j$, namely, +1 if d is accepted for the given topic by m_i, -1 if d is rejected. These results can be easily combined into a meta result:

$$Meta(d) = Meta(R(m_1, d), \ldots, R(m_k, d)) \in \{+1, -1, 0\} \tag{7.1}$$

A family of such meta methods is the linear classifier combination with thresholding as described in Section 4.2:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(m_i, d) \cdot w(m_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(m_i, d) \cdot w(m_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

The restrictive behavior is achieved by the choice of the thresholds: we dismiss the documents where the linear result combination lies between t_1 and t_2.

If a fixed set U of unlabeled documents (that does not change dynamically) is given, we can classify the documents with a user-acceptable loss of L as follows:

1. for all documents in U compute their classification confidence $\sum_{i=1}^{n} R(m_i, d) \cdot w(m_i)$
2. sort the documents into decreasing order according to their confidence values
3. classify the $(1 - L)|U|$ documents with the highest confidence values according to their sign and dismiss the rest

In our experiments we assigned equal weights to each classifier, and instead of $R(m_i, d)$, we considered a "confidence" value $conf(m_i, d)$ for the classification of document d by the classifier. For SVM we chose the SVM scores, i.e., the distance of the test points from the hyperplane. A more enhanced method to map SVM outputs to probabilities is described, e.g., in [97].

Note that meta classifiers can be, similar as base classifiers, easily transferred between peers as tuples

$$(m_1, \ldots, m_k, w(m_1), \ldots w(m_k), t_1, t_2). \tag{7.3}$$

Meta Clustering Algorithms on k Peers

For clustering, each peer p_i contains a document collection U_i of unlabeled data. Every peer wants to cluster its unlabeled data.

Analogously to the classification task every peer p_i can execute a clustering algorithm on its own data U_i to build the model m_i; a representation of the resulting clustering models m_i can be propagated to the other peers.

For the k-means algorithm (see Section 2.4.2), the clustering model m_i can be represented as $(\vec{z_1}, \ldots, \vec{z_l}, \vec{l})$, where the $\vec{z_i}$ are vector representations of the computed centroids, and $\vec{l}$ contains encodings of the feature dimensions, and provides us with a synchronization of the feature spaces, as described above for the supervised case.

Now we can apply a meta clustering algorithm as described in Section 6: First by applying a meta mapping, we obtain the meta labels $res_i(d) := map_i(m_i(d))$ for the clustering of document d with clustering model m_i. Then we can compute the overall meta result:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{k} res_i(d) \cdot w(m_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{k} res_i(d) \cdot w(m_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

Thus by an intermediate meta mapping step we have a completely analogous situation to the one for the supervised case described above: Class labels for classification correspond to meta labels for clustering. Confidence values $conf(m_i, d)$ for the clustering of a document d by the base methods m_i can be obtained, say for k-means clustering, by computing the similarity (e.g., using the cosine measure) to the nearest centroid. The restrictive behavior can be obtained in exactly the same way as for the supervised case.

Estimators and Tuning

For a restrictive meta classifier, we are interested in its behavior in terms of *error* and *loss* (fraction of unclassified documents). A typical scenario could be a number of users in different peers accepting a loss up to a fixed bound, to obtain a lower classification error for the remaining documents. In Section 4.4 the tuning of the number k of classifiers for a user-acceptable loss threshold was described. We will not repeat this here and will instead focus on the P2P specific aspects.

The main ingredients of the estimation and tuning process are:

1. estimators for base classifiers (based on cross-validation between the training sets T_i)
2. estimators for the pairwise correlations between the base classifiers $\{m_1, \dots, m_k\}$
3. probabilistic estimators for loss and error based on 1. and 2.

For the cross-validation, at least two peers, p_i and p_j, must cooperate: p_i sends a tuple $(m_i, IDs(T_i))$, consisting of its classifier m_i and a list of IDs (not contents!) of its training documents, to p_j. The peer p_j uses the list of submitted IDs to identify duplicates in both collections and performs cross-validation by m_i on $T_j - T_i$. (In the Web context, the IDs of T_i can be easily obtained by computing content-based 'fingerprints' or 'message digests' (e.g. MD5 [101])). The resulting error estimator (a simple numerical value) for m_i can be forwarded from p_j back to p_i or to other peers.

For the computation of pairwise covariance, at least three peers, p_i,p_j and p_m, must cooperate: p_i and p_j send their classifiers and document IDs to p_m, and p_m cross-validates in parallel both classifiers on $T_m - T_i - T_j$. By this procedure we also obtain error estimators.

Finally, the estimators for *covariance* and *error* (numerical values) can be distributed among the peers and estimators for the overall meta classifier can be built. When the estimated quality of the resulting meta classifier does not meet the application-specific peer requirements (e.g. the expected error is still above the specified threshold), the initiating peer may decide to invoke additional nodes for better meta classification. Note that for meta clustering, estimators *cannot* be built in the same easy way because for the unsupervised case, we cannot evaluate base methods by cross-validation.

7.4 Experiments

7.4.1 Setup

To simulate different P2P Web retrieval scenarios (crawling the Web, sending queries to 'Deep Web' portals, analyzing recent newsgroup discussions or publications in electronic journals) we performed multiple series of experiments with real-life data from

1. The academic WebKB dataset [34] that contains 8282 HTML Web pages from multiple universities, manually classified into the categories 'student', 'faculty', 'staff', 'department', 'course', 'project', and 'other'.
2. The Newsgroups collection described in Section 4.5.
3. The Reuters articles described in Section 5.5.
4. The Internet Movie Database (IMDB) described in Section 4.5.

We used the Porter stemming algorithm [99] in combination with stopword elimination to transform documents into the vector space model. In all discussed experiments, the standard bag-of-words approach [17] (using term frequencies to build L1-normalized feature vectors) was used for document representation.

7.4.2 Experiments with Supervised Learning Methods (Classification)

For each data set we identified all topics with more than 200 documents. These were 20 topics for Newsgroups, 6 for Reuters, 12 for IMDB, 4 for WebKB. Among these topics we randomly chose 100 topic pairs from Newsgroups and all possible combinations for the others, i.e. 66 topic pairs from IMDB, 15 for Reuters, and 6 for WebKB. For each topic pair we randomly chose 200 training documents per class and kept - depending on the available topic sizes in particular collections - a distinct and also randomly chosen set of documents for the validation of the classifiers.

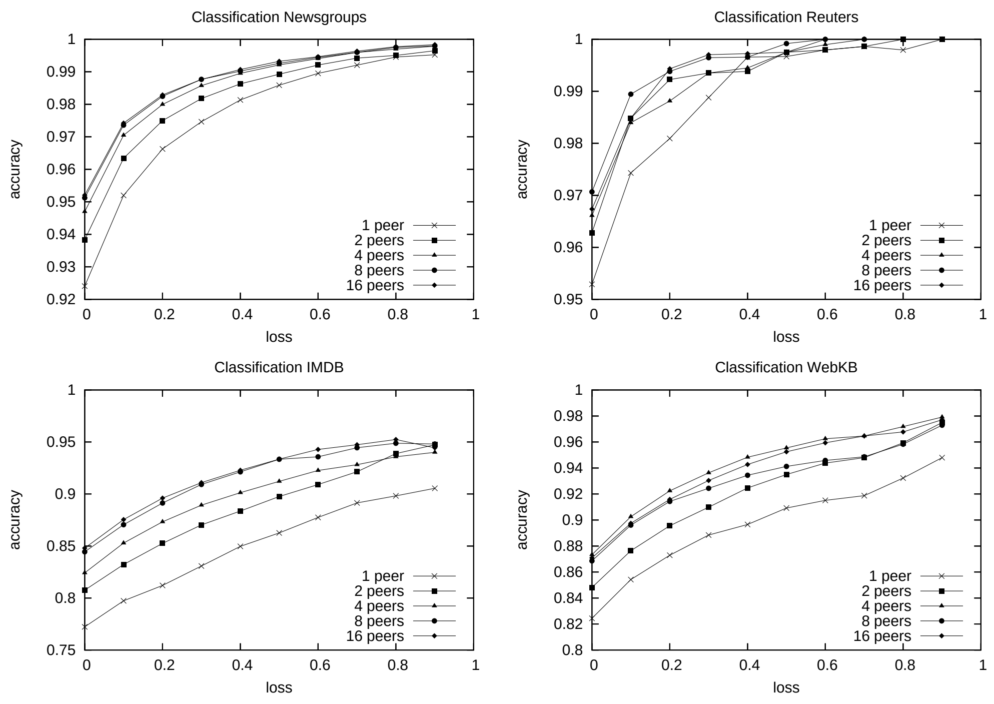

Figure 7.1: Results of Restrictive Meta Classification

In each experiment, the training data was distributed over 16 peers using equal-sized subsets with approximately 15% overlap (corresponding to peers that contain non-disjoint training data). Among these peers we randomly chose 1,2,4,8, and all 16 peers to simulate various P2P classification scenarios. We considered various numbers of cooperating peers that share their linear SVM classifiers and performed the meta classification on local subsets. The configuration with 1 cooperating peer corresponds to the 'local' peer classification that does not involve sharing of classifiers. As discussed in Section 7.3.2, we also compared the *restrictive* form of meta classification, where we dismissed at each peer exactly the same amount of documents with the lowest classification confidence using confidence values as discussed in Section 7.3. Our quality measure is the fraction of correctly classified documents (the *accuracy*) among the documents not dismissed by the restrictive algorithm. The *loss* is the fraction of dismissed documents.

Finally, we computed micro-averaged results along with their 95% confidence intervals for all groups of topic pairs. Figure 7.1 shows the observed dependencies between the numbers of cooperating peers, the induced loss, and the resulting accuracy for

	accuracy				
loss	1 peer	2 peers	4 peers	8 peers	16 peers
0.0	0.772 ± 0.007	0.808 ± 0.007	0.824 ± 0.006	0.844 ± 0.006	0.848 ± 0.006
0.1	0.797 ± 0.007	0.832 ± 0.007	0.853 ± 0.006	0.870 ± 0.006	0.875 ± 0.006
0.2	0.812 ± 0.007	0.853 ± 0.007	0.873 ± 0.006	0.891 ± 0.006	0.896 ± 0.006
0.3	0.831 ± 0.007	0.870 ± 0.007	0.889 ± 0.006	0.909 ± 0.006	0.911 ± 0.006
0.4	0.850 ± 0.008	0.884 ± 0.007	0.901 ± 0.006	0.921 ± 0.006	0.923 ± 0.006
0.5	0.863 ± 0.008	0.898 ± 0.007	0.912 ± 0.007	0.933 ± 0.006	0.933 ± 0.006
0.6	0.877 ± 0.009	0.909 ± 0.008	0.923 ± 0.007	0.936 ± 0.006	0.943 ± 0.006
0.7	0.891 ± 0.009	0.921 ± 0.008	0.928 ± 0.008	0.944 ± 0.007	0.947 ± 0.007
0.8	0.898 ± 0.011	0.939 ± 0.009	0.936 ± 0.009	0.949 ± 0.008	0.952 ± 0.008
0.9	0.905 ± 0.015	0.947 ± 0.012	0.940 ± 0.013	0.948 ± 0.012	0.944 ± 0.012

Figure 7.2: Classification Results: IMDB Collection

various reference collections. An example of our evaluation summary including 95% confidence intervals for the IMDB collection is shown in Figure 7.2 (we obtained analogous results for the remaining three collections).

It can be observed that the meta classification and restrictive meta classification by multiple cooperating peers clearly outperforms the single-peer solution for all settings of the user-defined *loss*, including the non-restrictive meta classification with $loss = 0$. The quality of the meta algorithm clearly increases with the number of participating peers. In general, the difference between the one-peer solution and the meta solution is statistically significant for 4 and more participating peers and all values of the induced loss. The only exceptions are the results for Reuters with $loss > 0.7$ (the accuracy of all peer combinations, including the one-peer experiment, becomes nearly 1.0) and the WebKB collection (due to the very limited number of possible topic combinations).

7.4.3 Experiments with Unsupervised Learning Methods (Clustering)

The same collections and topics were used to evaluate distributed meta clustering. All documents from randomly combined selections of 3 or 5 topics were considered as unlabeled data and distributed among peers analogously to classification experiments from the previous section, with approximately 15% overlap.

The goal of the clustering algorithm was to reproduce the partitioning into topics on each peer with as high accuracy as possible. Our quality measure describes the correlation between the actual topics of our datasets and the clusters found by the algorithm. Let k be the number of classes and clusters, N_i the total number of clustered documents in $class_i$, N_{ij} the number of documents contained in $class_i$ and having cluster label j. Unlike classification results, the clusters do not have explicit topic labels; we define the clustering accuracy as follows:

$$accuracy = \max_{(j_1,\ldots,j_k)\in perm((1,\ldots,k))} \frac{\sum_{i=1}^{k} N_{i,j_i}}{\sum_{i=1}^{k} N_i} \tag{7.5}$$

The *loss* is the fraction of documents dismissed by the restrictive algorithm.

For all peers, k-means was used as the underlying base method. We compared the one-peer clustering (i.e. clustering that can be executed by one peer on its local dataset without cooperation with others) with meta clustering by exchanging centroids from cooperating peers and correlation-based mapping (Chapter 6) of the final clusters. Analogously to classification experiments, we considered restrictive meta clustering, dismissing exactly the same number of documents with the worst clustering confidence on each peer.

The results are summarized in Figure 7.3. The main observations are similar to the ones discussed for the supervised case:

- The quality of the meta clustering results is consistently higher than for isolated one-peer solutions.
- The quality of the meta algorithm tends to increase with the number of participating peers and is in almost all cases statistically significant. For the Reuters collection, the difference between one-peer solution and the meta result is statistically significant for 8 and more participating peers and all values of the induced loss. For the IMDB and Newsgroups collections, the difference between the one-peer solution and the meta result is statistically significant for 4 and more participating peers and all loss values.

In the experiments with the Reuters dataset, the accuracy decreases for high loss values (greater 0.7). Possibly this can be explained by the fact that the Reuters topics - unlike the other considered reference collections - are very different in size (e.g. the topics '*earn*' and '*grain*' contain about 3900 and 280 documents, respectively). The in-depth analysis of such artifacts is subject of our future work.

In systematic evaluations of further application scenarios (Figure 7.4 shows an example for clustering with $k = 5$ topics on the IMDB collection) we observed similar results.

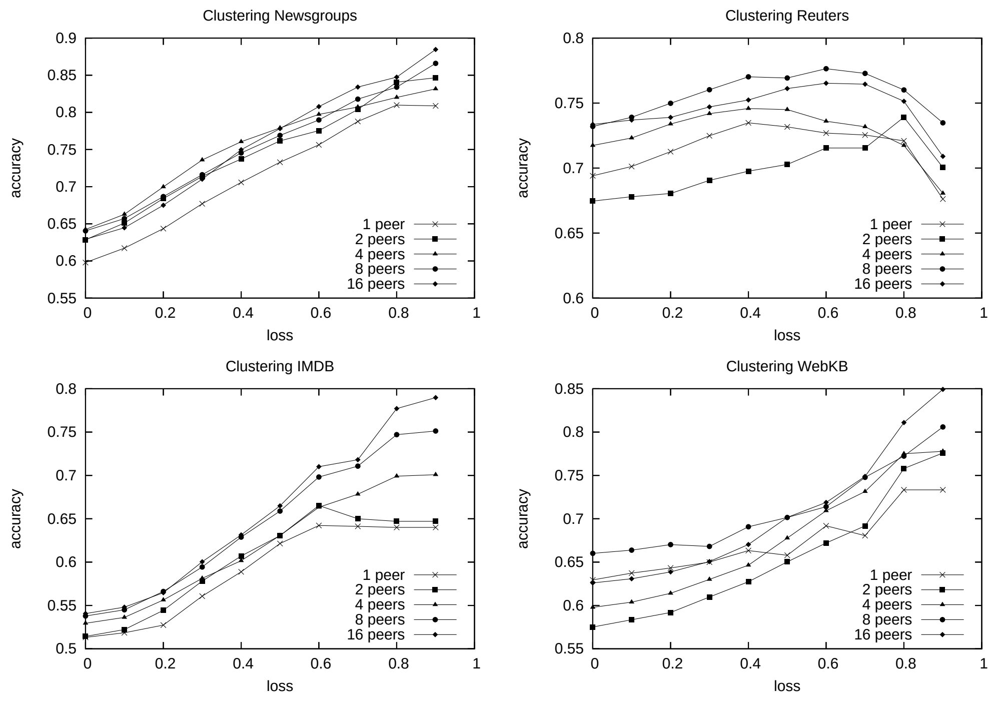

Figure 7.3: Results of Restrictive Meta Clustering, k=3 Clusters

7.4.4 Discussion

The accuracy of our restrictive meta methods outperforms the models that can be built separately on training sources of isolated peers and - more importantly - also the restrictive variant of such one-peer solutions with same induced loss. This holds for the supervised as well as for the unsupervised case.

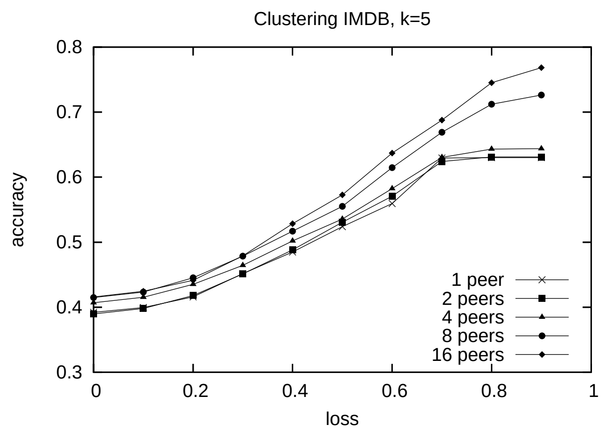

Figure 7.4: Clustering Results for k=5 Clusters, IMDB

8 Construction of Feature Spaces and their Combination

In this chapter we describe document representations and their combination for automatic classification for two different domains: Web document classification and author recognition.

8.1 Web Document Classification

Classification of Web content is one of the most important tasks in Data Mining. Conventional classification strategies apply statistical learning methods to purely term-based feature vectors. In this section we want to describe alternative features based on linked neighbors of a document, structural characteristics, etc. We combine the different document representations using meta classification.

8.1.1 Word Pairs

In many documents, words form compound expressions (e.g. "network protocol") or words typically co-occur in a certain context. The idea is to combine these words into tuples (*n-grams*).

To consider all possible n-tuples of words would result in an exponential explosion of the feature space. Instead, we consider the case $n = 2$ (*pairs*). This can be justified by the fact that higher dimensional tuples often do not occur in a significant number of documents. Furthermore, a correlation of N terms also implies a correlation of the $\binom{N}{2}$ possible term pairs.

Formally we represent a document d as set of tuples (t, pos), where t is a term in the document and pos is its position in the document (starting with position 1 for the first term in the document). Thus, considering term pairs, a document can be represented as a set of tuples

$$(t_a, t_b, pos_a, pos_b) \tag{8.1}$$

Here t_a and t_b are the terms and pos_a and pos_b are the positions of t_a and t_b respectively in the document. We appoint that $t_a \neq t_b$, i.e. we do not associate a term with itself.

We can restrict our term space by restricting to a set of terms $\tilde{T} = \{t_1, \ldots, t_q\}$, obtained, e.g., by feature selection.

For building pairs we can use a *sliding window strategy*, i.e., we consider only those pairs that are contained in a virtual window of size dis sliding over the text of the document:

$$\forall (t_a, t_b, pos_a, pos_b) \in d : |pos_a - pos_b| < dis \tag{8.2}$$

This way we avoid having too many pairs from different contexts.

For a pair $\{t_a, t_b\}$ we could now simply count the number of tuples (t_a, t_b, pos_a, pos_b) in a document d. But this could produce the following situation: For a term-position tuple (t_a, pos_a) there could exist two term-position pairs (t_b, pos_b) and (t_b, pos'_b) such that $pos'_b > pos_b$, i.e., we would count the term pair $\{t_a, t_b\}$ twice. But we assume, that in most of these cases, t_a should be associated with the nearest of the two terms t_b. We do not want to count the other pairs which we call the *pseudo pairs*, $Pseudo(d)$, of document d.

Thus for a pair $p = \{t_a, t_b\}$ we define its absolute frequency in document d as:

$$\begin{aligned} h_{abs}(p, d) \quad = \quad & \Big| \{(w_s, w_t, k, l) \in d \quad | \qquad\qquad (8.3) \\ & w_s = t_a \wedge w_t = t_b \wedge (w_s, w_t, k, l) \notin Pseudo(d) \wedge |k - l| < dis\} \Big| \end{aligned}$$

The components of the feature vector consist of the *relative* frequencies of each pair p:

$$tf(p, d) = \frac{h_{abs}(p, d)}{\sum_{q \in d} h_{abs}(q, d)} \tag{8.4}$$

Having obtained a set of pair-features, we can apply of course feature selection techniques as described in Section 2.5.1.

8.1.2 Anchor Terms

Anchor terms are the terms in the text belonging to the links in a HTML-page. They often provide us with a concise description of the linked Web page and therefore can be useful for automatic classification. We consider the concatenation of the anchor texts from the links pointing to the document of interest as one virtual document; thus we can consider the relative frequencies of the anchor terms as components of our feature vectors.

8.1.3 Document Union

For the *document union* strategy we consider the (linked) neighbors of a document d in the Web. Neighbor documents of document d are often related to d in terms of category and content. There are different ways to construct feature vectors for document d using its neighbors:

1. concatenate d and all of its neighbors into a single virtual document and compute the relative term frequencies
2. compute the relative term frequencies for d and its neighbors, and for each term compute an averaged term frequency as component of the feature vector

For both strategies we can apply different weighting schemes for document d and its neighbors.

8.1.4 Combining Features

In the previous section we have described several different representations of Web documents. In this section we describe two approaches to put these pieces of information together.

Combination Vectors

We can mix up different document representations by building combination vectors. One way to do this is to concatenate the components of different feature vector representations. For instance let $\tilde{T} = \{t_1, \ldots, t_n\}$ be our term space and $P = \{p_1, \ldots, p_m\}$ be our pair space. We can represent document d by a combination vector as follows:

$$(tf(t_1, d), \ldots, tf(t_n, d), tf(p_1, d), \ldots, tf(p_m, d)) \tag{8.5}$$

Features of the same type (such as text terms and anchor terms) can be combined alternatively by computing a weighted linear combination of their components.

Meta Classification

With a given set $V = \{v_1, \ldots, v_k\}$ of k binary classifiers (obtained by using different documents representations) we can perform meta classification using a linear classifier combination with thresholding, as described in Section 4.2:

category	number of training documents
Business and Economy	47
Computers and Internet	45
Health	44
Sports	46

Table 8.1: Categories and Number of Training Documents

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \quad (8.6)$$

8.1.5 Experiments

Setup

In our experiments we used the BINGO! [116] focused crawler for the acquisition of Web documents. The training categories and the number of training documents per class are listed in Table 8.1.

By manual evaluation of the categorized Web pages we obtained the accuracy, the fraction of correctly classified documents.

Results

We compared SVM classification applied to the following feature representations:

1. simple word-based features without feature selection (**AllTerms**)
2. word-based features with feature selection by MI (100 features for a given class) (**SelTerms(100)**)
3. word-based features and "contrast features" with feature selection by MI (100 features for a given class, 100 features for the other classes) (**ContrTerms(100,100)**)
4. document union with feature selection as in 2.) (**UnionA**)
5. document union with feature selection as in 3.) (**UnionB**)

strategy	classified	correctly classified	accuracy
AllTerms	242	231	95,45 %
SelTerms(100)	241	215	89,21 %
UnionA	242	218	90,08 %
UnionB	241	214	88,80 %
Pairs(pre:100,SW:4)	213	189	88,73 %
Pairs&Terms(pre:100,SW:4)	241	214	88,79 %
Anchors	85	63	74,12 %
Meta	183	182	99,45 %

Table 8.2: Classification Results for Web Pages with different Feature Spaces and their Combination

6. pair-based features; sliding window size dis = 4; pre-selection of Terms using MI (100 terms for a given class) (**Pairs(pre:100,SW:4)**)
7. combined feature vector on terms and pairs (terms as in 2., pairs as in 6.) (**Pairs&Terms(pre:100,SW:4)**)
8. Anchor Terms with the features selected in 2. (**Anchors**)
9. Meta Classification using a "Unanimous Decision"-strategy (**Meta**)

We did not classify documents which were transformed into a null vector. For meta classification we did not consider anchor terms, because here we obtained too many null vectors.

Table 8.2 shows the results of the experiments.

We can observe that meta classification outperforms the single methods in terms of accuracy.

8.2 Stylistic and Linguistic Features for Author Recognition

In this section, we provide several alternatives to the classical Bag-Of-Words model for automatic authorship attribution. To this end, we consider linguistic and writing style information such as grammatical structures to construct different document representations. Furthermore we describe two techniques to combine the obtained representations: combination vectors and ensemble based meta classification.

8.2.1 Word-Based Features

Using word based features is the most popular and, despite of its simplicity, very effective feature construction method. We briefly describe several variants from the literature that we will consider as a baseline for other methods.

Bag-Of-Words

In the Bag-Of-Words approach the ordering of the words is not considered. Optionally a stopword list can be used to eliminate very common terms like articles, prepositions, etc. Often additional techniques like stemming [99] are applied to the words. There are different options to construct feature weights: taking the absolute or relative frequency of term occurrences as components, constructing a binary feature vector by just considering the pure occurrence of a term, computing the tf*idf values of the terms, etc. [90, 103]. Because it is the state-of-the-art method for feature construction in automatic document classification we will consider Bag-Of-Words as a baseline for our experiments.

Function Words

In case of authorship attribution it can make sense to use "content-free" features, i.e., terms that do not contain information about the document's content such as prepositions, pronouns, determiners etc. These terms are called *function words* (see e.g. Diederich et. al. [39]). In our implementation we regard all words other than nouns, verbs and adjectives as function words.

POS Annotation

In this approach, the part of speech (POS) group of the words (e.g. verb, noun, adjective) is taken into account [95]. This can be used to filter documents, e.g., by considering only nouns or verbs. POS is also used for simple disambiguation, e.g., by distinguishing the verb "book" from the noun "book".

Feature Selection

The idea of feature selection is to just take the most discriminating features into account. Intuitively a well discriminating term for two classes A and B occurs frequently in documents of class A and infrequently in documents of class B or vice versa. Examples of feature selection measures are Mutual Information, Information Gain, and Chi Square [131].

Semantic Disambiguation

Here a thesaurus, e.g. Wordnet [47], is used to disambiguate terms (treating synonyms like "automobile" and "car" as the same feature). In some approaches also more complex relationships between words are taken into account [102, 107].

8.2.2 Using Linguistic Constituents

The structure of natural language sentences shows that word occurrences follow a specific order, called word order. Words are grouped into syntactic units (*constituents*) that can be deeply nested. Such constituents can be detected by their ability to occur in various positions and showing uniform syntactic possibilities for expansion (see [84]). Consider the sentence *Next, he examined the framework of the door we had broken in, assuring himself that the bolt had really been shot.* and its syntax tree representation in Figure 8.1. In particular, consider the part *he examined the framework.* This part is a constituent of the sentence with sub-constituents, e.g. *"the framework"*. The sub-constituents can change their positions inside the bigger constituent. Just considering that specific part, *he examined the framework* has the same meaning as *the framework he examined.* We can use this information about the word relationships by extracting constituents for feature construction. To this end, we first subdivide the document into sentences, and then construct a syntax tree as shown in Figure 8.1 for each sentence (note that the grey-boxed parts belong to another technique, the writing style, described in 8.2.4). In our framework, we use the Connexor Machinese Phrase Tagger [120] to subdivide a document into sentences and Lexparser [5] to build the syntax trees. We define a minimal and maximal length, min and max, of the constituents that we want to use for feature construction.

The simplest way to construct features would be to just concatenate the features inside a constituent using an appropriate separation character (e.g. "$").
For our example sentence, this would result in features such as $he\$examined\$the\$framework$. But such very specific features may occur very rarely in the document corpus. To obtain more "common" features a combination of some of the following options is applied:

- Performing stemming and stopword elimination on the words contained in a constituent.

- Abstracting from the ordering of words by putting the words into lexicographic order.
- Instead of a feature $x_1\$x_2\$\ldots\$x_n$ consider pairs $x_i\$x_j$ or triples contained in the constituents (bi- and trigrams).
- Performing feature selection on the constituent-features themselves. This can be done completely analogously to the feature selection for simple words.

In Section 8.2.7, we provide experiments on using whole constituents with stemming and stopword-elimination as well as using bigrams (also stemmed and stopword-cleaned).

8.2.3 Functional Dependencies

Functional dependencies represent relational information in sentences. Consider again a part of the sentence used in Figure 8.1, *he examined the framework of the door*. Here, *he* is the subject (agent) and *framework* and *door* are the objects of the predicate *examined* (action). We used the Connexor Machinese Syntax [120] to determine such dependencies. Our features have the form

$$x_1\$x_2\$\ldots\$x_n \tag{8.7}$$

where x_1 is the subject of an action, x_2 is the predicate and x_3 through x_n are the objects. To obtain a canonical form, words are reduced to their base forms, using the Connexor Machinese Phrase Tagger [120], and objects are sorted in lexicographic order. In our example case, we get the feature $he\$examine\$door\$framework$.

8.2.4 Writing Style: Using Syntax Trees

Different authors may construct sentences in their writings in a completely different way. The idea is to consider a syntax tree representation of their sentences as features. In the extreme case we could encode the whole tree into a string; but this would result in very sparse feature spaces. Instead we should restrict ourselves to nodes up to a certain maximum tree depth. In our experiments we observed that considering just the children of the root nodes of sentences and sub-clauses (labeled with S) provides us already with interesting features. So our example tree in Figure 8.1 could be encoded into the features $ADVP\$,\$NP\$VP$, VP, and two times $NP\$VP$ (emphasized by the grey boxes). Note that this method does not use any word information

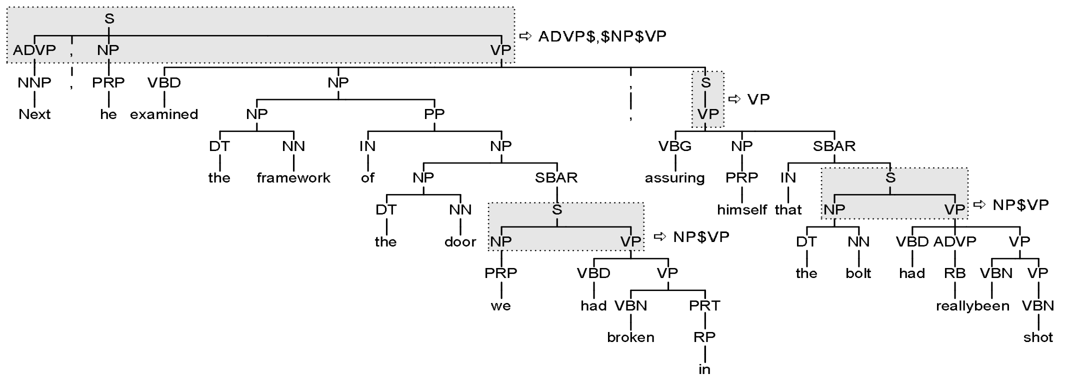
⇨ ADVP$,$NP$VP
⇨ VP
⇨ NP$VP
⇨ NP$VP
Next , he examined the framework of the door we had broken in , assuring himself that the bolt had really been shot

Figure 8.1: PCFG-Tree and Writing Style Features

at all. Table 8.3 shows the top-5 features for the authors A. C. Doyle and R. Burton according to their Mutual Information values (we considered only books available from the Gutenberg Project [3]; see Section 8.2.7 for details). We do not apply any kind of filtering mechanism to the structure features such as removing punctuation marks. Experiments showed that those marks provide interesting information about the sentence structure. Note, that "," in the feature $ADVP\$,\$NP\$VP$ represents an annotation tag for a word phrase in the syntax tree, not the comma itself.

A.C. Doyle		**R. Burton**	
Feature	**MI**	**Feature**	**MI**
S$,$CC$S$.	0.23	S$:$S	0.26
PP$NP$VP$.	0.16	S$CC$S	0.23
SBAR$,$NP$VP$.	0.14	X$X$NP$VP	0.21
SBAR$,$X$NP$VP$.	0.13	S$:$S$.	0.20
PP$,$NP$VP$.	0.11	S$:$CC$S	0.18

Table 8.3: TOP 5 Writing Style Features according to MI

8.2.5 Syntax Tree Depth

Other research discovered the benefit of sentence length as feature either by computing the average sentence length [36, 39], or by using histograms over the sentence length [124]. Another simple but, to our knowledge, novel approach to distinguish different writing styles is to consider the depth of the syntax trees in the documents. We consider two approaches.

Statistical Moments

Statistical moments are one way to characterize a distribution. The *k-th moment* of a random variable X is defined as $E(X^k)$. The expression $E([X - E(X)]^k)$ is called the *k-th central moment* of X.

For a given document d, containing n sentences (and so n trees), we can approximate the k-th moment and the k-th central moment as follows:

$$E(X^k) = \frac{1}{n}\sum_{j=1}^{n} x_j^k \tag{8.8}$$

and

$$E([X - E(X)]^k) = \frac{1}{n}\sum_{j=1}^{n}[X - E(X)]^k \tag{8.9}$$

where x_i is equal to the syntax tree depth of the i-th sentence of document d. Note, that $E(X)$ is known as the exspectation value and $E([X - E(X)]^2)$ the variance of the random variable X.[1]

The values for different k vary in their order of magnitude. To avoid an overestimation of higher moments we take the k-th root of the k-th moment. For the construction of the feature vectors we consider the first three moments and the second and third central moments.[2] Thus we can represent our document d by the following vector:

$$\left(E(X), \sqrt{E(X^2)}, \sqrt[3]{E(X^3)}, \sqrt{E([X - E(X)]^2)}, \sqrt[3]{E([X - E(X)]^3)}\right) \tag{8.10}$$

Histogram Approach

The most common form of a histogram is obtained by splitting the range of the data into equal-sized bins. Then for each bin, the number of points from the data set that fall into the bin is counted [6].

In our scenario the data consists of the syntax tree depths of a document d. The value assigned to a bin is the number of trees within a certain range of depth (for example all trees of depth 6 to 10). Let $b(i)$ be the value of the i-th bin. As components of

[1]The fact that the central moment is not perfectly unbiased is not an issue for large n.

[2]The first central moment, $E([X - E(X)])$, is equal to 0.

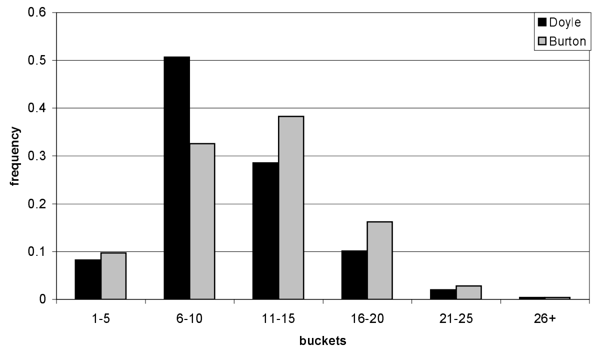

Figure 8.2: Tree Depth Histogram for two Authors

the feature vector for document d we consider these values normalized by the overall number n of trees in d and obtain the following vector:

$$\left(\frac{b(1)}{n}, \dots, \frac{b(m)}{n}\right) \tag{8.11}$$

We used 5 as concrete bin-size in our implementation. Figure 8.2 shows a comparison between the tree depth distributions for the authors A. C. Doyle and R. Burton (books from Gutenberg Project [3] - see Section 8.2.7) in the form of histograms.

8.2.6 Combining Features

In the previous sections we have described several alternative document representations, providing us with different kinds of information about content and style. In this section we describe two approaches to put these pieces of information together.

Combination Vectors

The idea of combination vectors is to merge the vectors obtained by different document representations into a single vector. This can be done by the concatenation of feature spaces. More precisely we are given k vector representations

$$\vec{v_1}(d), \dots, \vec{v_k}(d) \tag{8.12}$$

for document d with

$$\vec{v_i}(d) = (v_{i1}(d), \dots, v_{im_i}(d)) \tag{8.13}$$

where m_i is the size of the feature space for the i-th representation.

These vectors can be combined into a combination vector as follows:

$$\Big(\frac{v_{11}(d)}{n_1}, \ldots, \frac{v_{1m_1}(d)}{n_1}, \ldots\ldots, \frac{v_{k1}(d)}{n_k}, \ldots, \frac{v_{km_k}(d)}{n_k}\Big) \tag{8.14}$$

Here the values n_i are normalization constants. The rationale for this normalization is that strong variations between the order of magnitude of the components of the feature vectors might occur (this holds, e.g., for Bag-of-Words vs. moments of syntax tree depth distributions). We choose the normalization constants such that the average component value is the same for all subspaces corresponding to the original feature spaces. Formally, for a document set D, we choose the constants n_i such that the following requirement is satisfied:

$$\frac{1}{n_i}\frac{1}{m_i}\sum_{d\in D}\sum_{l=1}^{m_i} v_{il}(d) = \frac{1}{n_j}\frac{1}{m_j}\sum_{d\in D}\sum_{l=1}^{m_j} v_{jl}(d)$$
$$\text{for all } i,j \in \{1,\ldots,k\} \tag{8.15}$$

We can assign to one of the n_i an arbitrary value (say $n_1 = 1$); then the other normalization constants can be computed by elementary transformations of equations 8.15.

Meta Classification

With a given set $V = \{v_1, \ldots, v_k\}$ of k binary classifiers (obtained by using different documents representations) we can perform meta classification using a linear classifier combination with thresholding, as described in Section 4.2:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(v_i,d)\cdot w(v_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(v_i,d)\cdot w(v_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{8.16}$$

If a fixed set U of unlabeled documents (that does not change dynamically) is given, we can classify the documents with a user-acceptable loss of L as follows:

1. for all documents in U compute their classification confidence $\sum_{i=1}^{n} R(v_i,d)\cdot w(v_i)$
2. sort the documents into decreasing order according to their confidence values

Author	# Books	# Test Documents
Richard Burton	49	7425
Charles Dickens	55	7869
Arthur Conan Doyle	40	3473
Henry Rider Haggard	55	6882
George Alfred Henty	60	7169
Jack London	38	3566
Edgar Allan Poe	7	636
William Shakespeare	89	4025
Robert Louis Stevenson	45	6451
Mark Twain	129	9087

Table 8.4: Authors used from Gutenberg Corpus

3. classify the $(1-L)|U|$ documents with the highest confidence values according to their sign and dismiss the rest

In our experiments we assigned equal weights to each classifier, and instead of $R(v_i, d)$, we considered a "confidence" value $conf(v_i, d)$ for the classification of document d by the classifier. For SVM we considered the SVM scores, i.e., the distance of the test points from the hyperplane. A more enhanced method to map SVM outputs to probabilities is described, e.g., in [97].

8.2.7 Experiments

Setup

For the validation of the presented techniques, we considered a literature data set obtained from the Gutenberg Project [3], a volunteer effort to digitize, archive, and distribute cultural works. We selected 10 English and American authors with a sufficient number of books (listed in Table 8.4). For each author we divided each book into parts with 20 paragraphs and stored each part as a document in the database. From these documents, we randomly choose 600 per class for our experiments. We divided these documents for each author into a training set (100 documents) and an test set (500 documents).

For our experiments we considered binary classification on all 45 possible pairs of authors (e.g. "Burton" vs. "Dickens"). For every pair we chose $T \in \{20, 40, 60, 80, 100\}$

documents from the authors' training sets as positive and the same number of documents as negative samples. The classification was performed on the corresponding test sets.

Then, we computed the micro-averaged *error*, i.e. the ratio of incorrectly classified documents among all test documents. For restrictive meta classification we considered in addition the *loss*, the fraction of documents dismissed by the restrictive classifier. Additionally, we computed the 95 percent confidence interval for the error.

We compared the following methods for feature construction:

1. word based features
   a) Bag-of-Words using porter stemming and stopword elimination - see Section 8.2.1 (**BoW**)
   b) Function words - see Section 8.2.1 (**FW**)
   c) Part of Speech extraction of nouns and verbs; annotation with Connexor Machinese Phrase Tagger, using base forms of words constructed by Connexor - see Section 8.2.1 (**N&V**)
   d) n-grams within constituents; using the Stanford Lexparser, considering constituents of each sentence represented as PCFG-tree - see Section 8.2.2 (**Constit.**)
2. structure based features
   a) functional dependencies using Connexor Machinese Syntax for dependency tagging - see Section 8.2.3 (**FunctDep**)
   b) writing style using the Stanford Lexparser - see Section 8.2.4 (**Style**)
   c) histograms for syntax tree depth distribution - see Section 8.2.5 (**Hist.**)
3. combination vectors using Bag-of-Words, writing style, and tree depth histograms - see Section 8.2.6 - (**Combi**)

As classification method we chose standard linear SVM with parameter $C = 1000.0$. We used the popular *SVMlight* implementation [63].

Results

In our first experiment we compared the classification results of the different feature construction methods and their combination (see Table 8.5, Figure 8.3 for a chart

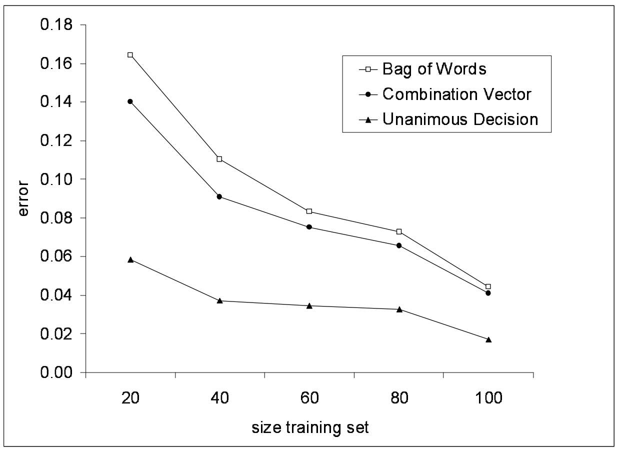

Figure 8.3: Comparison: Bag-of-Words and Combination Techniques on the Gutenberg Corpus

representation of Bag-of-Words - the best base classifier - vs. combination methods). As meta method we used a simple unanimous decision (**Unanimous Decision**) classifier with base classifiers based on Bag-of-Words, writing style, and tree depth histograms.

In a second experiment we took the confidence values for classification into account and induced different loss values for the meta classification as described in Section 8.2.6 (Table 8.6 and Figure 8.4).

The main observations are:

- The stylistic features work significantly better than random; nevertheless Bag-Of-Words provides us with better results. Obviously, in the Gutenberg corpus there is a high correlation between authors' vocabulary and generes of their novels (e.g., crime, horror, etc.).
- By combining Bag-Of-Words with the alternative features, we obtained significant improvements. For combination vectors we have especially improvements for a low number of training documents. With restrictive meta methods we accept a certain loss, but obtain a much lower error on the remaining documents.

	BoW	**FW**	**N&V**	**Style**	**FunctDep**	**Hist.**	**Constit.**	**Bigrams**
T	error	error	error	error	error	error	error	error
20	0.164 ±0.0034	0.354 ±0.0044	0.225 ±0.0039	0.186 ±0.0036	0.171 ±0.0035	0.299 ±0.0042	0.356 ±0.0044	0.458 ±0.0046
40	0.110 ±0.0029	0.240 ±0.0039	0.159 ±0.0034	0.138 ±0.0032	0.123 ±0.0030	0.278 ±0.0041	0.279 ±0.0041	0.390 ±0.0045
60	0.083 ±0.0026	0.177 ±0.0035	0.096 ±0.0027	0.123 ±0.0030	0.123 ±0.0030	0.273 ±0.0041	0.245 ±0.0040	0.323 ±0.0043
80	0.073 ±0.0024	0.147 ±0.0033	0.084 ±0.0026	0.114 ±0.0029	0.116 ±0.0030	0.275 ±0.0041	0.221 ±0.0038	0.285 ±0.0042
100	0.044 ±0.0019	0.115 ±0.0030	0.065 ±0.0023	0.103 ±0.0028	0.089 ±0.0026	0.272 ±0.0041	0.204 ±0.0037	0.230 ±0.0039

	Combination	**Meta Unanimous Decision**	
T	error	error	loss
20	0.142 ±0.0032	0.059 ±0.0016	0.482
40	0.092 ±0.0027	0.037 ±0.0014	0.399
60	0.074 ±0.0024	0.035 ±0.0013	0.362
80	0.065 ±0.0023	0.033 ±0.0013	0.349
100	0.040 ±0.0018	0.017 ±0.0010	0.345

Table 8.5: Error for Classification based on Different Features and their Combination on the Gutenberg Corpus

Loss	T = 20	T = 40	T = 60	T = 80	T = 100
0 %	0.165	0.111	0.083	0.073	0.044
10 %	0.146	0.092	0.069	0.058	0.033
20 %	0.130	0.079	0.058	0.048	0.027
30 %	0.114	0.068	0.048	0.039	0.021
40 %	0.099	0.059	0.040	0.032	0.017
50 %	0.086	0.049	0.033	0.025	0.014
60 %	0.075	0.043	0.028	0.021	0.011
70 %	0.064	0.038	0.023	0.016	0.008
80 %	0.053	0.034	0.019	0.013	0.008
90 %	0.039	0.030	0.016	0.010	0.009

Table 8.6: Error for Different User-Provided Loss Values using a Meta Classifier with BoW, Writing Style, and Functional Dependencies on the Gutenberg Corpus

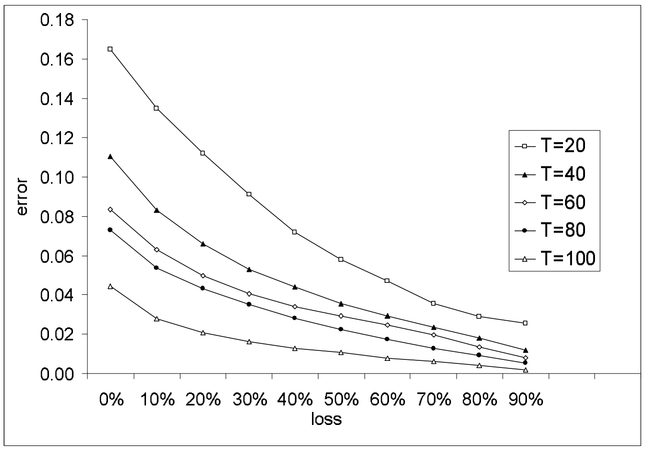

Figure 8.4: Comparison: Classification Results for Restrictive Meta Classification on the Gutenberg Corpus

9 Adaptive Retraining

This chapter addresses the problem of semi-supervised classification on document collections using retraining (also called self-training). A possible application is focused Web crawling which may start with very few, manually selected, training documents but can be enhanced by automatically adding initially unlabeled, automatically classified Web pages for re-training. Such an approach is by itself not robust and faces tuning problems regarding parameters like the number of selected documents, the number of retraining iterations, and the ratio of positive and negative classified samples used for retraining. We develop methods for automatically tuning these parameters, based on predicting the leave-one-out error for a re-trained classifier and avoiding that the classifier is diluted by selecting too many or weak documents for retraining.

9.1 A Simple Base Algorithm

Consider a training set T and a set of unlabeled data U. We can perform retraining by iteratively building a classifier C on T, classifying the documents in U and adding the documents with the highest classification confidence to the training set. Classification confidence could be estimated, e.g., by the distance from the separating hyperplane in the SVM case or by the probability of accepting a document for a class. This procedure can be described more precisely by the algorithm in Figure 9.1.

This algorithm provides us with a tradeoff. On one hand, a higher number of training examples could potentially improve the classification accuracy; on the other hand, there are potentially incorrectly labeled documents among the docs in U_{pos} and U_{neg}, which can dilute the training set. The algorithm has two important tuning parameters:

1. the number m of iterations
2. the ratio p/n between new positively classified and negatively classified docs used for retraining

```
Input: training set T = T_0
       set of unlabeled Data U = U_0

for i = 1, ..., m do
    build classifier C on T
    classify U
    U_pos := top-p positively classified docs
    U_neg := top-n negatively classified docs
    T = T + U_pos + U_neg
    U = U - U_pos - U_neg
```

Figure 9.1: Simple Retraining Algorithm

In the following we show how we can automatically tune these parameters. Note that the total number of selected documents for each retraining step, $r := p + n$ could be considered as an additional tuning parameter. However, we can simply choose it sufficiently small to be on the conservative side.

9.2 Tuning the Number of Iterations

Because of the tradeoffs mentioned above, a higher number of iterations do not necessarily imply a lower error. Our idea now is to approximate this error curve on the test set U by an estimated error curve.

For a retraining step we can build an error estimator by performing leave-one-out validation of the current classifier C on the original training set T_0, i.e., the part of the training set that consists of the manually labeled documents (which are assumed to be correctly labeled). This *partial* leave-one-out is described in Figure 9.2.

For a set of sample estimates

$$\{(i_0, estError(i_0)), \ldots, (i_l, estError(i_l))\}, \tag{9.1}$$

where the i_j values are the iteration numbers and $estError(i_j)$ is the estimated error, we can now approximate the overall error curve by fitting the sample estimates, as illustrated in Figure 9.3.

There are various approaches to this curve fitting. In our experiments we obtained good performance using cubic splines. Cubic splines are used in many areas, e.g., bio

```
Input: initial training set T = T_0
       L = set of automatically labeld data
           used for retraining
       C = classifier trained on T+L
Output: estimated error for C

classifiedDocs = 0
incorrectlyClassified = 0

for all docs d in T do
     build classifier C_L1o  on T+L-{d}
     if ( realClass(d) != classsify(d,C_L1o) )
          incorrectlyClassified++
     classifiedDocs++

return incorrectlyClassified/classifiedDocs
```

Figure 9.2: Partial Leave-One-Out

medicine, signal processing, and computer graphics [32, 94, 109]. In our experiments we also tested other approaches like linear splines, and error estimation by the less time consuming k-fold-cross-validation instead of leave-one-out.

Having approximated the error estimation curve $S(x)$, we choose the retraining classifier C in the iteration i with minimum $S(i)$ (see Figure 9.3). Choosing the number of supporting points for the fitting is an efficiency issue. The more supporting points the better the approximation but the higher the overall cost for computing the estimator values.

The classifier can be optimized in the same way for other quality measures like the F-measure (the harmonic mean of precision and recall).

9.3 Tuning the Ratio of Positive and Negative Samples

For an effective classification the training set should be an appropriate representation of the test set. For binary classification, it is especially helpful if the ratio between positive and negative documents is approximately the same for the test and

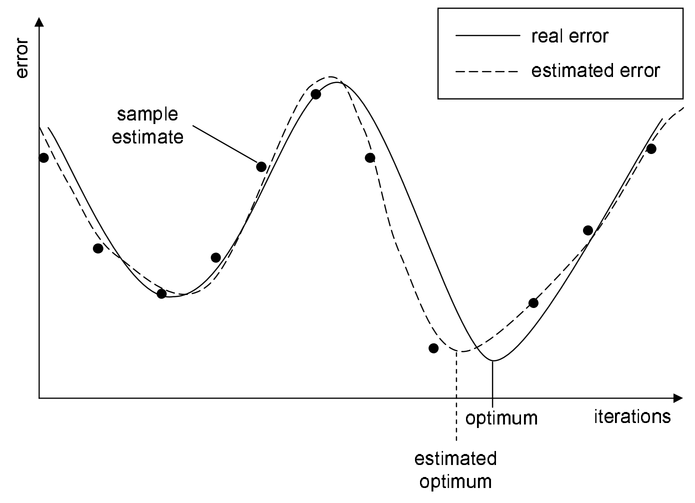

Figure 9.3: Optimal Number of Iterations: Approximation by Estimated Error Curve

the training set. For example, Bayesian classifiers take the prior class probabilities explicitly into account. For SVM a badly proportionate training set can also lead to a disadvantageous bias [25]. The assumption of having a training set with the same ratio of positive and negative documents as a test set is not at all self-guaranteed or easy to satisfy in practice. Typically a human, collecting training documents, would rather choose roughly the same number of documents for each class, even if there are significant (but a priori unknown) differences in the real world.

The idea is to overcome this problem by adjusting the training set such that it better represents the test set. To do so, in each iteration of our retraining algorithm we approximate the ratio between positive and negative documents by applying the current classifier to the set of initially unlabeled data U_0 (test data). Among a small number r of new retraining documents we choose the number of positive and negative documents, n and p, such that the difference between the overall ratio of positive and negative training docs and the estimated ratio on the unlabeled data is minimized.

More formally let t_{pos} be the number of positive, t_{neg} be the number of negative training documents in the current iteration, v_{pos} be the number of unlabeled documents classified as positive by the current classifier C, and v_{neg} be the number of documents classified as negative. Then we choose the number of *newly* added positive and negative documents for retraining, p and n, such that the ratio $(t_{pos} + p) : (t_{neg} + n)$ between the overall number of positive and negative training documents provides the best approximation for the ratio $v_{pos} : v_{neg}$ of positive and negative test documents estimated by the current classifier:

$$p = \arg\min_{x \in \{0,\ldots,r\}} \left| \frac{t_{pos} + x}{t_{neg} + r - x} - \frac{v_{pos}}{v_{neg}} \right| \tag{9.2}$$

and

$$n = r - p \tag{9.3}$$

As an illustrative example consider the case that the *initial* (manually labeled) training set consists of 100, and the set of initially unlabeled (test) data, that we aim to classify automatically, consists of 200 documents. Furthermore assume that, after i iterations, 100 of the test documents were automatically classified during previous iterations, and were, along with the initial training set, used to build the current classifier. Assume that the initial training set contains 50 positive and 50 negative examples and the 100 additional retraining documents after i iterations contain 60 examples classified as *pos* and 40 examples classified as *neg*. Thus after the i-th iteration, the ratio $\frac{t_{pos}}{t_{neg}}$ on the total number of examples used for training the current classifier is $\frac{50+60}{50+40} \approx 1.22$.
For the $i+1$-th iteration, we apply the current classifier (obtained in the $i-th$ iteration) to the the 200 initially unlabeled documents, resulting in, say, 120 positively and 80 negatively classified documents, i.e. $\frac{v_{pos}}{v_{neg}} = \frac{120}{80} = 1.5$. In the next step of the $i+1$-th iteration, we add $r = 10$ new retraining documents, such that the ratio *pos* : *neg* for the overall number of training documents used for the next classifier provides the best approximation for $\frac{v_{pos}}{v_{neg}}$. In our example this holds for $p = 10$ positive and $n = 0$ negative retraining documents. For this choice the fraction of positive and negative examples for the new classifier is $\frac{50+60+10}{50+40+0} \approx 1.33$, a better approximation than the ratio 1.22 after iteration i.

9.4 The Enhanced Retraining Algorithm

With the parameter tuning methods described above, our retraining algorithm now works as follows: We retrain as long as documents for retraining are available. In each retraining iteration we add a small number r of documents to the training set, determining the ratio between new positive and negative training documents as described in Section 9.3. Every *stepsize* iterations we compute and save an error estimator. We apply curve fitting to the estimated error, and choose the classifier corresponding to the minimum estimated error (see Section 9.2).

The pseudo code in Figure 9.4 summarizes our modified retraining algorithm.

```
Input: training set T = T_0
       set of unlabeled Data U = U_0
       stepsize

set of classifiers C-Set = empty
set of supporting points Support-Set = empty
iteration number i = 0;
while (U is not empty) do
   build classifier C on T
   add (i,C) to C-Set
   estimate p and n \\see Section 9.3
   classify U
   U_pos := top-p positively classified docs
   U_neg := top-n negatively classified docs
   T = T + U_pos + U_neg
   U = U - U_pos - U_neg
   if (i mod stepsize = 0)
      estimate error estError of C by leave-one-out on T_0
      add (i,estError) to Support-Set
   i++

compute interpolating curve S on Support-Set \\see Section 9.2
choose j which minimizes S(i)
return Classifier c from C-Set with iteration number = j
```

Figure 9.4: Enhanced Retraining Algorithm

9.5 Experiments

9.5.1 Setup

We performed a series of experiments with real-life data from the following sources.

1. The Newsgroups collection described in Section 4.5.

2. The Reuters articles described in Section 5.5.
3. The Internet Movie Database (IMDB) described in Section 4.5.

For every data collection we considered each class with at least 300 documents. We obtained 20 classes for Newsgroups, 8 for Reuters and 9 for IMDB. For each class we randomly chose 100 documents as positive training examples and 100 negative examples from all other classes. For validation we considered two cases:

1. the symmetric case: we chose equal numbers of positive and negative test documents for each class (200 per class)
2. the asymmetric case: we chose the number of positive and negative test documents in a ratio of 1 : 6 (i.e., 200:1200).

In all experiments, the standard bag-of-words model [17] (using term frequencies to build L1-normalized feature vectors, stemming with the algorithm of Porter [99], and deletion of stopwords) was used for document representation. We used binary classifiers so as to recognize documents from one specific topic against all other topics; this setup was repeated for every topic.

For each data collection we computed the macro-averaged error (i.e., the average ratio of incorrectly classified documents to the number of test documents) along with the 95 percent confidence interval and the macro-averaged F1 value (the harmonic mean of precision and recall).

9.5.2 Results

We compared the following classification methods:

1. Standard linear SVM (**SVM**)
2. Standard linear TSVM. Here the fraction f of unlabeled examples to be classified into the positive class is a selectable parameter. As default setting we used the ratio between the positive and the negative examples in the training data. (**TSVM**)
3. Linear TSVM where the ratio f between positive and negative test documents was set according to the SVM classification (Method 1) on the test documents. (**TSVM+est**)

Method	**Newsg.** avg(error)	**IMDB** avg(error)	**Reuters** avg(error)	**Newsg.** avg(F1)	**IMDB** avg(F1)	**Reuters** avg(F1)
SVM	0.106 ± 0.0067	0.262 ± 0.0144	0.06 ± 0.0082	0.891	0.734	0.941
TSVM	**0.1 ± 0.0066**	0.261 ± 0.0143	0.059 ± 0.0082	0.9	0.739	0.941
TSVM+est	0.105 ± 0.0067	0.262 ± 0.0144	**0.059 ± 0.0082**	**0.892**	0.734	0. **942**
EM-Bayes	0.129 ± 0.0073	**0.245 ± 0.014**	0.072 ± 0.009	0.884	0. **764**	0.93
SGT	0.119 ± 0.0071	0.277 ± 0.0146	0.088 ± 0.0098	0.882	0.727	0.914
RetCsplL1o	0.123 ± 0.0072	0.263 ± 0.0144	0.068 ± 0.0087	0.867	0.723	0.934
RetCsplCv	0.116 ± 0.007	0.257 ± 0.0143	0.062 ± 0.0083	0.877	0.729	0.939
RetLsplL1o	0.123 ± 0.0072	0.256 ± 0.0143	0.066 ± 0.0086	0.871	0.731	0.936
RetLsplCv	0.115 ± 0.007	0.258 ± 0.0143	0.062 ± 0.0083	0.879	0.731	0.939
RetCv	0.111 ± 0.0069	0.254 ± 0.0142	0.061 ± 0.0083	0.882	0.735	0.94

Figure 9.5: Macro-averaged Results for **Symmetric** Test Set: Baseline and Retraining Methods

Method	**Newsg.** avg(error)	**IMDB** avg(error)	**Reuters** avg(error)	**Newsg.** avg(F1)	**IMDB** avg(F1)	**Reuters** avg(F1)
SVM	0.097 ± 0.0035	0.246 ± 0.0075	0.075 ± 0.0049	0.726	0.481	0.783
TSVM	0.364 ± 0.0056	0.401 ± 0.0086	0.362 ± 0.0089	0.434	0.376	0.437
TSVM+est	0.096 ± 0.0035	0.249 ± 0.0075	0.076 ± 0.0049	0.728	0.475	0.78
EM-Bayes	0.202 ± 0.0047	0.267 ± 0.0077	0.093 ± 0.0054	0.596	0.498	0.75
SGT	0.216 ± 0.0048	0.329 ± 0.0082	0.167 ± 0.0069	0.543	0.402	0.606
RetCsplL1o	**0.077 ± 0.0031**	0.207 ± 0.0071	**0.058 ± 0.0043**	**0.749**	0.497	**0.818**
RetCsplCv	0.08 ± 0.0032	0.211 ± 0.0071	0.059 ± 0.0044	0.749	0.496	0.817
RetLsplL1o	0.081 ± 0.0032	0.212 ± 0.0071	0.058 ± 0.0043	0.744	0.49	0.813
RetLsplCv	0.083 ± 0.0032	0.209 ± 0.0071	0.06 ± 0.0044	0.744	0.491	0.812
RetCv	0.084 ± 0.0032	**0.204 ± 0.007**	0.059 ± 0.0044	0.745	**0.499**	0.816

Figure 9.6: Macro-averaged Results for **Asymmetric** Test Set: Baseline and Retraining Methods

4. The augmented EM-iterated Bayesian classifier with weighting of the unlabeled data as described in [91]. Here we determined the weighting parameter λ by leave-one-out validation (considering the values between 0 and 1 with a step width of 0.2), choosing the λ with the lowest estimated error. (**EM-Bayes**)

5. Spectral Graph Transduction as described in [67] (**SGT**)

6. Our retraining approach with linear SVM (Method 1) as the underlying base classifier and 10 new retraining documents per iteration and

   a) error/F1 prediction by leave-one-out estimation invoked after every 10 iterations and cubic spline interpolation (**RetCsplL1o**)

   b) error/F1 prediction by leave-one-out estimation invoked after every 10 iterations and linear spline interpolation (**RetLsplL1o**)

c) error/F1 prediction by 5-fold cross-validation invoked after every 10 iterations and cubic spline interpolation (**RetCsplCv**)

d) error/F1 prediction by 5-fold cross-validation invoked after every 10 iterations and linear spline interpolation (**RetLsplCv**)

e) error/F1 prediction by 5-fold cross-validation invoked after every iteration - and no interpolation (**RetCv**)

For SVM and TSVM we used the popular *SVMlight* implementation [63] with parameter C = 1000 (tradeoff between training error and margin). For the Spectral Graph Transductor we used the *SGTlight* implementation with parameterization as described in [67].

The average results for the symmetric and asymmetric test sets are shown in Figures 9.5 and 9.6. Detailed results for the different classes are shown in Figures 9.7 through 9.9. (The best values are highlighted in boldface.)

The main observations are:

- For the *symmetric* test case, there is no clear winner. For IMDB the EM-Bayes Method performs best, for Newsgroups and Reuters the TSVM algorithm is the best algorithm. An explanation for the good performance of standard TSVM is, that, for the symmetric case, the parameter f agrees completely with the real ratio between positive and negative documents in the test set.

- In the *asymmetric* test case, our retraining algorithm clearly provides the best performance on all three datasets. For example, on the IMDB data, which is the hardest test case in terms of the absolute accuracy that was achievable, we reduce the error from approximately 25-27 percent (for SVM and TSVM with estimator and for EM-iterated Bayes) to 20.7 percent, quite a significant gain. The very bad performance of standard TSVM can be explained by the big gap between the parameter f, estimated on the training set, and the real ratio between positive and negative documents in the asymmetric test set.

As we regard the asymmetric test case, significantly more unacceptable test documents than acceptable ones, as the far more realistic setting (e.g. in focused crawling, news filtering, etc.), we conclude that the newly proposed retraining method is the clear winner and outperforms the previously known state-of-the-art algorithms by a significant margin.

class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	error	error	error	error	error	error
alt.atheism	0.108	0.361	0.105	0.238	0.229	**0.071**
comp.graphics	0.116	0.36	0.113	0.258	0.186	**0.093**
comp.os.ms-windows.misc	0.085	0.357	0.084	0.171	0.115	**0.081**
comp.sys.ibm.pc.hardware	0.117	0.36	0.12	0.218	0.174	**0.113**
comp.sys.mac.hardware	0.106	0.363	0.109	0.222	0.285	**0.102**
comp.windows.x	0.084	0.36	0.089	0.298	0.191	**0.071**
misc.forsale	0.058	0.369	0.056	0.099	0.246	**0.051**
rec.autos	0.079	0.361	0.077	0.148	0.272	**0.066**
rec.motorcycles	0.066	0.364	0.064	0.324	0.248	**0.049**
rec.sport.baseball	0.07	0.363	0.071	0.089	0.237	**0.069**
rec.sport.hockey	0.048	0.361	0.048	0.084	0.129	**0.044**
sci.crypt	0.061	0.361	0.063	0.104	0.199	**0.056**
sci.electronics	0.199	0.363	0.201	0.201	0.307	**0.176**
sci.med	0.141	0.364	0.138	0.093	0.219	**0.064**
sci.space	0.074	0.37	**0.07**	0.354	0.182	0.074
soc.religion.christian	0.07	0.361	0.066	0.361	0.189	**0.066**
talk.politics.guns	0.096	0.369	0.094	0.162	0.251	**0.069**
talk.politics.mideast	0.066	0.364	0.066	0.128	0.167	**0.036**
talk.politics.misc	0.142	0.367	0.142	0.231	0.199	**0.092**
talk.religion.misc	0.153	0.371	0.153	0.257	0.288	**0.103**
class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	F1	F1	F1	F1	F1	F1
alt.atheism	0.708	0.438	0.716	0.544	0.524	**0.771**
comp.graphics	0.665	0.44	0.674	0.51	0.551	**0.679**
comp.os.ms-windows.misc	0.725	0.444	**0.73**	0.617	0.681	0.725
comp.sys.ibm.pc.hardware	**0.675**	0.44	0.667	0.559	0.561	0.659
comp.sys.mac.hardware	**0.7**	0.436	0.692	0.553	0.467	0.699
comp.windows.x	0.742	0.44	0.729	0.488	0.571	**0.754**
misc.forsale	0.808	0.427	0.812	0.72	0.493	**0.819**
rec.autos	0.757	0.438	0.761	0.65	0.498	**0.768**
rec.motorcycles	0.787	0.433	0.796	0.463	0.513	**0.826**
rec.sport.baseball	**0.776**	0.436	0.772	0.753	0.51	0.769
rec.sport.hockey	0.845	0.438	0.845	0.77	0.674	**0.855**
sci.crypt	0.811	0.438	0.806	0.728	0.548	**0.824**
sci.electronics	0.541	0.436	0.535	**0.557**	0.449	0.555
sci.med	0.644	0.433	0.651	**0.747**	0.549	0.684
sci.space	0.767	0.424	**0.78**	0.439	0.583	0.767
soc.religion.christian	0.787	0.438	**0.8**	0.438	0.576	0.791
talk.politics.guns	0.723	0.427	0.727	0.63	0.513	**0.768**
talk.politics.mideast	0.801	0.433	0.801	0.687	0.602	**0.866**
talk.politics.misc	0.641	0.429	0.641	0.548	0.547	**0.718**
talk.religion.misc	0.622	0.422	0.622	0.521	0.459	**0.684**

Figure 9.7: Asymmetric Test Set: Detailed Results for Newsgroups

class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	error	error	error	error	error	error
Action	0.267	0.403	0.283	0.281	0.456	**0.239**
Adventure	**0.309**	0.416	0.312	0.284	0.421	0.324
Comedy	0.349	0.434	0.355	**0.304**	0.396	0.32
Documentary	0.128	0.364	0.125	0.107	0.174	**0.094**
Drama	0.324	0.43	0.328	0.404	0.308	**0.259**
Horror	0.217	0.397	0.223	0.246	0.324	**0.199**
Sci-Fi	0.179	0.38	0.176	0.182	0.186	**0.114**
Thriller	0.308	0.41	0.311	0.492	0.482	**0.22**
Western	0.13	0.377	0.127	0.106	0.217	**0.094**
class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	F1	F1	F1	F1	F1	F1
Action	0.445	0.373	0.412	**0.475**	0.333	0.435
Adventure	0.393	0.353	0.387	**0.42**	0.304	0.383
Comedy	0.359	0.324	0.349	**0.402**	0.338	0.341
Documentary	0.664	0.433	0.672	0.7	0.561	**0.718**
Drama	0.357	0.331	0.349	**0.369**	0.35	0.364
Horror	0.486	0.382	0.473	**0.493**	0.371	0.484
Sci-Fi	0.573	0.409	0.58	0.576	0.525	**0.626**
Thriller	0.392	0.362	0.386	0.337	0.323	**0.41**
Western	0.657	0.413	0.664	0.712	0.516	**0.715**

Figure 9.8: Asymmetric Test Set: Detailed Results for IMDB

class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	error	error	error	error	error	error
acq	0.107	0.36	0.113	**0.083**	0.205	0.096
crude	0.031	0.361	0.031	0.094	0.149	**0.026**
earn	0.049	0.369	0.049	0.057	0.15	**0.031**
grain	0.083	0.363	0.086	**0.056**	0.187	0.064
interest	0.098	0.359	0.098	0.13	0.146	**0.08**
money-fx	0.079	0.363	0.081	0.115	0.191	**0.062**
trade	0.064	0.359	0.065	0.136	0.151	**0.04**
wheat	0.091	0.364	0.088	0.069	0.156	**0.063**
class	**SVM**	**TSVM**	**TSVM+est**	**EM-Bayes**	**SGT**	**RetCsplL1o**
	F1	F1	F1	F1	F1	F1
acq	0.71	0.44	0.695	**0.773**	0.572	0.722
crude	0.897	0.438	0.897	0.746	0.629	**0.912**
earn	0.836	0.427	0.836	0.804	0.607	**0.885**
grain	0.759	0.436	0.751	**0.803**	0.568	0.794
interest	0.738	0.442	0.738	0.684	0.655	**0.77**
money-fx	0.772	0.436	0.768	0.711	0.568	**0.798**
trade	0.811	0.442	0.807	0.675	0.624	**0.865**
wheat	0.741	0.433	0.749	**0.8**	0.628	0.797

Figure 9.9: Asymmetric Test Set: Detailed Results for Reuters

10 Conclusion

In this thesis we have started out to investigate the engineering and, in particular, tuning issues of using automatic classification and clustering algorithms for text document organization.

We have developed a constructive and practically efficient methodology for tuning a repertoire of classifiers and meta methods to the application's specific goals in terms of classification error and document loss. A key element in our approach has been to devise analytic estimators that can predict the error and loss for a given parameter setting sufficiently accurately. Although our techniques are anchored upon empirical leave-one-out or cross-validation estimators on the underlying training data to some extent, we have taken great care to avoid computationally expensive steps that would involve repeated retraining. While the approach applies to a wider repertoire of classifiers, we have especially worked on the k-split meta classifier based on SVM or Centroid, and we have experimentally shown that we can tune these methods to meet the application goals with a classification error that is competitive to that of conservative SVM variants at drastically reduced training cost.

We have shown, by a probabilistic model as well as by experiments on various data sets, that restrictive classification methods can be used to eliminate junk documents. Theory and experiments show that the junk reduction is significantly higher than the loss, and the classification error is decreased. This holds for restrictive base methods as well as meta methods.

We proposed an approach for automatically grouping heterogenous document collections by using restrictive clustering methods. A key element in our approach has been to construct restrictive meta methods that result in higher cluster purity. The introduced algorithms of meta mapping ensure better accuracy and make clustering results robust and accurate at the cost of moderate loss of uncertain samples. While the introduced approach applies to a wide range of partitioning methods, we have especially worked on k-means and its extensions. We have experimentally shown that meta clustering has higher accuracy than particular clustering methods and, more importantly, performs better than the restrictive version the underlying base methods with the same loss.

We used meta classification and clustering to construct distributed machine learning algorithms for P2P Web exploration applications. The results of the evaluation clearly show the advantages of cooperation between nodes for building meta decision models. Our method does not require the comprehensive exchange of private data collections between peers and thus provides substantial advantages for aspects of privacy, network bandwidth, storage, and computational expense. Furthermore, our restrictive meta methods clearly outperform the models that can be separately built on training sources of isolated peers and, more importantly, also the restrictive variant of such one-peer solutions with the same induced loss.

We described classification with different document representations. In addition to well known features like document terms in the Bag-Of-Words model, POS tagging, etc., we considered alternative stylistic features like the depth or the structure of syntax trees. We combined the feature representations using two techniques: 1) combination vectors, where we constructed a single vector from the different feature vectors with automatic normalization of the combination vector's components, 2) meta methods combining the classification results based on the different representations into a meta result. Our experiments on the author recognition task show that our new features are suitable for discriminating different styles and, used within combination techniques, lead to significant improvements of the classifier performance. We obtained similar results for the combination of document representations for Web document classification.

Our work on semi-supervised classification has been motivated by the fact that the availability of training data is often a critical point in real applications. This has led us to a semi-supervised learning approach with iterative retraining using initially unlabeled data. An additional difficulty that real applications often pose is the imbalance between acceptable and unacceptable documents in the corpus that creates a mismatch with the ratio of positive and negative training samples and may result in a wrong bias of the classifier. In Web applications, but also for large-scale intranet corpora, this is a typical situation and creates a major impediment to the previously proposed state-of-the-art techniques for semi-supervised classification. Our method successfully addresses these practically relevant issues, which were largely disregarded by prior work, and significantly outperforms the other methods in terms of classification accuracy.

Bibliography

[1] The 20 newsgroups data set. *http://www.ai.mit.edu/ jrennie/20Newsgroups/.*

[2] dmoz - open directory project. *http://dmoz.org/.*

[3] Gutenberg project. *http://www.gutenberg.org/.*

[4] Internet movie database. *http://www.imdb.com.*

[5] Lexparser. *http://www-nlp.stanford.edu/downloads/lex-parser.shtml.*

[6] National Institute of Standards and Technology, Gaithersburg (MD), USA. *http://www.nist.gov/.*

[7] The open-source biojava project. *http://www.biojava.org.*

[8] K. Aas and L. Eikvil. Text categorisation: A survey. *Technical report, Norwegian Computing Center*, 1999.

[9] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105, Seattle, Washington, United States, 1998. ACM Press.

[10] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994*, pages 487–499, Santiago de Chile, Chile, 1994. Morgan Kaufmann.

[11] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

[12] Massih-Reza Amini and Patrick Gallinari. The use of unlabeled data to improve supervised learning for text summarization. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 105–112, Tampere, Finland, 2002. ACM Press.

[13] I. Androutsopoulos, J. Koutsias, Chandrinos Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M.n van Someren, editors, *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000.

[14] Michael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 49–60, Philadelphia, Pennsylvania, United States, 1999. ACM Press.

[15] H. Baayen, H. van Halteren, A. Neijt, and F. Tweedie. An experiment in authorship attribution. In *Actes des 6èmes Journées Internationales d'Analyse des Données Textuelles (JADT)*, INRIA (Institut National de Recherche en Informatique et en Automatique), France, 2002.

[16] H. Baayen, H. van Halteren, and F. Tweedie. Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution. *Literary and Linguistic Computing*, 11(3):121–131, 1996.

[17] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

[18] K. R. Beesley. Language identifier: A computer program for automatic natural-language identification on on-line text. In *29th Annual Conference of the American Translators Association, Medford (NJ), USA*, 1988.

[19] Kristin P. Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 368–374. MIT Press, 1999.

[20] Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin. Exploiting unlabeled data in ensemble methods. In *KDD '02: Proceedings of the eighth ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 289–296, Edmonton, Alberta, Canada, 2002. ACM Press.

[21] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–229, Berkeley, California, United States, 1999. ACM Press.

[22] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Rev.*, 37(4):573–595, 1995.

[23] Henk Ernst Blok, Djoerd Hiemstra, Sunil Choenni, Franciska de Jong, Henk M. Blanken, and Peter M.G. Apers. Predicting the cost-quality trade-off for information retrieval queries: facilitating database design and query optimization. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 207–214, Atlanta, Georgia, USA, 2001. ACM Press.

[24] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, Madison, Wisconsin, United States, 1998. ACM Press.

[25] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Training text classifiers with SVM on very few positive examples. *Technical Report MSR-TR-2003-34, Microsoft Corp.*, 2003.

[26] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[27] Pável Calado, Marco Cristo, Edleno Silva de Moura, Nivio Ziviani, Berthier A. Ribeiro-Neto, and Marcos André Gonçalves. Combining link-based and content-based methods for web document classification. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 394–401, New Orleans, LA, USA, 2003. ACM Press.

[28] W. B. Cavner and J. M. Trenkle. Text categorization and information retrieval using wordnet senses. In *Third Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[29] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan-Kauffman, 2002.

[30] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 307–318, Seattle, Washington, United States, 1998. ACM Press.

[31] P. Chan. An extensible meta-learning approach for scalable and accurate inductive learning. *PhD thesis, Department of Computer Science, Columbia University, New York*, 1996.

[32] E.Q. Chen and C.F. Lam. Predictor-corrector with cubic spline method for spectrum estimation in compton scatter correction of spect. *Computers in biology and medicine, 1994, vol. 24, no. 3, pp. 229, Ingenta.*

[33] David A. Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000*, pages 430–436, Denver, CO, USA, 2000. MIT Press.

[34] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 509–516, Madison, Wisconsin, USA, 1998.

[35] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, Tampere, Finland, 2002. ACM Press.

[36] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Rec.*, 30(4):55–64, 2001.

[37] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, British Columbia, Canada, 1987. ACM Press.

[38] Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, pages 245–260, 2000.

[39] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123.

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

[41] Evgenia Dimitriadou, Andreas Weingessel, and Kurt Hornik. A combination scheme for fuzzy clustering. In *AFSS '02: Proceedings of the 2002 AFSS International Conference on Fuzzy Systems.*, pages 332–338, Calcutta, 2002. Springer-Verlag.

[42] Harris Drucker, Vladimir Vapnik, and Dongui Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[43] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley, 2000.

[44] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, Bethesda, Maryland, United States, 1998. ACM Press.

[45] M. Ester, H.-P. Kriegel, and J. Sander. *Knowledge Discovery in Databases.* Springer, 2001.

[46] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. AAAI Press, 1996.

[47] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[48] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML '04: Proceedings of the twenty-first*

international conference on Machine learning, page 36, Banff, Alberta, Canada, 2004. ACM Press.

[49] D. Fisher. Improving inference through conceptual clustering. In *Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence*, pages 461–465, Seattle, WA., USA, 1987. AAAI Press.

[50] Michelle Fisher and Richard M. Everson. When are links useful? experiments in text classification. In Fabrizio Sebastiani, editor, *Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003*, Lecture Notes in Computer Science, pages 41–56, Pisa, Italy, 2003. Springer.

[51] Ana L. N. Fred and Anil K. Jain. Data clustering using evidence accumulation. In *International Conference on Pattern Recognition ICPR (4)*, pages 276–280, Quebec, Canada, 2002. IEEE Computer Society.

[52] Yoav Freund. An adaptive version of the boost by majority algorithm. *Mach. Learn.*, 43(3):293–318, 2001.

[53] Johannes Fürnkranz. Exploiting structural information for text classification on the www. In David J. Hand, Joost N. Kok, and Michael R. Berthold, editors, *Advances in Intelligent Data Analysis, Third International Symposium, IDA-99*, volume 1642 of *Lecture Notes in Computer Science*, pages 487–498, Amsterdam, The Netherlands, 1999. Springer.

[54] J. Gennari, P. Langley, and D. Fisher. Models fro incremental concept formation. In *Artificial Intelligence*, pages 40:11–61, 1989.

[55] Eric J. Glover, Kostas Tsioutsiouliklis, Steve Lawrence, David M. Pennock, and Gary W. Flake. Using web structure for classifying and describing web pages. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 562–569, Honolulu, Hawaii, USA, 2002. ACM Press.

[56] José M. Gómez Hidalgo, Manuel Maña López, and Enrique Puertas Sanz. Combining text and heuristics for cost-sensitive spam filtering. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of the Fourth Computational Natural Language Learning Workshop, CoNLL-2000*, pages 99–102, Lisbon, Portugal, 2000.

[57] H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations*

Volume 6, Special issue on learning from imbalanced datasets, Pages: 30 - 39 (June 2004).

[58] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.

[59] M.M. Hasan and Y. Masumoto. Document clustering: before and after the singular value decomposition. Technical Report Information Processing Society of Japan, Natural Language Technical Reports, No.134, 1999.

[60] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65, New York City, New York, USA, 1998. AAAI Press.

[61] D.I. Holmes. The Evolution of Stylometry in Humanities Scholarship. *Literary and Linguistic Computing*, 13(9):111–117, 1998.

[62] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, Nashville, TN, USA, 1997. Morgan Kaufmann Publishers Inc.

[63] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In Claire Nedellec and Céline Rouveirol, editors, *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer, 1998.

[64] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 200–209. Morgan Kaufmann, 1999.

[65] Thorsten Joachims. Estimating the generalization performance of an svm efficiently. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 431–438, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[66] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *SIGIR '01: Proceedings of the 24th annual*

international ACM SIGIR conference on Research and development in information retrieval, pages 128–136, New Orleans, Louisiana, United States, 2001. ACM Press.

[67] Thorsten Joachims. Transductive learning via spectral graph partitioning. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 290–297. AAAI Press, 2003.

[68] Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 250–257, Williams College, Williamstown, MA, USA, 2001.

[69] Hillol Kargupta, Weiyun Huang, Krishnamoorthy Sivakumar, and Erik L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[70] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[71] D. Khmelev and F.J. Tweedie. Using Markov Chains for Identification of Writers. *Literary and Linguistic Computing*, 16(3):299–308, 2001.

[72] Ron Kohavi and George John. Automatic parameter selection by minimizing estimated error. In Armand Prieditis and Stuart Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 304–312, Tahoe City, California, USA, July 1995. Morgan Kaufmann.

[73] M. Koppel, S. Argamon, and A.R. Shimoni. Automatically Categorizing Written Texts by Author Gender. *Literary and Linguistic Computing*, 17(4):401–412, 2002.

[74] Balaji Krishnapuram, David Williams, Ya Xue, Alexander Hartemink, Lawrence Carin, and Mario Figueiredo. On semi-supervised classification. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17.* MIT Press, Cambridge, MA, 2005.

[75] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning, ICML'97, Nashville, TN, U.S.A., 179-186*, 1997.

[76] H.W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

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

[78] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 448–455. AAAI Press, 2003.

[79] D. Lewis. *Representation and learning in information retrieval.* PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[80] David D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318, Asilomar, CA, USA, February 1991. Defense Advanced Research Projects Agency, Morgan Kaufmann.

[81] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. Algorithms for clustering high dimensional and distributed data. *Intelligent Data Analysis*, 7(4):305–326, 2003.

[82] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[83] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkley Symp. Math. Statist, Prob.,*, pages 1:281–297, 1967.

[84] C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[85] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[86] Francesco Masulli and Giorgio Valentini. Comparing decomposition methods for classification. In *International Conference on Knowledge-Based Intelligent Engineering Systems and Applied Technologies, KES*, pages 788–792, Brighton, UK, 2000.

[87] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *International Conference on Data Mining (ICDM'03), Melbourne, FL*, 2003.

[88] T. Mitchell. *Machine Learning.* McGraw Hill, 1996.

[89] Alessandro Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In Sharon McDonald and John Tait, editors, *Advances in Information Retrieval, 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004, Proceedings*, volume 2997 of *Lecture Notes in Computer Science*, pages 181–196, 2004.

[90] N. Nanas, V. Uren, and A. de Roeck. Learning with positive and unlabeled examples using weighted logistic regression. In *15th International Workshop on Database and Expert Systems Applications(DEXA'04), Zaragoza, Spain*, 2004.

[91] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Intelligence*, 39(2/3), 2000.

[92] Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Anti-aliasing on the web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 30–39, New York, NY, USA, 2004. ACM Press.

[93] Michael Oakes. Ant colony optimisation for stylometry: The federalist papers. In *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, pages 86–91. Nottingham Trent University, 2004.

[94] E.I. Okanla and P.A. Gaydecki. A real-time audio frequency cubic spline interpolator. *Signal processing, 1996, vol. 49, no. 1, pp. 45, Ingenta.*

[95] Bo Pang and Lillian Lee. Thumbs up? sentiment classification using machine learning techniques. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, USA, 2002.

[96] W.P. Pierskalla. The multi-dimensional assignment problem. *Operations Research*, 16:422–431, 1968.

[97] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers, MIT Press*, 1999.

[98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, Melbourne, Australia, 1998. ACM Press.

[99] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[100] C. Van Rijsbergen. A theoretical basis for the use of co-occurence data in information retrieval. *Journal of Documentation, 33:2, pp. 106-119*, 1977.

[101] R. Rivest. The MD5 message digest algorithm. *RFC 1321*, 1992.

[102] Paolo Rosso, Edgardo Ferretti, Daniel Jiménez, and Vicente Vidal. Text Categorization and Information Retrieval Using WordNet Senses. In Petr Sojka, Karel Pala, Pavel Smrž, Christiane Fellbaum, and Piek Vossen, editors, *Proceedings of the Second International WordNet Conference—GWC 2004*, pages 299–304, Brno, Czech Republic. Masaryk University Brno, Czech Republic.

[103] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 1988, p. 513-523.*

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

[105] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 313–321, Nashville, Tennessee, USA, 1997. Morgan Kaufmann Publishers Inc.

[106] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, Tampere, Finland, 2002. ACM Press.

[107] Sam Scott and Stan Matwin. Text classification using WordNet hypernyms. In Sanda Harabagiu, editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[108] M. Seeger. Learning with labeled and unlabeled data. *Tech. Rep., Institute for Adaptive and Neural Computation, University of Edinburgh, UK*, 2001.

[109] Chris Seymour and Keith Unsworth. Interactive shape preserving interpolation by curvature continuous rational cubic splines. *Special issue: computational methods in computer graphics. J. Comput. Appl. Math. 102 (1999), no. 1, 87–117, Math. Sci. Net.*

[110] Stefan Siersdorfer, Andreas Kaster, and Gerhard Weikum. Combining text and linguistic document representations for authorship attribution. In *SIGIR-Workshop on Stylistic Analysis of Text for Information Access (STYLE)*, Salvador, Bahia, Brazil, 2005.

[111] Stefan Siersdorfer and Sergej Sizov. Konstruktion von Featureräumen und Metaverfahren zur Klassifikation von Webdokumenten. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *Datenbanksysteme für Business, Technologie und Web (BTW) : 10. GI-Fachtagung*, volume P-26 of *Lecture Notes in Informatics*, Leipzig, Germany, February 2003. German Informatics society (GI), Bonner Köllen.

[112] Stefan Siersdorfer and Sergej Sizov. Restrictive clustering and metaclustering for self-organizing document collections. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 226–233, Sheffield, United Kingdom, 2004. ACM Press.

[113] Stefan Siersdorfer, Sergej Sizov, and Gerhard Weikum. Goal-oriented methods and meta methods for document classification and their parameter tuning. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 59–68, Washington, D.C., USA, 2004. ACM Press.

[114] Stefan Siersdorfer and Gerhard Weikum. Automated retraining methods for document classification and their parameter tuning. In *The 6th International Conference on Web Information Systems Engineering (WISE)*, New York City, USA, 2005.

[115] Stefan Siersdorfer and Gerhard Weikum. Using restrictive classification and meta classification for junk elimination. In David Losada and Juan M. Fernández Luna, editors, *Proceedings of the 27th European Conference on Information Retrieval (ECIR '05)*, volume 3408 of *Lecture Notes in Computer*

Science, pages 287–299, Santiago de Compostela, Spain, 2005. Information Retrieval Specialist Group of the British Computer Society (BCS-IRSG), Springer.

[116] Sergej Sizov, Martin Theobald, Stefan Siersdorfer, Gerhard Weikum, Jens Graupmann, Michael Biwer, and Patrick Zimmer. The BINGO! system for information portal generation and expert Web search. In *CIDR 2003. Proceedings of the 2003 Conference on Innovative Data Systems Research, January 5-8, 2003*, pages 69–80, Asilomar, USA, 2003. VLDB.

[117] Ellen Spertus. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pages 1058–1065, Providence, Rhode Island, USA, 1997. AAAI Press / The MIT Press.

[118] A. Strehl and J. Gosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research 3, pp. 583-617*, 2002.

[119] C. M. Tan, Y. F. Wang, and C. D. Lee. The use of bigrams to enhance text categorization. *Information Processing and Management., vol. 30, No. 4, pp. 529-546, 2002.*

[120] Pasi Tapanainen and Timo Järvinen. A non-projective dependency parser. In *Proceedings of the fifth conference on Applied natural language processing*, pages 64–71, Washington, DC, USA, 1997. Morgan Kaufmann Publishers Inc.

[121] Alexander Topchy, Anil K. Jain, and William Punch. Combining multiple weak clusterings. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 331, Melbourne, Florida, USA, 2003. IEEE Computer Society.

[122] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417–424, Philadelphia, PA, USA, 2002.

[123] Jaideep Vaidya and Chris Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM'04)*, Lake Buena Vista, Florida, USA.

[124] H. van Halteren. Writing Style Recognition and Sentence Extraction. In *Workshop on Text Summarization, DUC*, Philadelphia, Pennsylvania, USA, 2002.

[125] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[126] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, Washington, D.C., 2003.

[127] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[128] Janyce Wiebe, Eric Breck, Chris Buckley, Claire Cardie, Paul Davis, Bruce Fraser, Diane J. Litman, David R. Pierce, Ellen Riloff, Theresa Wilson, David Day, and Mark T. Maybury. Recognizing and organizing opinions expressed in the world press. In Mark T. Maybury, editor, *New Directions in Question Answering*, pages 12–19. AAAI Press, 2003.

[129] D.H. Wolpert. Stacked generalization. *Neural Networks, Vol. 5, pp. 241-259*, 1992.

[130] Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1-2):69–90, 1999.

[131] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[132] Hwanjo Yu, Kevin Chen-Chuan Chang, and Jiawei Han. Heterogeneous learner for web page classification. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 538, Washington, DC, USA, 2002. IEEE Computer Society.

[133] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA, 2004.

[134] Zhi-Hua Zhou, Ke-Jia Chen, and Yuan Jiang. Exploiting unlabeled data in content-based image retrieval. In *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings*, volume 3201 of *Lecture Notes in Computer Science*, pages 525–536. Springer, 2004.