An Evaluation of Bag-of-Concepts Representations in Automatic Text Classification

OSCAR TÄCKSTRÖM

KTH Numerical Analysis
and Computer Science

Master's Degree Project
Stockholm, Sweden 2005

TRITA-NA-E05150

Numerisk analys och datalogi
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

An Evaluation of Bag-of-Concepts Representations in Automatic Text Classification

OSCAR TÄCKSTRÖM

TRITA-NA-E05150

Master's Thesis in Computer Science (20 credits)
at the School of Media Technology,
Royal Institute of Technology year 2005
Supervisor at Nada was Magnus Rosell
Examiner was Stefan Arnborg

Abstract

Automatic text classification is the process of automatically classifying text documents into pre-defined document classes. Traditionally, documents are represented in the so called *bag-of-words* model. In this model documents are simply represented as vectors, in which dimensions correspond to words. In this project a representation called *bag-of-concepts* has been evaluated. This representation is based on models for representing the meanings of words in a vector space. Documents are then represented as linear combinations of the words' meaning vectors. The resulting vectors are high-dimensional and very dense. The author have investigated two different methods for reducing the dimensionality of the document vectors: feature selection based on gain ratio and random mapping.

Two domains of text have been used: abstracts of medical articles in english and texts from Internet newsgroups. The former has been of primary interest, while the latter has been used for comparison. The classification has been performed by use of three different machine learning methods: Support Vector Machine, AdaBoost and Decision Stump. Results of the evaluation are difficult to interpret, but suggest that the new representation give significantly better results on document classes for which the classical method fails. The representations seem to give equal results on document classes for which the classical method works fine. Both dimensionality reduction methods are robust. Random mapping shows greater variance, but is much less computationally expensive.

En utvärdering av *bag-of-concepts* för representation av dokument vid automatisk textklassificering

Sammanfattning

Automatisk textklassificering innebär att med datorns hjälp klassificera textdokument från en textmängd i fördefinierade dokumentklasser. Traditionellt representeras dokument i den så kallade *bag-of-words*-modellen. I denna modell representeras ett dokument av en vektor, i vilken ord utgör dimensioner. I detta examensarbete har en representation kallad *bag-of-concepts* undersökts. Denna bygger på modeller för att representera ords betydelser i en vektorrumsmodell. Ett dokument representeras därefter som en linjärkombination av ords meningsvektorer. De resulterande vektorerna är högdimensionella och väldigt täta. Författaren har undersökt två metoder för att reducera dimensionaliteten hos dokumentvektorerna: särdragsselektion baserat på informationsvinstskvot samt slumpavbildning.

Två textdomäner har använts: engelskspråkiga sammanfattningar till medicinska artiklar samt texter från nyhetsgrupper på Internet. Fokus har lagts på den förra, medan den senare har använts i jämförande syfte. Själva klassificeringen har utförts med hjälp av tre olika metoder från maskininlärningsområdet: supportvektormaskin, boosting och beslutsträdsstump. Resultatet av utvärderingen är svårtolkat, men pekar på att den nya representationen ger signifikant bättre resultat på dokumentklasser för vilka den klassiska metoden inte alls fungerar, medan representationerna verkar ge likvärdiga resultat på dokumentklasser för vilka den klassiska fungerar bra. Båda dimensionsreduceringmetoderna visar sig vara robusta. Random mapping ger större varians, men kräver samtidigt väsentligt mindre beräkningsresurser.

Thank You!

Magnus Rosell at KTH Computer Science and Communication for supervision, inspiration and encouragement.

Martin Hassel at KTH Computer Science and Communication for the implementation of Random Indexing and the Porter Stemmer.

Catharina Rehn at Karolinska Institutet for help with MeSH and MedLine.

Magnus Sahlgren at SICS for some inspiring discussions.

Other participants in the Infomat project.

Henrik Lindström for proof reading and interesting discussions.

Henrik, Peter, Carolin, Fredrik and Erik for making the time at KTH a great time.

Caroline for all your love and support.

Contents

List of Symbols and Abbreviations

Symbol	Description
$\mathcal{C}$	Set of document classes
$\mathcal{D}$	Document domain
Ω	Set of pre-classified documents
$\check{\Phi}$	Classification function
Φ	Classifier
T	True
F	False
$\mathcal{F}$	Set of features
ξ	Dimensionality reduction factor
k	Dimensionality of random vectors (and number of folds in cross validation)
e	Number of non-zero entries per random vector
$\mathcal{X}$	Feature space
$\mathcal{H}$	Hypothesis space
$\mathcal{L}$	Learner
$\Re$	Set of real numbers
δ	Margin of separating hyperplane
L	Loss function
R_{emp}	Empirical risk
R	Expected risk (and random matrix in random mapping)
κ	Kernel function
C	Slack tolerance
σ	Variance of RBF-kernel
Tr	Training set
Va	Validation set
Te	Testing set
π	Precision
ρ	Recall

Abbreviation	Complete Term
TC	Text Classification
IR	Information Retrieval
ML	Machine Learning
BoW	Bag-of-Words
BoC	Bag-of-Concepts
DOR	Document Occurrence Representation
TCOR	Term Co-occurrence Representation
LSI	Latent Semantic Indexing
FS	Feature Selection
IG	Information Gain
GR	Gain Ratio
RM	Random Mapping
RI	Random Indexing
SVM	Support Vector Machine
AdaBoost	Adaptive Boosting
DS	Decision Stump
CV	Cross Validation

Chapter 1

Introduction

The amount of textual data in the world increase every day. Be it web documents on the World Wide Web or documents in the medical domain, if the texts are to be useful to the users some form of structure is needed. On the Internet one often rely on the users to find what they need by using search engines. In the medical domain, which is characterized by disparate language use and an extraordinary amount of highly specialized sub-domains, good search engines are more difficult to implement. Instead one has devoted much time and resources to develop hierarchical structures of the domain and its sub-domains. One such structure is the *Medical Subjects Headings*, MeSH, briefly described in section 6.2.

In this thesis we study the problem of *document representation* in the context of automatic text classification. *Text classification*, TC, described in the next sections, is the process of assigning free text documents to predefined classes. Traditionally documents are represented in the *bag-of-words*, BoW, model, in which the *semantics* or *content* of a document is viewed simply as the set of words it contains. We investigate a novel representational model called *bag-of-concepts*, BoC, which instead relies on the intuition that the semantics of a document may be viewed as the union of the *meanings* of the words it contains. This model was proposed by Sahlgren and Cöster in [26].

With the increasing performance of search engines and other *information retrieval*, IR, tools, one might question the need for text classification. Why should we impose a static structure on such a dynamic thing as a document domain? This is not at all a simple issue, but we believe that information retrieval and search engines are good when the user knows what she wants and knows which words to use to find it. Text classification can aid in this too, but it is really good when the user does not know exactly what she wants or knows what she wants but not how to find it. This is especially the case when the classification scheme is in the form of a hierarchy of classes, since it permits *browsing* the document domain in a structured fashion. We also believe that text classification can be an effective tool for imposing a standardized structure on such a disparate domain as the medical domain, which in turn can aid in the adoption of a standardized conceptual framework and a standardized language.

Text classification can be used in many applications. One example is *document organization*, i.e. the organization of documents into different categories. Another is *text filtering*, i.e. the on-line classification of a stream of incoming documents, e.g. e-mail, into *relevant* and *irrelevant* documents. *Authorship attribution* and *genre classification*, i.e. the classification of a document into its genre or its author, is yet another application [32]. In this thesis we are interested in document organization. Hopefully the results gained here are applicable in other applications as well.

Much research have been devoted to the field of automatic text classification and there is an extensive amount of literature. Two excellent surveys which have formed the starting point for our work are [31] and [32]. The following two sections are to a large extent based on [31].

1.1 Text Classification

To automate the process of text classification, a process traditionally carried out by humans, we must somehow mimic the knowledge used by human classifiers. Throughout the years many different methods for carrying out this task have been proposed. Of these, the most successful ones have originally been developed in the *machine learning*, ML, community. Document classes are mostly defined by use of ostensive definitions. We thus have a problem of inducing an internal representation of document classes based only on pre-classified documents. This is exactly the general problem studied in the field of machine learning.

The machine learning approach can be contrasted with the *knowledge engineering* approach which was prevailing in the first half of the 1980's. Here human experts built deductive systems, often in some form of *if <DNF-formula>, then <Category>*[1] rules [31]. In this approach the knowledge of human classifiers is directly used in the construction of the classifier, while in the machine learning approach this knowledge has to be used indirectly.

The problem of text classification can be contrasted with the related problem of *clustering* in which one wants to find a partitioning of a set of documents into clusters, such that documents in the same cluster are more similar than documents in different clusters. An important and interesting problem in itself, clustering has been used as a pre-processing step in some TC systems. This thesis deals exclusively with the problem of text classification, however the insights gained on the issue of document representation may be useful in areas such as clustering as well.

Before continuing with the definition of TC, we make a note on language use: the terms *text categorization* and *text classification* are both used in the literature, with a small preference towards the former. In this thesis we exclusively use the term text classification. Our rationale for this is the extensive use of the term *classifier* in the machine learning community.

1.2 Definition

Following [31], *hard* TC is defined as a problem of approximating the function $\check{\Phi} : \mathcal{D} \times \mathcal{C} \mapsto \{T, F\}$, where $\mathcal{D}$ is the document domain, $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ is a finite set of pre-defined classes and T, F denotes *True* and *False*. With $\Omega \subset \mathcal{D}$ we denote the set of pre-classified documents. $\check{\Phi}(d_j, c_i) = T$ if document d_j belongs to class c_i otherwise $\check{\Phi}(d_j, c_i) = F$. In the case of *soft TC* one instead wants to approximate the function $CSV : \mathcal{D} \times \mathcal{C} \mapsto [0, 1]$, with $CSV(d_j, c_i)$ expressing the certainty with which document d_j belongs to class c_i. CSV here stands for *classification status value*. The approximation of $\check{\Phi}$ is denoted Φ. This function forms a *classifier* which can be used to classify documents from $\mathcal{D}$.

With soft TC it is possible to rank documents with respect to approprietness to classes and vice versa giving what [39] call *category ranking TC* and *document ranking TC*. This can be useful to humans for information searching and retrieval purposes. It can also be used to support human indexers in their work since they can pay more attention to classes or documents with high expected appropriateness [32]. Soft TC can easily be transformed

[1] DNF stands for *disjunctive normal form*

into hard TC by assigning a threshold τ_i to each class c_i. This is sometimes necessary since some function approximation methods work with real-valued output only. In this thesis we consider only the problem of hard TC.

The problem of text classification can be further divided into *single-label TC* and *multi-label TC*. [31] The former is a special case of what we could call *k-label TC* with $k = 1$. This means that exactly one category is assigned to every document d_j. Multi-label TC on the other hand means that any number, between 0 and $|\mathcal{C}|$, of categories can be assigned to each document d_j. *Binary TC* is a special case of single-label TC, in which each document d_j is assigned either to class c_i or to its complement $\bar{c}_i$.

According to [31] binary TC is the most theoretically interesting case since multi-label TC can be achieved by combining $|\mathcal{C}|$ binary classification problems under $\{c_i, \bar{c}_i\}$ to solve the problem of classification under $\{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$. With no knowledge of the interdependence among classes, the instances of binary TC are usually assumed to be statistically independent. However, this is not a settled issue and there are several methods that make use of inter-class dependence, e.g. regression based models. When topological information is available about the relationships between classes, such as with a hierarchical class structure, a more or less strong assumption of dependence between the binary instances can also be fruitful.

This general setting of the problem does not mention anything about the representation of documents, the intrinsic properties of the functions $\check{\Phi}$ and Φ, how Φ is built or how the "goodness" of Φ is measured. This is beneficial since it allows us to study the problem in a modular fashion.

In [32] the construction of a TC system is divided into three separate phases: *document indexing*, *classifier learning* and *classifier evaluation*. This division is motivated by the fact that historically these phases have been studied in isolation to each other. However, it is not the case that these phases are independent of each other. The performance of most machine learning methods, for example, depend quite strongly on the representation of documents, which is built during the document indexing phase.

Bearing this in mind, we make the same division but instead of using the term *document indexing*, we use the term *document representation*. This use is not optimal since the term *representation* does not indicate a phase in a process in the same way as the other two. However the term *document indexing* is potentially confusing due to its strong connotation with the practice performed by human indexers.

1.3 Hypothesis, Motivation and Delimitation

The main hypothesis underlying this thesis is that more elaborate bag-of-concepts representations of documents may be beneficial in automatic text classification. Although much research has been directed at the problem of document representation over the years, the majority of the research has been concerned with improving the classification algorithms. When representational issues have been studied the interest have mainly been directed towards details of the prevailing bag-of-words model.

The idea is that since text classification is stated as a task concerning semantics or content, it should be possible to improve results by using representations which better capture the semantics of documents. A second hypothesis is thus that bag-of-concepts representations actually do capture semantics better than the standard bag-of-words representation. This hypothesis is not particularly well developed. For example we have not defined what *content* or *semantics* really mean. However interesting these questions are from a more philosophical or cognitive viewpoint, we are forced to take a strictly instrumental stance,

i.e. we are only interested in whether the representations lead to improved results or not.

We are primarily interested in the relative merits of the different representations and we have not designed our experimental setups etc. in order to be comparable with experiments in the literature. This would of course be preferable, but since there are so many parameters involved, such comparisons would be very hard to conduct. Therefore the results of this thesis should not be seen as automatically valid in other document domains, with other algorithms and parameter choices.

Due to time constraints we have not been able to fine-tune parameter settings or other implementational details. If this had been our first priority, we would have been forced to keep the evaluation much more narrow in scope. Of course such an approach has its merits, but we opt for a broader view of the area.

Unfortunately, we have only been able to work with documents in English. Hopefully some of the insights gained in this thesis will be applicable for other languages as well.

1.4 Thesis Outline

The rest of this thesis is organized as follows: In chapter 2 different ways of representing documents and weighting the dimensions of these representations are described together with some common preprocessing steps. Chapter 3 contains a survey of different dimensionality reduction methods for use with the created document representations. In chapter 4 we move on to describe the different machine learning techniques employed. In chapter 5 the problem of evaluating the resulting classifier is discussed. The specific methods, data sets and parameter settings used is described in chapter 6. Chapter 7 contains the results and a discussion. Finally, the main conclusions are summarized in chapter 8, which also contains some suggestions for further research.

Chapter 2

Representation

In this chapter we give an account of different models for representing documents in text classification and information retrieval tasks. We start with a brief description of the general vector space model used by all representations. We then discuss different ways of defining the feature space. More specifically we describe the classical bag-of-words model which is contrasted with the novel bag-of-concepts model. Following this is a description of the weighting scheme used for weighting the dimensions of the feature space. Finally we discuss some commonly used preprocessing techniques.

2.1 The Vector Space Model

The representation of documents significantly influences the performance of the resulting classification system. All representations of interest to us in this thesis are variations of a general vector space model in which each document $d_j \in \mathcal{D}$ is represented as a vector $\bar{d_j} = (x_{1j}, x_{2j}, \ldots, x_{|\mathcal{F}|j})$ of *feature weights*, where $\mathcal{F}$ is the set of *features*. Most often words are used as features. A document is then represented as a vector of weighted word occurrence counts. The assumption underlying all instances of this representational scheme is that the value of x_{ij} can be interpreted as a measure of the contribution of feature f_i to the semantics of document d_j [31]. In almost all representations the feature weights are calculated by passing the frequencies of the features in the documents to a weighting function, see section 2.3.

The documents in $\mathcal{D}$ are collected in a $|\mathcal{F}| \times |\mathcal{D}|$ *features-by-document* matrix. Despite the fact that some features in $\mathcal{F}$ probably are more important than others and most linear combinations of feature or document vectors does not have any meaningful interpretation, the documents are often seen as points in a $|\mathcal{F}|$-dimensional space. By imposing some distance measure, e.g. *Euclidean distance* or angles between vectors, we can explore this space geometrically, e.g. compute the degree of similarity between documents [2].

2.2 Defining the Feature Space

Theoretically, there are infinitely many ways of defining $\mathcal{F}$. In practice, however, we generally want the features to correspond to as meaningful, general and unambiguous entities as possible. Here *meaningfulness* means that features should have significantly high degrees of semantic content, *generality* means that we prefer having one feature capture small variations in semantic content over having many features capture the "same" semantic content and *unambiguousness* means that we prefer having each feature capture one semantic

concept over many. Note that these constraints are not thoroughly defined and serve merely as an intuitive base when we discuss different representations.

Related to these intuitions are the linguistic notions of *synonymy* and *polysemy*. Synonymy denotes "the semantic relation that holds between two words that can (in a given context) express the same meaning".[1] This is exemplified by the words *car* and *motor* which in some contexts have the same meaning. Synonymy is not necessarily an equivalence relation, i.e. two synonymous words need not express the exact same meaning and they probably never do. The prevalence of synonymy in natural language is shown in experiments conducted by Deerwester et al. [5], in which two people choose different key words to describe a well-known object over 80% of the time.

Polysemy, on the other hand, denotes "the ambiguity of an individual word or phrase that can be used (in different contexts) to express two or more different meanings". An example of a polysemous word is *bank* which can refer to a financial institution as well as the land slope near water. Synonymy and polysemy are problematic when we are creating representations for use with IR or TC systems. In both domains synonymy often lead to a decrease in *recall*, while polysemy often lead to a decrease in *precision*, see section 5.

2.2.1 Bag-of-Words and Related Surface Forms

As mentioned above, features are mostly identified directly with surface forms[2], such as words or word-stems, giving what is known as the *bag-of-words*, BoW, representation. Sometimes larger units of text are used. This include n-grams, i.e. sequences of n words, often with $n = 2$ or $n = 3$, and phrases (most commonly noun phrases). The rationale underlying this is the fact that TC is cast as a problem concerning *semantics* and larger units of text are assumed to contain more semantic information. These larger units, however, have not proven significantly better for TC than words or word-stems [31]. The reason for this is unclear: either the increase in semantic information is not useful for TC or the sampling noise introduced when using these larger language units is prohibitive.

In addition to these feature definitions which Joachims [10] call the *word level* and *multi-word level*, he additionally mentions the *sub-word level* at which features are identified with for example bigrams or trigrams of characters. This can be useful for languages which contain many composite forms. In the following we refer to the features defined on these different levels as *terms*.

We note that the word level constitutes a middle way in terms of the constraints mentioned above. Features at the sub-word level are general at the cost of ambiguity and a low degree of semantic content. Features at the multi-word level on the other hand are more specific and have a higher degree of semantic content at the cost of generality.

Some ML techniques depend explicitly on these constraints, e.g. the Naive Bayes classifier which is based on the assumption that features are statistically independent, or in other words that all features are optimally general. Ambiguousness on the other hand introduce noise. Highly specific features such as phrases are problematic primarily since we need a vast amount of data to get statistically reliable estimates of feature weights.

The assumption that the semantic content of a document can be modeled simply as the bag of words it contains have shown to be justified in many applications. It is for example the model underlying practically all search engines. However, since the model does not take

[1] These definitions of synonymy and polysemy are cited from WordNet, an online lexical reference system containing information on amongst other things synonymy relations and polysemy. WordNet can be found at *http://wordnet.princeton.edu/*.

[2] By *surface form* we mean language units accessible without performing any deep parsing or other more advanced processing.

any semantic relationships between words into consideration, it has problems especially with synonymous and polysemous words.

The problem of synonymy can be somewhat alleviated by using some preprocessing to map words with similar morphological roots to the same term, see section 2.4. Such preprocessing may map words with very similar meanng, e.g. *bicycle* and *bicycles*, to the same term. But the simple BoW approach can never fully utilize the fact that e.g. the words *car* and *motor* may denote the same entity. The model may then perform well in text classification insofar as all the words related to a certain document class appear with sufficiently high frequency in training examples belonging to that class. The problem of polysemy can be reduced by using some form of *PoS*-tagging[3] to separate e.g. the noun form and verb form of the same word.

2.2.2 Representing Term Meaning

The semantics or *meanings* of terms can also be represented in a vector space model. Following Sahlgren and Cöster [26], we use the term *concept vector* for the resulting vectors. Next we survey two different schemes for creating the concept vectors. This overview is based on [16].

There are two different ways of representing term meanings in a vector space: the *document occurrence representation*, DOR, and the *term co-occurrence representation*, TCOR. Both representations are based on the *distributional hypothesis* which states that words that frequently occur in the same contexts are semantically similar [13].

In the DOR term t_j is represented as a vector $\bar{t}_j = (c_{1j}, c_{2j}, \ldots, c_{|\mathcal{C}|j})$ of *context weights*, where c_{ij} is the weighted frequency of term t_j occurring in context c_i. In this representation the meaning of a term is viewed simply as the bag-of-contexts in which it occurs. This representation is thus the dual of the bag-of-words representation, if we identify contexts with documents.

In the TCOR, on the other hand, term t_j is represented as a vector $\bar{t}_j = (t_{1j}, t_{2j}, \ldots, t_{|\mathcal{T}|j})$ of *term weights*. The meaning of a term is viewed as the bag-of-terms with which it co-occurs in some context.

What should constitute a context in these representations is an open question. In the *Hyperspace Analogue to Language*, HAL, model which is a TCOR-based model, a fixed sized sliding window is passed over the document, ignoring sentence boundaries, and term co-occurrences within this sliding window are counted [19]. The *Latent Semantic Indexing*, LSI, model, a DOR-based model, on the other hand uses whole documents as contexts [5]. Other possible context definitions include sentences, subsections and even document domains. Note that the TCOR model always yield a square co-occurrence matrix of size $|\mathcal{T}| \times |\mathcal{T}|$ while the DOR model yield an occurrence matrix of size $|\mathcal{T}| \times |\mathcal{C}|$.

Sahlgren [25] argue that a narrow context, i.e. a sentence or a narrow sliding window, captures what a word *means* as opposed to a larger context, i.e. a document or a large sliding window, which captures what a word *is about*. In this they do not make any difference between the DOR and the TCOR.

However as Lavelli et al. [16] point out, there is a fundamental difference between the DOR and the TCOR and they show that the TCOR significantly outperforms the DOR in some term classification tasks, when using documents as contexts. This is specifically the case for words that are perfect synonyms. While some authors use two terms with "identical" meaning interchangeably, others tend to use only one of the terms throughout a document. In this case the DOR model will not capture the fact that these words are

[3] Part-of-speech tagging.

semantically similar, while the TCOR model will, since the two synonymous terms will probably co-occur with similar terms [16]. When short documents such as abstracts of articles are used, the problem will probably be aggravated since there is no room to use both terms in the same document.

2.2.3 Bag-of-Concepts

Rather than identifying features directly with some surface form, Sahlgren and Cöster [26] introduce a representation which they call *bag-of-concepts*, BoC. This representation is based on the intuition that the meaning of a document can be approximated by the union of the meanings of the terms in the document. To accomplish this a document is represented as the weighted vector sum of the concept vectors corresponding to the terms occuring in the document. This heuristic has to our knowledge not been used in the context of text classication. It is, however, used with the LSI method mentioned above, for creating query vectors in an IR system [5].

The assumption that the meaning of a document is simply the union, or weighted sum, of the meanings of the terms it contains have proven to work quite well in IR systems, despite its simplicity. As we will see later in chapter 7 it seems to work quite well in TC applications as well. However it has its limitations and it is not cognitively justified. The main weakness of the heuristic is that it makes no difference between different semantical categories of words and that it makes no use of contextual relations in the weighting of the term meanings. This is illustrated by Song and Bruza in [35] who propose a heuristic based on a general theory of *conceptual spaces* for combining term meanings. If and how this heuristic could be used in creating representations for text classification tasks is an interesting question which we due to time constraints are forced to leave open. Another interesting possibility is to use some form of *recurrent artificial neural network*, RANN, to model the contextual nature of meaning.

An interesting possibility, which may be beneficial for text classification, is that the concept vectors can be collected in an *unsupervised*[4] fashion. If the idea that the concept vector associated with a term adequately represents the meaning of that term, we should in principle be able to gain "knowledge" of the domain by collecting occurrence or co-occurrence statistics on any text from the domain. Hence, by pre-building the DOR or the TCOR on unclassified documents, we might be able to improve the performance of the system. This approach has to our knowledge not been used in the context of text classification.

2.2.4 Computational Considerations

The bag-of-words representations are often very sparse, since each document often contains only a small fraction of all terms in the lexicon. Constructing the vectors is also very computationally efficient (linear in the number of term occurrences per document). Constructing the bag-of-concepts representations, on the other hand, is highly computationally expensive. This is especially the case when using the TCOR with a wide sliding window or a large context, such as the whole document. Constructing the concept vectors with TCOR and document context has time complexity quadratic in the number of word occurrences per documents. The complexity of summing these concept vectors depends on the denseness of the vectors, which will vary across different terms. However, the resulting document vectors will be very dense.

[4] In the context of machine learning, the term *unsupervised* means that the learner works with data which is not pre-classified, see chapter 4.

With the DOR, or TCOR with a narrow sliding window, the situation is better. Since the concept vectors in the DOR is the dual of the document vectors in BoW, constructing and summing these vectors has time complexity linear in the number of term occurrences per document. The resulting document vectors will, however, be very dense in this representation as well.

2.3 Weighting

In this section we discuss how to weight the importance of terms. Importance in this context should measure how much the term contributes to the semantics of a particular document. In the case of BoW this weight is applied to the term frequncies. Whereas, in BoC representations we weight the importance of terms as in BoW, but we then apply the weighting to the whole concept vector representing the semantics of the term. The term weights in the TCOR representation are thus plain counts of co-occurrence frequency, and in DOR plain counts of document occurrences. This is not a particularly well founded heuristic, see [16] for a proposal of how to weight the individual dimensions of context vectors.

Over the years a great number of different term weighting schemes have been proposed in the IR community. Most of these consist of three independent parts, each based on a different assumption. The general formula for weighting of term t_i in document d_j takes the following form:

$$w_{ij} = tf_{ij} \times idf_i \times norm_j, \tag{2.1}$$

where the parts are based on the following assumptions:

- A term which appears multiple times in a document is more important[5] than a term which appears a single time. This gives the term frequency weight tf_{ij}.
- Rare terms are as important as common terms. This gives the inverse document frequency weight idf_i.
- Short documents are as important as long documents. This gives the normalization weight $norm_j$.

Zobel and Moffat [43] has performed the most extensive evaluation of the proposed weighting schemes. They estimate that the number of unique combinations of (2.1) in conjunction with BoW is of the order of 100.000.[6] Unfortunately they reach the conclusion that this "weight space" can not be evaluated analytically and of the experimentally tested versions no particular w_{ij} performed consistently better than the others.

It should be noted that while the weighting assumptions described above may be valid in the field of IR, they are not automatically justified for use in TC. For example, Sebastiani and Debole [33] show that replacing idf_i with a measure based on term t_i:s distribution among classes in training data may give better performance. Yang and Pedersen [41] furthermore show that these class based measures are highly correlated with the document frequency df_i of term t_i. Thus it may be the case that the assumptions from the field of IR are not valid, or that they are even counterproductive, in the context of TC. If this is the case a better characterization of the term weighting function would be: $w_{ij} = l_{ij} \times g_i \times norm_j$, where l_{ij} denotes the *local* weighting of t_{ij} and g_i denotes the *global* weighting of t_i [2].

[5] Importance in this context denotes the differential quality of terms, i.e. how useful the term is as a feature in the task at hand.

[6] The formula used by Zobel and Moffat has a slightly more complex form than (2.1). However, their general result should apply to the form presented here as well.

Since we are not interested in the weighting scheme per se in this thesis work, we leave open the problem of weighting and use a commonly used and simple weighting scheme which has given quite satisfactory results in IR as well as TC applications [33]:

$$tf_{ij} = \begin{cases} 1 + \log \#(t_i, d_j) & \text{if } \#(t_i, d_j) \geq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2.2}$$

where $\#(t_i, d_j)$ denotes the frequency of t_i in d_j.

$$idf_i = \log \frac{|\Omega|}{\#\Omega(t_i)}, \tag{2.3}$$

where $\#\Omega(t_i)$ denotes the number of documents in Ω in which $\#(t_i, d_j) \geq 1$.

$$norm_j = \frac{1}{\sqrt{\sum_{k=1}^{|\mathcal{T}|}(tf_{kj} \cdot idf_k)^2}}, \tag{2.4}$$

where the denominator equals the Euclidean norm $\|\vec{d_j}\|_2$ of document vector $\vec{d_j}$ before normalization. Equation (2.4) gives all document vectors unit length, i.e. all documents are represented as points on the $|\mathcal{F}|$-dimensional unit sphere.

2.4 Preprocessing

Any language contains words that bear no semantic content by themselves. In English words like for example "the", "it" and "or" are of no use for text classification, since they probably occur evenly over all classes. Therefore one often removes these *stop words* before creating the document representation. Another common preprocessing step is to remove all terms with less than k occurrences, usually with $1 \leq k \leq 5$, and to remove all documents with fewer than n words, usually with $5 \leq n \leq 20$. This is done to reduce noise stemming from the bad statistical estimates gained by these low frequency terms and too short documents.

In addition to these greedy preprocessing methods, to perform some form of morphological normalization has shown often to improve results in information retrieval. This helps to reduce problems with synonymy since words which share morphological root may be seen as semantically equal for most purposes. Morphological normalization can thus be seen as a way of creating more general features, see section 2.2.1. Two methods for performing the normalization is *stemming* and *lemmatization*.

Stemming uses simple transformation rules to remove common inflexional affixes. The transformation rules does not use any lexical information and so is not particularly accurate, for example the word *witness* will be transformed to the word *wit* using the *Porter stemmer* [23], which is the most common stemming algorithm for working with English text. This means that while stemming in many cases reduce problems with synonymy it sometimes make things worse by introducing polysemy. In order to alleviate this problem, the stemmer is often customized for the specific application.

Lemmatization is a less greedy approach in which words are transformed from their inflexional form into their root by use of more advanced language analysis (usually only on the word level). This can be implemented in the form of morphological transformation rules in conjunction with a lexicon. Another possibility is to use a lexicon which contain mappings of inflectional forms to morphological roots for all words. This will, however, be difficult in practice since there for many languages exist a large magnitude of different inflectional forms. Lemmatization is probably more useful with more inflectional languages than English for which stemming has proven sufficient in many applications.

Chapter 3

Dimensionality Reduction

In this chapter we discuss the need for dimensionality reduction and describe different methods for performing this reduction. The chapter is divided into three parts. In the first part we discuss why dimensionality reduction is useful. In the second part we describe variations of the feature selection approach which is divided into the general methods of filtering and wrapping. In the third part we describe the feature extraction approach with a focus on the random mapping technique. We end with a discussion of the incremental random indexing implementation of random mapping.

3.1 The Need for Dimensionality Reduction

The pre-processing described in the previous section somewhat reduces the dimensionality $|\mathcal{F}|$ of the document vectors. However, in most real world TC applications $|\mathcal{F}|$ is still of the order of 10.000 or even 100.000 terms. This is prohibitive for several reasons. Many learning algorithms are of quadratic or higher time and memory complexity in the dimensionality of data. Thus, high dimensionality may be intractable for strictly computational reasons. Furthermore, the classification quality of many learning algorithms scale poorly with the dimensionality of data [31]. However according to Joachims [10] this is not the case with the Support Vector Machine classifier, see section 4.3.

There are also problems concerning the data itself. Since a high-dimensional term space will inherently contain a high amount of low frequency terms, the data will probably contain a high degree of noise due to sampling errors. The problem is further aggravated in applications dealing with textual data. This is primarily due to the synonymous and polysemous nature of language.

Sampling noise is a problem in many machine learning classification methods where it leads to what is known as *overfitting*. This means that the automatically generated classifier captures aspects of the training data that are contingent with respect to classes, see section 4. According to Sebastiani [31], to completely avoid overfitting requires that the number of training documents is proportional to $|\mathcal{F}|$.

These problems can be alleviated by performing a reduction of the dimensionality of the data. This can be done by means of *feature selection* or *feature extraction* discussed in the following sections. We denote the new reduced set of features $\mathcal{F}'$ and define the reduction factor ξ, i.e. the relative reduction of dimensionality as [33]:

$$\xi = \frac{|\mathcal{F}| - |\mathcal{F}'|}{|\mathcal{F}|}. \tag{3.1}$$

Joachims [10] state that feature selection is not necessary and that it "merely covers up deficiencies of the learning algorithm" [10, p.18]. This statement is based on the fact that the ML algorithm used in [10], the Support Vector Machine, see section 4.3, is computationally efficient when dealing with sparse document vectors as those generated by the surface form models. This is generally the case for all ML methods working on inner-product spaces. When dealing with dense document vectors as those generated by the BoC model we believe that this statement does not hold regardless of which distance measure is being used. However, Joachims correctly points out that all the feature selection measures that fall within the filtering approach, described in the next section, rely on some independence assumption which in general is not sound.

When implementing multi-label classification as $|\mathcal{C}|$ binary classifiers we make a difference between *local* dimension reduction, i.e. a unique set $\mathcal{F}'_i$ is used by each classifier c_i, and *global* dimension reduction, i.e. the same set $\mathcal{F}'$ is used by all classifiers. When the classifiers are considered independent we prefer local reduction over global reduction, since the former can be adapted to each specific classification problem.

3.2 Feature Selection

The most straightforward way to achieve dimensionality reduction is to simply select a subset $\mathcal{F}' \subset \mathcal{F}$ of features and use only these for classification. This is known as feature selection and is equivalent to setting the dimensions that doesn't belong to $\mathcal{F}'$ to zero, an operation which is known as pruning. There are two different ways to perform feature selection: the *filtering* approach and the *wrapper* approach.

3.2.1 Filtering

In the filtering approach, $\mathcal{F}'$ is selected by the use of a value function which is independent of the classification algorithm. This value function is called *term evaluation function*, TEF, in the context of BoW [33]. Since different representations give different feature definitions we use the term *feature evaluation function*, FEF, instead. Usually a FEF which measures the correlation of each feature with each class is used. The most important are two functions from information theory, *Information Gain*, IG [41] and *Gain Ratio*, GR [33]. There are several ways to define these functions, the following definitions cover the case of binary classification.

Information Gain measures the number of bits of information for the binary classification under c_j that are gained by knowing the presence or absence of feature f_i:

$$IG(f_i, c_j) = \sum_{x \in \{f_i, \bar{f}_i\}} \sum_{y \in \{c_j, \bar{c_j}\}} \Pr(x, y) \log_2 \frac{\Pr(x, y)}{\Pr(x) \Pr(y)}, \tag{3.2}$$

where $\Pr(f_i, c_j)$ denotes the probability that a document randomly drawn from $\mathcal{C}$ contain feature f_i and belongs to class c_j; $\bar{f}_i$ and $\bar{c}_j$ denotes the fact that f_i is not present in the document and the fact that the document does not belong to class c_j, respectively.

Gain Ratio is a normalized form of Information Gain, where the latter is normalized with respect to the entropy of class c_j.

$$GR(f_i, c_j) = \frac{IG(f_i, c_j)}{H(y)}, \tag{3.3}$$

where the denominator denotes the entropy $H(y)$ of $y = \{c_j, \bar{c_j}\}$:

$$H(y) = -\sum_{y \in \{c_j, \bar{c_j}\}} \Pr(y) \log_2 \Pr(y). \tag{3.4}$$

The definitions above all treat features as boolean-valued. This has shown to work well with sparse representations as BoW in which features roughly can be estimated as having two values corresponding to zero and non-zero. However when we use dense representations in the form of BoC, features do not show this "on-off" behaviour. The solution to this is to apply some form of *discretization* to the continuous feature values and then apply the FEF. Thus we find an optimal split-point and calculate the FEF using this split-point.

When using global feature selection the "goodness" of feature f_i is evaluated either by computing the weighted average over classes:

$$FEF_{avg}(f_i) = \sum_{c_j \in \mathcal{C}} \Pr(c_j) FEF(f_i, c_j), \tag{3.5}$$

or the maximum over classes:

$$FEF_{max}(f_i) = \max_{c_j \in \mathcal{C}} FEF(f_i, c_j), \tag{3.6}$$

where the FEF is one of the functions in equations (3.2) and (3.3).

All three feature evaluation functions are supervised, i.e. they use information on the distributions of features over classes to assess the relevance of features. Interestingly, as noted in section 2.3, Yang and Pedersen [41] show that when using BoW the document frequency df_i of term t_i seems to be highly correlated with the IG_{avg} and χ^2_{max} and that it allows a reduction factor of 80–90% without a loss in effectiveness. This is probably the case for GR as well since it is a normalized form of IG. This, however, have not been experimentally verified. Neither has it been verified if it is correlated with IG_{max}. Yang and Pedersen further show that TEF_{max} seems to be the best globalization measure, at least for a χ^2-test. Sebastiani and Debole [33] reach the conclusion that this is the case for all three functions.

3.2.2 Wrapping

In the wrapper approach the actual classifier algorithm is taken into account.[1] This approach is motivated by the fact that different classifier learning algorithms are affected in different ways by feature selection. To capture this fact the *optimal feature subset* [15] is defined as the subset of features $\mathcal{F}' \subseteq \mathcal{F}$ such that the performance of the given classification algorithm is optimal with respect to a given measure of effectiveness.

The feature evaluation functions used in the filtering approach are defined to capture some notion of *relevance*. Kohavi and John [15] formulate two very general definitions of what they call *strong relevance* and *weak relevance* with respect to an *optimal Bayes* classifier which is a theoretically optimal classifier given complete knowledge of the underlying data distribution. Next they show that even the most theoretically justified definition of relevance does not entail an optimal feature subset and that an optimal feature subset does not entail relevance. This means that even the most relevant features can give suboptimal performance and that an optimal feature subset may include irrelevant features.

[1] This subsection is based on [15].

The only way to solve this problem in general is by using the classification algorithm itself (or in some cases knowledge about the algorithms performance under feature selection) in the selection of features. After evaluating every possible subset of features against the classifier, the optimal feature subset can be picked and used for classification. Unfortunately, since there are $2^{|\mathcal{F}|}$ different subsets this is computationally intractable in most practical applications. The subset space can be partially searched by using for example forward selection or backward elimination based on hill climbing or more advanced heuristics. The approximative solutions given by these algorithms, however, only match the filtering methods described above and this at a much higher computational cost.

3.3 Feature Extraction

The intuition underlying all feature extraction methods is that instead of selecting some dimensions of the feature space and just throwing away the others we can project, or map, the feature space into a new feature space of lower dimensionality. This new feature space should represent most of the information in the original space and the amount of noise introduced by this operation should be kept to a minimum. With some methods one even hopes to reduce noise.

3.3.1 Latent Semantic Indexing

The most famous feature extraction technique is probably *Latent Semantic Indexing*, LSI, which was introduced by Dumais et al. in the context of query expansion [6]. The method is also known under the name *Latent Semantic Analysis*. In LSI the original feature vectors (most commonly in the form of a terms-by-documents matrix) is mapped into a space of lower dimensionality using *singular value decomposition*, SVD. By keeping only the k largest singular values and its corresponding singular vectors, the best rank-k approximation (as measured by the Frobenius-norm) of the original feature vectors is gained [2]. It has been experimentally verified that LSI with an appropriate choice of k, usually $50 \leq k \leq 200$, not only reduces the dimensionality of the feature space, but that it also partially solves the problem of synonymy. A commonly used explanation for this is that semantically similar features and documents are mapped to closer regions in the new space. See [22] for a more rigorous analysis based on a primitive probabilistic corpus model.

LSI can be used in TC applications by first performing the SVD on the training set and then applying the mapping on the training set and the test set. According to Sebastiani [31] one benefit of this mapping is that a set of features which together give strong support for a certain document class can be mapped to the same feature in the new space, while a potential drawback is that a feature which in the original space gave strong support for a certain class can be weakened.

The main drawback of LSI is its computational complexity. Performing the SVD takes time $O(|\mathcal{D}||\mathcal{F}|c)$, where c is the average number of non-zero entries per column in the features-by-documents matrix, i.e. the average number of features per document. Using random mapping, described in the next section, as a pre-processing step Papadimitriou et al. [22] show that the computational complexity can be reduced substantially with only a marginal amount of added noise.

3.3.2 Random Mapping

A much more computationally efficient method for performing the dimensionality reduction is *Random Mapping*, RM, also known as *Random Projection*. In this method the original

space is mapped onto a lower dimensional subspace. This is done by multiplying the original $s \times d$ matrix A, consisting of s d-dimensional samples, by a random $d \times k$ matrix R, resulting in the reduced-dimensional $s \times k$ matrix B:

$$B_{s \times k} = A_{s \times d} R_{d \times k}. \tag{3.7}$$

The method can be seen as an implementation of the following lemma by Johnson and Lindenstrauss (cited from [1]): Given $\gamma > 0$ and an integer n, if we let integer $k \geq O(\gamma^2 \log n)$, then for every set P of n points in $\Re^d$ there exists a map $f : \Re^d \mapsto \Re^k$ such that for all $u, v \in P$

$$(1-\gamma)\|u-v\|^2 \ \leq \ \|f(u)-f(v)\|^2 \ \leq \ (1+\gamma)\|u-v\|^2.$$

Note that the lemma only guarantees that a map with this property exists. It tells nothing about how to construct the actual map. However, there are constructive versions of the lemma as well. In most of these lemmas f is a linear map in the form of a random matrix with elements independently $N(0,1)$-distributed, i.e. distributed as the Normal random variable with mean 0 and variance 1. Such dense real valued matrices are unfortunately not optimally useful in practice for computational reasons.

However, alternatives exist and Achlioptas [1] has shown that any given matrix R with elements r_{ij} being independent random variables from either one of the following probability distributions satisfies a version of the lemma which tighter bounds the degree of perturbation:

$$r_{ij} = \frac{1}{\sqrt{k}} \cdot \begin{cases} +1 & \text{with probability} \quad 1/2 \\ -1 & .. \qquad\qquad 1/2 \end{cases}, \tag{3.8}$$

$$r_{ij} = \frac{3}{\sqrt{k}} \cdot \begin{cases} +1 & \text{with probability} \quad 1/6 \\ 0 & .. \qquad\qquad 2/3 \\ -1 & .. \qquad\qquad 1/6 \end{cases}. \tag{3.9}$$

Note that in practice the scaling factors are taken into account after the random mapping is performed. Thus R only contains 0:s and ± 1:s.

The distributions above can be generalized by varying the dimensionality k and the denseness e of the random matrix [27]:

$$r_{ij} = \frac{1}{e} \cdot \begin{cases} +1 & \text{with probability} \quad \frac{e/2}{k} \\ 0 & .. \qquad\qquad \frac{k-e}{k} \\ -1 & .. \qquad\qquad \frac{e/2}{k} \end{cases}. \tag{3.10}$$

Since the number of $+1$'s and -1's must be equal according to this scheme, the value of e must be an even integer. In the following we use $e \approx a$, where a is some real number, to refer to the smallest even integer $\geq a$.

The merit of the lemma by Johnson and Lindenstrauss, and hence the constructions of Achlioptas, is that they give theoretical bounds on the perturbation of the Euclidean length of arbitrary vectors. This means that we also have bounds on the amount of distortion of relative document similarities as measured with the *Euclidean distance*. The lemma does, however, not say anything about how *inner products* are distorted and we can not a priori assume that the distortion is of the same magnitude in this case. On the contrary results reported by Bingham and Manilla [3] indicate that distance measures based on the inner product is more sensitive to distortion under random mapping.

We instead turn to Kaski [14] who provide an analysis of the distortion of inner products caused by the random mapping. The downside is that he assumes that the elements of R

are $N(0,1)$ distributed. The following derivation have been slightly reformulated from the one in [14].

Let R be a $d \times k$ random matrix with rows normalized to Euclidean unit length. Furthermore, let $\vec{m}$ and $\vec{n}$ be two arbitrary vectors in the original space. After a random mapping according to (3.7) the inner product of the resulting vectors $\vec{x}$ and $\vec{y}$ can be written as:

$$\vec{x} \cdot \vec{y}^\top = \vec{m} R R^\top \vec{n}^\top. \tag{3.11}$$

The inner product of the vectors in the reduced-dimensional space is thus distorted by the matrix $RR^\top$. If this matrix was equal to the identity matrix I, then the distortion would be zero. Let ϵ be the deviation matrix of $RR^\top$ from the identity matrix:

$$\epsilon = RR^\top - I. \tag{3.12}$$

Since the rows of R have been normalized to unit length, $\vec{r_i} \cdot \vec{r_j}^\top = 1 = I_{ij}$ for $i = j$. Hence:

$$\epsilon_{ij} = \begin{cases} \vec{r_i} \cdot \vec{r_j}^\top, & \text{for } i \neq j \\ 0, & \text{otherwise} \end{cases}. \tag{3.13}$$

Distributivity together with (3.11) – (3.13) gives:

$$\vec{x} \cdot \vec{y}^\top = \vec{m} \cdot \vec{n}^\top + \vec{m} \epsilon \vec{n}^\top = \vec{m} \cdot \vec{n}^\top + \sum_{i \neq j} \epsilon_{ij} m_i n_j. \tag{3.14}$$

From (3.14) we see that the distortion on inner products imposed by random mapping depend on the distribution of ϵ_{ij} as well as on the distribution of $m_i n_j$.

Turning first to ϵ_{ij}, it is simple to show that if the elements r_{ij} of the random matrix R are distributed with mean 0, then ϵ_{ij} is distributed with mean 0 as well. Furthermore, Kaski [14] and also Bingham and Manilla [3] show that if the elements of R are independently $N(0,1)$-distributed, then the variance of ϵ_{ij} is $1/k$, where k is the number of columns of R. For the distributions above we know of no bounds on the variance of ϵ_{ij}. For an illustration of how the distribution of ϵ_{ij} behave for different values of k and e, where e is the number of non-zero entries per row in R as defined in (3.10), see figure 3.1.

Turning to $m_i n_j$, the situation is the same and there are to our knowledge no theoretical analysis of its distribution in the context of textual data, although Kaski, again, notes that the probability of $m_i n_j$ being non-zero is proportional to the *sparseness* of the original data vectors, i.e. to the ratio of non-zero elements and zero elements. Unfortunately, random mapping is most useful when the original data is dense. The reason for this is that it is often possible to optimize vector similarity calculations for sparse data and to store sparse data requires memory proportional to the number of non-zero elements per vector. In general when we use machine learning methods which can be highly optimized for sparse data we would only consider random mapping as a feature reduction technique if the mean number of non-zero features per sample is larger than k.

3.3.3 Random Indexing

Sahlgren and Cöster [26] propose that, in the context of document representations based on DOR or TCOR, the random mapping should be implemented in an incremental fashion. In conventional random mapping one would first build the original $s \times d$ matrix by counting document occurrences (DOR) or term co-occurrences (TCOR) and then this matrix would be mapped to the lower dimensional subspace by multiplication with the $d \times k$-dimensional random matrix.

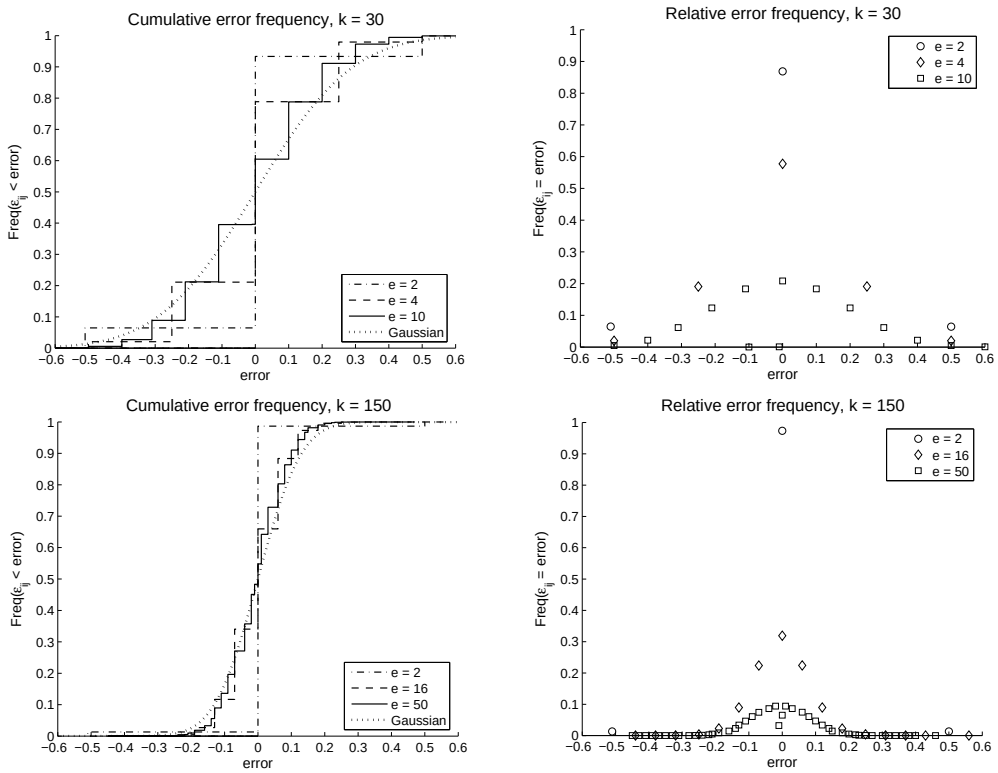


Figure 3.1. Distribution of ϵ_{ij} for $k = 30$ and $k = 150$ together with $e \approx k/100$, $e \approx k/10$ and $e \approx k/3$. The cumulative Gaussian with mean 0 and variance $1/k$ is plotted for reference. It is clear that the cumulative distribution for $e \approx k/3$ approaches a cumulative normal distribution with mean 0 and variance $1/k$ faster than the other values of e. From the graphs one can also see that it is possible to make a trade-off between the size of the distortion and the frequency of distortion.

Random Indexing, RI, instead works incrementally and assigns a k-dimensional sparse *random label*, i.e. a sparse random vector, to each unique document (DOR) or unique term (TCOR). In the case of DOR, every time a term occurs in a document its random label is added to the context vector associated with the term. In the case of TCOR, every time a term co-occurs with another term their respective random labels are added to the others term vector. Since matrix multiplication is distributive, the matrix consisting of all these *index vectors*[2] after indexing t terms will be equivalent to the matrix gained if we had stopped building the $s \times d$ matrix after the same t-th term and then performed the random mapping. The method has been used for analyzing term semantics [12] as well as for BoC-representations [26].

When working with dense representations the benefit of RI is clear in terms of memory consumption, since we only have to store the d e-dense k-dimensional random labels together with the s approximately k-dense k-dimensional index vectors. However, the computational complexity is worse than for standard random projection, as the following simple analysis shows.

Beginning with Random Indexing with DOR it is easily seen from algorithm 1 that the

[2]In papers on random indexing, the term *index vector* is usually used instead of *term vector* and *context vector*.

total time complexity is $O(DTe)$, where D is the number of documents, T is the average number of term occurrences in each document and e is the number of non-zero elements per random label.

Random Mapping with DOR, see algorithm 2, on the other hand, has time complexity as follows. Building the matrix takes time $O(DT)$. If we use sparse representations for A and R the matrix multiplication then takes time $O(DT'e)$, where T' is the average number of unique terms per document. This gives a total time of $O(DT + DT'e)$, ignoring the creation of the random labels which takes time $O(De)$ with both methods. When $T' < T$, RI will be significantly slower for sufficiently large values of e. Due to its simplicity we leave out an analysis for the case of TCOR, which shows a similar result.

Algorithm 1 Random Indexing – DOR

```
procedure RandomIndexing.DOR(D,k,e)
    for all documents d ∈ D do
        l_d ← CreateDocLabel(k,e)   ▷ Creates a k-dimensional e-dense random vector
        for all term occurrences t' ∈ d of term t ∈ T do
            B_t ← B_t + l_d          ▷ Add random label to row t of resulting matrix B
        end for
    end for
    return B
end procedure
```

Algorithm 2 Random Mapping – DOR

```
procedure RandomMapping.DOR(D,k,e)
    for all documents d ∈ D do
        for all term occurrences t' ∈ d of term t ∈ T do
            A_{t,d} ← A_{t,d} + 1          ▷ Update occurrence count of (t, d)
        end for
    end for
    B ←RandomMap(A,k,e)
    return B
end procedure

procedure RandomMap(A,k,e)
    for i ← 1, rows(A) do
        R_i ←CreateDocLabel(k,e)   ▷ Creates a k-dimensional e-dense random vector
    end for
    B ← A · R
    return B
end procedure
```

RM should thus be faster than RI, but requires more memory. If memory constraints are moderate, we propose that the random mapping is implemented in a batch-wise fashion, thus striking a balance between speed and memory consumption. *Batch-wise RM* works as follows: The set of documents is partitioned into batches and the matrix of frequency counts is collected for each batch. After each batch is collected it is multiplied by the random matrix and the resulting matrix is added to the matrix from the previous batch.

Chapter 4

Classifier Learning

In this chapter we give a brief background to, and an account of the different machine learning methods used in the thesis. We begin by defining the learning situation. Then we describe the concept of separating hyperplanes and a simple training rule for learning such hyperplanes. We go on to describe the concept of structural risk minimization and how to implement it in the form of a Support Vector Machine. Finally we briefly describe the AdaBoost and Decision Stump algorithms which are used for reference and as a baseline classifier respectively.

4.1 The Learning Situation

We face the following learning situation: given a set of labeled documents, how should we label new unseen documents? In most cases this is a *knowledge free* problem which means that no formalized meta-data or extrinsic knowledge are available [31]. Since we only have ostensive definitions of classes available we need to perform some form of *inductive learning*. This kind of learning situation have been studied by the *machine learning*, ML, community for some decades now. See [21] for a good overview of general machine learning techniques.

With some exceptions e.g. [40], almost all ML approaches to multi-label text classification treat the classification problem as consisting of $|\mathcal{C}|$ binary classification problems. When a class hierarchy is available the dependence of classes can be exploited to gain efficiency [42] and also effectiveness [4], i.e. better as well as faster classification. Unfortunately due to time constraints we have not been able to implement this approach and so in the following we make the assumption that classes are independent.

Recall from section 1.2 that we want to approximate the function $\breve{\Phi} : \mathcal{D} \times \mathcal{C} \to \{T, F\}$. Formalizing the multi-label binary classification problem we seek the functions $\Phi_i^b : \mathcal{D} \to \{T, F\}$ for all $i \in \{1, \ldots, |\mathcal{C}|\}$. The machine learning algorithms we will use do not work directly with text documents and they presuppose that each binary classification problem is under the set $\{\pm 1\}$. We therefore use a map $\mathcal{D} \times \mathcal{C} \mapsto \mathcal{X} \times \{\pm 1\}^{|\mathcal{C}|}$, where $\mathcal{X}$ is the *feature space*, i.e. a $|\mathcal{F}|$-dimensional vector space.

When using this representation, instead of seeking the functions Φ_i^b we seek the functions $h_i : \mathcal{X} \to \{\pm 1\}$ for all $i \in \{1, \ldots, |\mathcal{C}|\}$. Since we work in a vector space model the only principal constraint on the functions h_i is that they are able to separate the feature space into regions.

To ease notation and exposition we will drop references to the $|\mathcal{C}|$ independent classification problems and concentrate on the case of a single binary problem. This is possible, since by the independence assumption all conclusions reached in the following sections transfer directly to the multi-label case.

Before we continue to describe the different learning algorithms a note on notation is in place. We refer to the vector representations $\vec{x_i} \in \mathcal{X}$ of documents $d_i \in \mathcal{D}$ as *instances*. By $y_i \in \{\pm 1\}$ we denote the label of instance $\vec{x_i}$. The combination $(\vec{x_i}, y_i)$ of instances and labels are referred to as *examples*. By Ω we denote the set of all pre-classified examples. The functions h are referred to as *hypotheses*. The general inductive process which are to infer the hypothesis h from Ω is called the *learner*. The set $\mathcal{H}$ of hypotheses which the learner is allowed to work with given a certain feature space $\mathcal{X}$ is called the *hypothesis space*.

Since manual text classification is a subjective practice, i.e. classes are not formally defined, the learner is forced to learn only from examples. This implies the use of *supervised* or semi-supervised a.k.a. *bootstrapping* machine learning methods. The latter may be the preferable choice when examples are scarce. We are only interested in supervised methods in this thesis.

Over the years a wide range of supervised machine learning approaches to text classification has been proposed, see e.g. [31], [32] or [39] which are great surveys of the area. Notable ML-methods used are amongst others *Naive Bayes*, *k-Nearest Neighbour*, *Artificial Neural Networks*, *Decision Trees*, *Support Vector Machines* and meta-methods such as *Bagging* and *Boosting*.

In this thesis we concentrate on the Support Vector Machine and Boosting, see sections 4.3 and 4.4. The reason for this is threefold. First of all they have proven to be the most effective algorithms in many cases. Secondly, they have also proven to be among the most efficient algorithms. Thirdly, they are the most motivated according to *computational learning theory*, i.e. the formal study of learning algorithms [32]. Thirdly, the algorithms work quite differently to achieve similar goals. While SVM uses all the features for classification, Boosting of Decision Stumps use only a handful. This allows us to assess the quality of individual features by comparing the results for the two algorithms.

4.2 Separating Hyperplanes

Before we go on to explain the *Support Vector Machine* learning method we need to understand the concept of *separating hyperplanes*. It is also instructive to look at a very simple algorithm for finding these hyperplanes.

Suppose we have a set of examples (X, Y), where $x \in \Re^2$ and $y \in \{\pm 1\}$, i.e. the examples are labeled points in the plane. Under a certain labeling a separating hyperplane is a line which divides the plane into two regions, each containing only points labeled $+1$ or only points labeled -1, see the left of figure 4.1. The situation is analogous for the case of $\Re^N$, in which we have an $N-1$ dimensional hyperplane. Note that not all labelings of the examples can be separated by a hyperplane, see the right of figure 4.1. If, under a certain labeling, there exist a hyperplane which separates all the examples we say that the examples are *linearly separable* under that labeling.

If $\mathcal{X}$ is an inner product space, then the following function implicitly define a separating hyperplane [10]:

$$h(\vec{x}) = sign(\vec{w}^\top \vec{x} + b) = \begin{cases} +1, & \text{if } \vec{w}^\top \vec{x} + b > 0 \\ -1, & \text{otherwise} \end{cases}, \tag{4.1}$$

where $\vec{x} \in \mathcal{X}$ is an instance, $\vec{w} \in \mathcal{X}$ is the *weight vector* perpendicular to the hyperplane

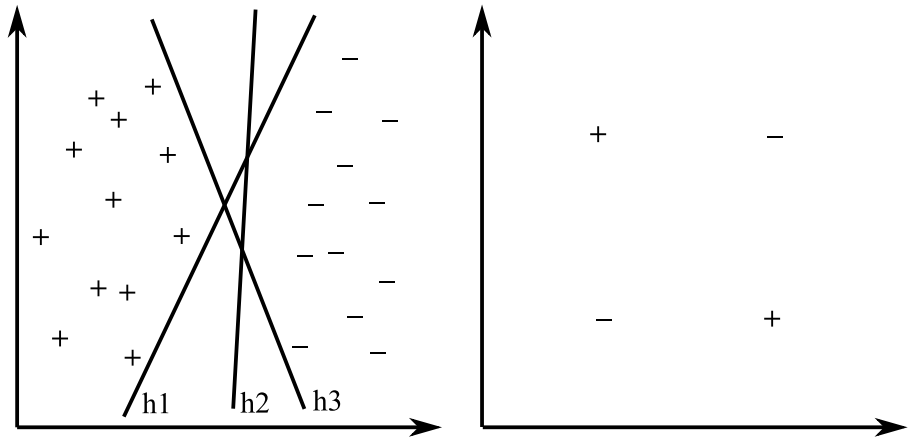


Figure 4.1. Left: Three different hyperplanes separating positive examples + from negative examples −. Right: The famous XOR-problem in two dimensions. There exist no hyperplane which separates all the positive examples from the negative.

and $b \in \Re$ is the *bias*. For the case $b = 0$, (4.1) define a separating hyperplane through the origin. For the case $b \neq 0$ the situation is more complicated. The direction of $\vec{w}$ still defines the direction of the hyperplane, while the length $\|\vec{w}\|$ together with b defines the distance of the hyperplane from the origin.

4.2.1 The Perceptron

We now turn to a simple algorithm for finding a separating hyperplane. The *perceptron* algorithm was introduced by Rosenblatt in 1957 and Minsky and Papert in 1969 proved it to converge to a separating hyperplane, given that one exists for the current labeling. In the case of $b = 0$ it works as follows [21]:

The weight vector $\vec{w}$ is initialized to a random vector in $\mathcal{X}$. Then iterating through the training examples $(x_i, y_i) \in \Omega$, at each step the weight vector is updated according to the *perceptron training rule*:

$$\vec{w_{i+1}} \leftarrow \vec{w_i} + \eta(y_i - f_i(\vec{x_i}))\vec{x_i}, \tag{4.2}$$

where $\vec{w_i}$ is the weight vector at iteration i, η is the *learning rate*, typically a small number, e.g. 0.1, and $f_i(\vec{x_i})$ is the output of (4.1) at iteration i. At each iteration, if the current weight vector makes f_i misclassify instance $\vec{x_i}$, then the weight vector is either pushed towards or away from $\vec{x_i}$ depending on how the instance was misclassified, i.e. depending on the sign of $y_i - f_i(\vec{x_i})$.

The perceptron algorithm works remarkably well in many cases when examples are linearly separable. One downside is that examples are not always linearly separable. Another downside is that it *generalizes* poorly, i.e. even though training examples may be classified correctly performance on unseen instances from the same distribution may be bad. Why this is the case and how it can be handled is explained in section 4.3.

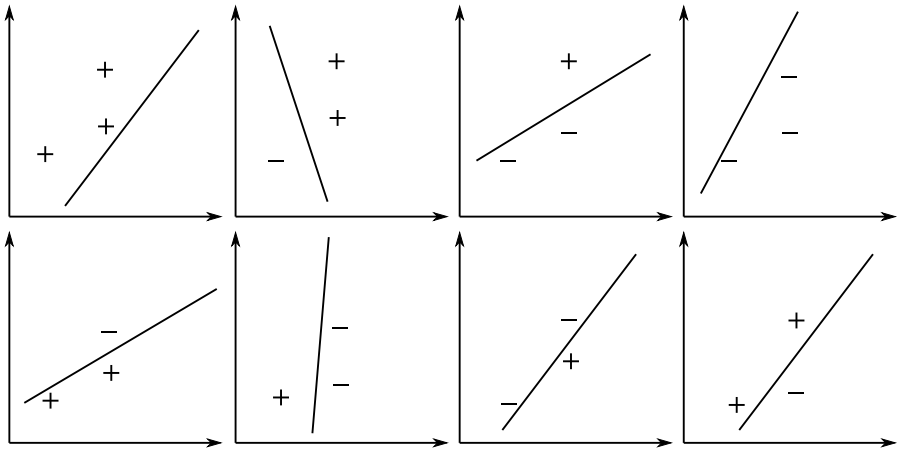


Figure 4.2. A set of 3 instances in $\Re^2$, shattered by hyperplanes.

4.2.2 Vapnik-Chervonenkis Dimension

We have mentioned that a given set of instances may not be linearly separable under a certain labeling. A natural question then is: can we be certain whether a set of instances are linearly separable irrespective of their labeling? Fortunately this question is answered in a very general way by *Vapnik-Chervonenkis* theory, VC-theory. See e.g. [21] or [30] for comprehensive overviews, or [38] for a complete account.

A fundamental notion in VC-theory is the *VC-dimension* of a hypothesis space. This is based on the concept of *shattering*. If for a set of instances a certain hypothesis space $\mathcal{H}$, e.g. separating hyperplanes, can separate the instances irrespective of their labeling we say that the hypothesis space *shatters* the set of instances. Now the VC-dimension, $VC(\mathcal{H})$ of hypothesis space $\mathcal{H}$ defined over feature space $\mathcal{X}$ is defined as the cardinality of the largest finite subset of $\mathcal{X}$ shattered by $\mathcal{H}$ [21].

We are now able to answer the question posed above. If the hypothesis space $\mathcal{H}$ consists of separating hyperplanes defined over $\mathcal{X} = \Re^n$, then it can be proved that $VC(\mathcal{H}) = n+1$, i.e. arbitrary subsets of $n+1$ instances in $\Re^n$ are linearly separable. This is illustrated for the case $n = 2$ in figure 4.2.

4.3 Support Vector Machines

The relationship between the dimensionality of the feature space and the VC-dimension of the corresponding hypothesis space implies that given a sufficiently high dimensional feature space $\mathcal{X}$, subsets of $\mathcal{X}$ will be linearly separable. Another way to look at this is that the expressiveness of separating hyperplanes is proportional to the VC-dimension of the hypothesis space.

To see why this is important we have to look at a result from statistical learning theory, but first we need to define two forms of *risk* which arises with inductive learning of classifiers.

4.3.1 Structural Risk Minimization

The first form of risk of interest to us is the *empirical risk* which measures the average *loss*, i.e. the performance according to some measure, over training examples [30]:

$$R_{emp}(h) = \frac{1}{m}\sum_{i=1}^{m} L(h(\vec{x_i}), y_i), \tag{4.3}$$

where $L(h(\vec{x}), y)$ is some *loss function*. In this thesis we use the *zero-one* loss function, which only measures if the classification was correct or not [10]:

$$L_{0/1}(h(\vec{x}), y) = \frac{1}{2}|h(\vec{x}) - y| = \begin{cases} 0 & \text{if } h(\vec{x}) = y \\ 1 & \text{otherwise} \end{cases}. \tag{4.4}$$

As we will see in chapter 5, this loss function is not optimal for use in text classification tasks. The reason we use it in this thesis is that there were no SVM implementations available which could handle other loss functions. However, recently Joachims have shown how to construct SVM:s which handles practically arbitrary loss functions [11].

The second form of risk is referred to as the *expected risk* or the *generalization error*. This is defined as the expected loss averaged over unseen examples drawn from the underlying distribution $\Pr(\vec{x}, y)$ [30]:

$$R(h) = \int L(h(\vec{x}), y)\, \mathrm{d}\Pr(\vec{x}, y). \tag{4.5}$$

In practice this risk is estimated by sampling the underlying distribution and counting the loss as measured with the loss function L, i.e. the integral in 4.5 is replaced by the weighted sum of loss on a set of samples.

The fundamental problem in inductive learning is that a good performance as measured by (4.3) does not imply a good performance as measured by (4.5). This is a problem with for example the perceptron which generalizes poorly, i.e. the actual risk is much higher than the empirical risk. This happens with the perceptron because it converges to a hyperplane which is not necessarily the best one. More generally the problem is that when the hypothesis space has a too high *capacity*, i.e. the hypotheses can capture too complex relationships in the feature space, then a hypothesis may have zero loss over training examples and still perform very bad on all other instances. Thus by decreasing the capacity of the hypothesis space we should be able to improve generalization. As noted above $VC(\mathcal{H})$ can be seen as a measure of the expressiveness of the hypotheses in $\mathcal{H}$, i.e. the capacity of $\mathcal{H}$.

This fundamental relationship is captured by the following result from statistical learning theory which bounds the risk by the sum of the empirical risk and a capacity term [10]:

$$R(h) \leq R_{emp}(h) + O\left(\frac{VC(\mathcal{H})\ln(\frac{n}{VC(\mathcal{H})}) - \ln(\eta)}{n}\right), \tag{4.6}$$

where n is the number of training examples over which $R_{emp}(h)$ is measured and the inequality holds with probability at least $1 - \eta$. The theorem in effect states that in order to gain a low risk we must find a VC-dimension that is sufficiently high to capture the complexity of the feature space and at the same time keep it as low as possible to get good generalization. This strategy is referred to as *structural risk minimization*.

To achieve structural risk minimization we must find a way to control the VC-dimension of the hypothesis space. Fortunately there is a straightforward way to do this in the case of separating hyperplanes: just change the *margin*, see figure 4.3, between the hyperplane and the training examples. Formally the margin δ is defined as the distance from the separating hyperplane to the closest examples. Thus if we impose a higher to lower ordering on the hypothesis spaces $\delta(\mathcal{H}_1) \geq \ldots \geq \delta(\mathcal{H}_i) \geq \ldots$ of separating hyperplanes according to their margin, then this ordering corresponds to a lower to higher ordering of the VC-dimensions

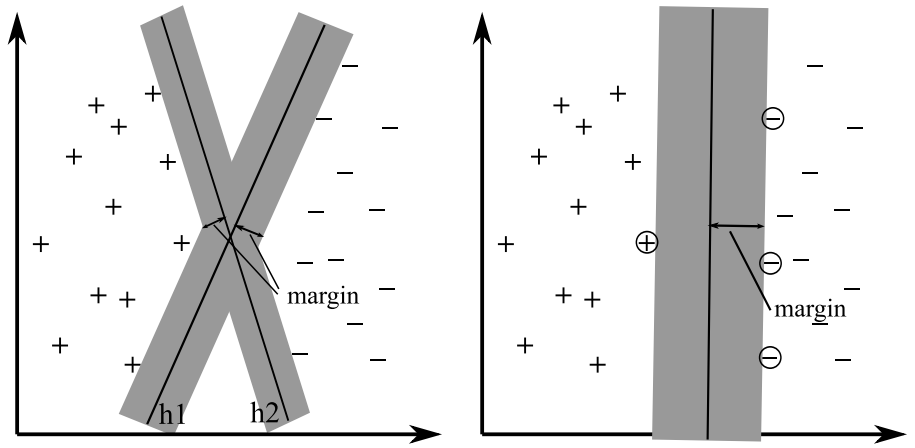


Figure 4.3. Left: Even though the margin constraints the placement of the separating hyperplanes there still exists more than one separating hyperplane. Right: When the margin is maximized there exists only one separating hyperplane. The training examples lying on the margin, here marked with circles, are called *support vectors*, and they uniquely define the separating hyperplane.

$VC(\mathcal{H}_1) \leq \ldots \leq VC(\mathcal{H}_i) \leq \ldots$. This fundamental fact was discovered by Vapnik who also proved a bound of the VC-dimension in terms of the margin, see for example [38].

4.3.2 Hard Margin SVM

The definition of separating hyperplanes in section 4.2 did not include any reference to the margin. However, for an algorithm to implement structural risk minimization in the form sketched out above this is needed. The following constraints define what is referred to as the *canonical* separating hyperplane:

$$\begin{array}{ll} \vec{w}^\top \vec{x} + b \geq 1 & \text{when } y = 1, \\ \vec{w}^\top \vec{x} + b \leq -1 & \text{when } y = -1, \end{array} \tag{4.7}$$

where the equalities hold for the positive and negative *support vectors* respectively, i.e. the examples lying on the margin, see right of figure 4.3. The inequalities state that all examples should lie on the correct side of the separating hyperplane, see right of figure 4.3. When these constraints are fulfilled for all examples $(\vec{x}, y)$ it can be shown that

$$\delta = \frac{2}{\vec{w}^\top \vec{w}} = \frac{2}{\|\vec{w}\|^2}, \tag{4.8}$$

where δ is the margin as defined above.

Combining (4.7) and (4.8) give rise to the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\vec{w}\|^2, \\ \text{subject to} & y_i(\vec{w}^\top \vec{x_i} + b) \geq 1, \forall_{i=1}^{l}, \end{array} \tag{4.9}$$

where l is the cardinality of the training set.

Since the vector norm is a convex function and the constraints are linear and thus convex the optimization problem is convex. This means that solving the optimization problem gives the unique optimal separating hyperplane [30]. In practice (4.9) is solved by using *Lagrange*

multipliers to rewrite the optimization problem into a form solvable by standard *Quadratic Programming* methods, see [10] or [30] for details.

The solution of (4.9) is expressed as a linear combination of the Lagrange multipliers α_i and the examples $(\vec{x}_i, y_i)$:

$$\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i, \tag{4.10}$$

where $\alpha_i \geq 0$, with $\alpha_i > 0$ iff $\vec{x_i}$ is a support vector.

Combining 4.10 and the linear threshold function of 4.1 give rise to the Support Vector Machine classification function:

$$h(\vec{x}) = \text{sign}(\vec{w}^\top \vec{x} + b) = \text{sign}(\sum_{i=1}^{l} \alpha_i y_i \vec{x_i}^\top \vec{x} \ + \ y' - \vec{w}^\top \vec{x}'), \tag{4.11}$$

where $(\vec{x}', y')$ is an arbitrary example with $\vec{x}'$ being a support vector. The expression for b follows directly from (4.7).

4.3.3 Soft Margin SVM

The derivation of the Support Vector Machine given above is only valid when the examples are linearly separable. This can be seen in the definition of the canonical separating hyperplane (4.7), where the constraints impose a strict separation of positive and negative examples. However, as was noted in the discussion of the perceptron, it is not always the case that examples are linearly separable. Fortunately this can be taken care of by introducing *slack* variables ξ_i and rewriting (4.7) into

$$\forall_{i=1}^{l}\colon y_i(\vec{w}^\top \vec{x_i} + b) \geq 1 - \xi_i, \tag{4.12}$$

with $\xi_i \geq 0$ and $\xi_i > 0$ if $(\vec{x_i}, y_i)$ lie inside the margin and $\xi_i > 1$ if $(\vec{x_i}, y_i)$ lie on the wrong side of the separating hyperplane.

Using (4.12) instead of (4.7) the *Soft Margin SVM* is constructed by solving the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l}\xi_i, \\ \text{subject to} & y_i(\vec{w}^\top \vec{x_i} + b) \geq 1 - \xi_i, \forall_{i=1}^{l}, \end{array} \tag{4.13}$$

where C is a parameter controlling the trade off between the margin size and the sum of errors. Hence, this optimization problem make implicit use of the $L_{0/1}$ loss function as discussed above.

The upshot of this formulation is that the method works when examples are not linearly separable, which is often the case in real world situations, either due to the complexity of the problem or due to noise. The downside is that we have turned a parameter free method into one that is quite sensitive to parameter choice, see chapter 7.

4.3.4 Kernels for Non-linear Mappings

Soft margin SVM is one way of handling the case when data are non-separable. Another method, which can be combined with the former, is to non-linearly map the feature space into another feature space of higher dimensionality, i.e. instead of working in $\mathcal{X}$ we work in $\phi(\mathcal{X})$, where ϕ is a non-linear map.

Another motivation for using such non-linear maps is that even though a sufficiently high dimensionality of the feature space often implies that examples are linearly separable, it is not always the case that a linear function in the feature space best capture the relationships between instances and labels. It might thus be possible to find a non-linear function that better matches the complexity of the problem at hand.

Since the computational complexity of SVM depend on the dimensionality of the instances we would like to be able to perform the calculations in the low dimensional space, while implicitly working in the high dimensional space. This means that instead of calculating the inner product of $\phi(\vec{x_i})^\top \phi(\vec{x_j})$ in the high dimensional feature space $\phi(\mathcal{X})$ we want to calculate the inner product in $\mathcal{X}$. This is exactly what is done by a certain class of functions called *kernels*. These are functions which when used with the optimization problems defined above in place of the ordinary inner product, allows us to find an optimal separating hyperplane in $\phi(\mathcal{X})$ which corresponds to a non-linear decision function in $\mathcal{X}$. No changes to the optimization problems are needed, we only substitute all inner products $\vec{x_i}^\top \vec{x_j}$ with $\kappa(\vec{x_i}, \vec{x_j})$, where κ is a kernel.

The most popular kernels for use with SVM are [30]:

$$\begin{array}{ll} \text{Linear :} & \kappa(\vec{x_i}, \vec{x_j}) = \vec{x_i}^\top \vec{x_j}, \\ \text{Polynomial :} & \kappa(\vec{x_i}, \vec{x_j}) = (\vec{x_i}^\top \vec{x_j} + 1)^d, \\ \text{Radial Basis :} & \kappa(\vec{x_i}, \vec{x_j}) = e^{\frac{1}{2\sigma^2}\|\vec{x_i} - \vec{x_j}\|}, \end{array} \tag{4.14}$$

where d is the degree of the polynomial and σ^2 is the variance of the Gaussian. The linear kernel of course is no non-linear map but is included for completeness.

4.4 Boosting of Decision Stumps

Even though the Support Vector Machine is the algorithm of primary concern in this thesis, we use two other classification methods for comparison. In the following section we briefly describe a meta-algorithm called *AdaBoost*, which make use of *Decision Stump*, a very simple learning algorithm described in section 4.4.2.

4.4.1 AdaBoost

The intuition underlying *ensemble* methods is that the combination of several classifiers might perform better than a single classifier on its own. This has shown to be a good intuition and has been implemented successfully in the methods of *bagging* and *boosting*. In this thesis we choose boosting over bagging, since even though the methods are quite similar, boosting has a stronger theoretical foundation in computational learning theory and has shown to outperform bagging in most applications, even if at a slightly higher variance [24].

The boosting method is a general method based on the fact that so called *weak* learners, i.e. learners which perform just better than random guessing, can be combined to form a *strong* learner, i.e. a learner which performs arbitrarily good, given sufficient time and data. This has a formal interpretation in the *PAC* model of computational learning theory. Since this is a brief overview of the method we ignore these theoretical aspects and refer the reader to for example [20] for an introduction to these issues.

There are several different versions of the general boosting method e.g *AdaBoost* and *LogitBoost* [20]. We use the former in the form of *AdaBoost.M1* since it is the classical boosting method which means that it is well documented and works quite straightforwardly. The main idea underlying AdaBoost, short for *adaptive boosting*, is that by training a set of weak

learners sequentially and weighting training examples based on how difficult they are for the previous weak learners, the current weak learner can concentrate on classifying these difficult examples correctly. The weights can thus be seen as imposing a non-uniform distribution on the examples in Ω. After all the weak learners are trained on the weighted examples, a combined learner is created as a weighted linear combination of the weak learners. Here a weak learner is weighted higher if it performed well on the training examples.

Formalizing the idea above, let $t = 1, \ldots, T$ denote the rounds that are performed sequentially and let D_i^t be the distribution on training example i at round t. Furthermore, let $h_t : \mathcal{X} \rightarrow \{\pm 1\}$ denote the hypothesis output by the weak learner $\mathcal{L} : \mathcal{X} \times \{\pm 1\} \times D \rightarrow \mathcal{H}$ trained at round t under distribution D^t and let α_t be the weight of $h_t(\vec{x})$ in the final combined hypothesis. Then AdaBoost works as described in algorithm 3 [28].

Algorithm 3 AdaBoost

Require: $(\vec{x_1}, y_1), \ldots, (\vec{x_m}, y_m) \in \Omega : \vec{x_i} \in \mathcal{X}$, $y_i \in \{+1, -1\}$
  procedure AdaBoost(Ω,$\mathcal{L}$)
    for $i \leftarrow 1, m$ **do**
      $D_i^1 \leftarrow \frac{1}{m}$ ▷ Initialize to uniform distribution
    end for
    for $t \leftarrow 1, T$ **do**
      $h_t \leftarrow \mathcal{L}(\Omega, D_t)$
      $\epsilon_t \leftarrow \sum_{i=1}^{m} D_t(i) L_{0/1}(h_t(\vec{x_i}), y_i)$ ▷ Weighted zero-one loss of hypothesis h_t
      $\alpha_t \leftarrow \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ ▷ α_t chosen to gain exponential convergence
      $D_i^{t+1} = \frac{D_i^t e^{-\alpha_t y_i h_t(\vec{x_i})}}{\sum_{j=1}^{m} D_j^t e^{-\alpha_t y_j h_t(\vec{x_j})}}$ ▷ Distribution normalized s.t. $\sum_{i=1}^{m} D_i^{t+1} = 1$
    end for
    $H(\vec{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\vec{x}))$
    return $H(\vec{x})$
  end procedure

It can be proved that AdaBoost converges to a hypothesis with zero loss on training data exponentially fast, given that every weak learner performs better than random [28]. Even though this is not always the case in practice [20], the method often works very good. There are also generalization bounds similar to those for SVM and it can be shown that the general boosting method under certain conditions maximize the l_1-margin of the final hypothesis, whereas SVM maximizes the l_2-margin. See [20] for a survey of these issues.

In this thesis we use the very simple *Decision Stump* algorithm, described in the next section, for weak learners as preliminary experiments showed that using more complex learning methods (e.g. *C4.5* or *Backpropagation Neural Networks*) as weak learners tended to lead to overfitting. This approach is also taken in the BoosTexter method by Schapire and Singer [29]. There is however a risk involved in using such simple weak learners, since the convergence of AdaBoost to a perfect fit on training examples is based on the restriction that each hypothesis performs better than random, a restriction that is not always fulfilled by simple learners such as Decision Stumps.

4.4.2 Decision Stump

A Decision Stump is a one level *Decision Tree*, i.e. a rule which splits the feature space in two halves based on the value of a single feature [9]. In a feature space based on $\Re$ this can

be formalized as

$$DS_j^s(\vec{x}) = \begin{cases} +1 & \text{if } (\#_{j,s}^{+1} \geq \#_{j,s}^{-1} \wedge \vec{x}_j \geq s) \vee (\#_{j,s}^{+1} < \#_{j,s}^{-1} \wedge \vec{x}_j < s) \\ -1 & \text{otherwise} \end{cases}, \tag{4.15}$$

where $s \in \Re$ and $\#_{j,s}^c = |\{(\vec{x}, y) \in \Omega : \vec{x}_j \geq s \wedge y = c\}|$, i.e. the number of times class c co-occurs with the value of feature j being greater than s.

The optimal Decision Stump uses the feature which optimally splits the examples. It can be found by simply evaluating all possible Decision Stumps and then choosing the one which best splits the examples in Ω. The goodness of Decision Stump $DS_j^s(\vec{x})$ splitting feature j at point s can be measured for example by *information gain* (reformulated here in a more task-specific form than in (3.2)):

$$IG(\Omega, DS_j^s) = H(\Omega) - \frac{|\Omega_{j \geq s}| H(\Omega_{j \geq s}) + |\Omega_{j<s}| H(\Omega_{j<s})}{|\Omega|}, \tag{4.16}$$

where $\Omega_{j \geq s} \subseteq \Omega$ denotes the subset of examples $(\vec{x}, y) \in \Omega$ such that $\vec{x}_j \geq s$ and $H(\Omega)$ denotes the entropy of the set Ω of examples [21].

In addition to using Decision Stumps as weak learners in the AdaBoost algorithm, we use Decision Stump as a baseline classifier. This is motivated by two reasons. First, since it is based on only one feature in isolation it can be seen as an indicator of how good the individual features produced by the different representations are. Secondly this form of really simple and computationally cheap learning methods have shown to perform on par with more complex methods in many real world applications, at least on average [7].

Chapter 5

Classifier Evaluation

In evaluating a classifier we make a distinction between *effectiveness* which denotes the ability of the classifier to make good classifications with respect to some measure, and *efficiency* which denotes the performance of the classifier in terms of time or memory complexity. The latter is discussed briefly in chapter 4.

In this chapter we discuss some important issues faced when evaluating the effectiveness of the resulting classification system. First we explain why the task of evaluation is problematic, particularly in the context of text classification. We then describe how to best utilize available pre-classified data. Following this is a brief description of different evaluation measures used in evaluating text classification systems. Finally we give an account of further issues in evaluation.

5.1 Difficulties in Evaluation

In section 1.1 we discussed the interrelatedness of the different phases of the text classification task. This is a problem in all machine learning tasks which makes analytical evaluation of the resulting classifier practically impossible and empirical evaluation difficult [31]. Another problem in most practical machine learning applications is a lack of pre-classified data which leads to high variance in the empirical evaluation measures. In the case of text classification the evaluation task is even more difficult since we are trying to automate a subjective practice, defined only ostensibly. The data can be inconsistent since different human classifiers may classify some documents differently.

The subjective nature of TC is evident in the often quite low inter-indexer consistency $IC_{a,b}$ of two human indexers a and b.

$$IC_{a,b} = \frac{N}{A + B + N}, \tag{5.1}$$

where A is the number of unique classes assigned by one indexer, B is the number of unique classes assigned by the other and N the number of unique classes assigned by both indexers in common.

The inter-indexer consistency has been reported to be as low as about 35% – 40% in some domains [36]. Soergel [34] points out that a high consistency is bad if the quality of the classifications are bad and the quality of a system can only be judged relative to some purpose. However, when the purpose is to guide information seekers to a document on some topic, high consistency can be a very important quality since it makes it easier to *learn* where to find what one is looking for.

On a more technical side all machine learning methods presupposes that all the data samples are drawn form the same distribution and that this distribution is a stable underlying distribution. This assumption is probably not completely sound in the case of text classification, which is an ever evolving practice both in terms of the domain and the bias of the indexers. This together with the issues discussed above makes it difficult to define a *gold standard*, i.e. what should be counted as the best possible effectiveness according to some measure.

Next we describe some different schemes for utilizing as much as possible of the available data for evaluation in order to keep variance low and in section 5.3 we describe different evaluation measures.

5.2 Evaluation Schemes

As have been discussed in chapter 4 we are primarily interested in the generalization performance of the classifier. We are only interested in the effectiveness on training data in as much as it defines an upper bound on the effectiveness on unseen data. However, since the learner need as much pre-classified examples as possible to output a good generalizing classifier and since pre-classified examples are scarce, we need some way to optimize the use of these examples. Otherwise the variance of the measured effectiveness can be high.

The most simple way to achieve this is by use of *test-set* testing in which the set Ω of pre-classified examples is divided into the training set Tr, the evaluation set Te and, optionally, the validation set Va. The classifier is then constructed by learning on Tr and the effectiveness is measured by evaluation on Te. If parameters are to be optimized, this can be done by performing a parameter search with effectiveness measured on Va as the objective function. It is common to split Ω into Tr,Va and Te with the ratios $70 : 10 : 20$. This method, of course, does not lead to optimal utilization of the examples in Ω since all the examples in $Va \cup Te$ are "wasted".

A better scheme is *cross validation*, CV, in the form of *k-fold* cross validation or *leave-one-out* cross validation. The latter is a special case of the former in which $k = |\Omega|$. In CV, the effectiveness is taken as the average effectiveness of the classifier h output by the learner $\mathcal{L}$ trained on different *folds*. Each fold corresponds to a different partitioning of Ω, see algorithm 4. The effectiveness of h can be measured by an arbitrary loss function L, see section 4.3. A special case of cross validation is *stratified* CV, in which Ω is split such that the ratios of classes is retained in the training and testing set after the split. This is often preferred in text classification, where the number of documents belonging to different classes often varies substantially over a given data set.

Algorithm 4 k-fold Cross Validation

procedure CROSSVAL(Ω,$\mathcal{L}$,k,L)
  for $i \leftarrow 1, k$ **do**
    $Te_i \leftarrow$PARTITION(Ω,k,i) ▷ Partitions Ω into k non-overlapping subsets
    $Tr_i \leftarrow \Omega \setminus Te_i$
    $h_i \leftarrow \mathcal{L}(Tr_i)$
    $l_i \leftarrow$ EVALUATE(h_i,Te_i,L) ▷ Evaluates hypothesis h_i by L on Te_i
  end for
  $l_{avg} = \frac{1}{k} \sum_{i=1}^{k} l_i$
  return l_{avg}
end procedure

If we want to perform parameter optimization in the cross-validation scheme we can either partition Tr_i into a training set and a validation set as in test-set testing or we can perform a k-fold cross validation on Tr_i for each parameter setting and use this evaluation score as an objective function. Then the best parameter setting is used in the current fold of the outer cross-validation loop.

In theory leave-one-out cross validation should result in a better estimate of the effectiveness than k-fold cross validation. In practice the two methods have shown to perform on par with each other and the computational complexity of k-fold is of course much lower when modest values of k are used. Common choices are $k = 5$ and $k = 10$.

Note that the effectiveness as measured with the techniques above will mostly be slightly pessimistic. This is because when evaluating effectiveness, we do not use all the available data for training. Since the final classifier is trained on all examples in Ω and since more training data never leads to worse performance, it is probable that the final classifier will be somewhat more effective [32].

5.3 Evaluation Measures

A wide range of different evaluation measures have been used in evaluation of text classification systems. Due to time constraints we have not been able to investigate this interesting topic in detail. We therefore discuss only the most commonly used measures. But first a note on notation: In the following TP_i denotes the set of *true positives w.r.t.*[1] c_i, i.e. documents correctly classified as belonging to c_i; FP_i, denotes the set of *false positives w.r.t.* c_i, i.e. documents falsely classified as belonging to c_i; TN_i denotes the set of *true negatives w.r.t.* c_i, i.e. documents correctly classified as not belonging to c_i; and FN_i, denotes the set of *false negatives w.r.t.* c_i, i.e. documents falsely classified as not belonging to c_i.

A measure which may seem very natural for evaluating a classifier and which is sometimes used in the Machine Learning community is *accuracy*, A_i, defined as the probability that a randomly drawn document $d_x \in \mathcal{D}$ belongs to the set $Corr_i = TP_i \cup TN_i$ of correctly classified documents:

$$A_i = \Pr(d_x \in Corr_i). \tag{5.2}$$

This probability is approximated using the following formula:

$$\hat{A}_i = \frac{|Corr_i|}{|Te_i|}. \tag{5.3}$$

In addition to accuracy it's dual *error* defined as $E_i = 1 - A_i$ is sometimes used. The error measure corresponds to the empirical risk making use of the $L_{0/1}$ loss function as described in 4.3.1. Unfortunately these measures do not work well with multi-label classification. Due to the unbalance between $|c_i|$ and $|\bar{c_i}|$ in this case, the trivial classifier which classifies all documents as not belonging to c_i would get a high accuracy score, even though it is really a bad classifier [39].

Precision w.r.t. c_i, denoted π_i, and *Recall w.r.t.* c_i, denoted ρ_i, are two important evaluation measures originally used in the information retrieval community. These in combination form a measure of effectiveness, in the case of information retrieval systems, the ability of the system to retrieve relevant documents while leaving out irrelevant ones [37].

In the context of text classification π_i is defined as the conditional probability that, given that a random document d_x is classified as belonging to c_i, this decision is correct.

$$\pi_i = \Pr(\Phi(d_x, c_i) = T \mid \check{\Phi}(d_x, c_i) = T). \tag{5.4}$$

[1] Here *w.r.t.* is an abbreviation for *with respect to*

Conversely ρ_i is defined as the conditional probability that, given that a random document d_x should be classified as belonging to c_i, this decision is made. [31]

$$\rho_i = \Pr(\check{\Phi}(d_x, c_i) = T \mid \Phi(d_x, c_i) = T). \tag{5.5}$$

In practice these probabilities are estimated using the following formulae:

$$\hat{\pi}_i = \frac{|TP_i|}{|TP_i \cup FP_i|} \text{ and} \tag{5.6}$$

$$\hat{\rho}_i = \frac{|TP_i|}{|TP_i \cup FN_i|}. \tag{5.7}$$

These measures of precision and recall for individual classes can be combined to global measures in different ways. The most used are microaveraging and macroaveraging. In microaveraging each class contributes to the global measure proportional to its share of the positive examples:

$$\hat{\pi}^{\mu} = \frac{\sum_{c_i \in \mathcal{C}} |TP_i|}{\sum_{c_i \in \mathcal{C}} |TP_i \cup FP_i|} \text{ and} \tag{5.8}$$

$$\hat{\rho}^{\mu} = \frac{\sum_{c_i \in \mathcal{C}} |TP_i|}{\sum_{c_i \in \mathcal{C}} |TP_i \cup FN_i|}. \tag{5.9}$$

Whereas in macroaveraging all classes contribute equally:

$$\hat{\pi}^{M} = \frac{\sum_{c_i \in \mathcal{C}} \hat{\pi}_i}{|\mathcal{C}|} \text{ and} \tag{5.10}$$

$$\hat{\rho}^{M} = \frac{\sum_{c_i \in \mathcal{C}} \hat{\rho}_i}{|\mathcal{C}|}. \tag{5.11}$$

These two methods for averaging can give different results on the same classification problem. This is especially the case for non-uniform domains where there is a large difference in *generality*, i.e. how large a part of the document domain that belongs to a class, across classes [31]. When the classifier performs well on classes with low generality, macroaveraging scores can be high even if the classifier performs bad on a large part of the document domain. This suggests using microaveraging since each class of low generality then has less impact on the score and the probability that a randomly drawn document belongs to this class is relatively low. However, if for some reason the classification performance on classes of low generality is especially important, macroaveraging could be the proper choice. It is probably best to use both micro- and macroaveraging to ensure that performance is sufficiently good over the whole document domain.

Precision and recall are combined into a single measure called the F_β-measure, which is really an infinite class of measures:

$$F_\beta(\pi, \rho) = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho} \quad 0 \leq \beta < \infty, \tag{5.12}$$

where β is a parameter controlling the relative importance of precision and recall. When $\beta \to 0$ importance of $\rho \to 0$. When $\beta \to \infty$ importance of $\pi \to 0$. Setting $\beta = 1$ gives the F_1-measure which is the most used measure of effectiveness. This is equivalent to the harmonic mean of precision and recall. Thus it gives equal importance to both:

$$F_1(\pi, \rho) = \frac{2\pi\rho}{\pi + \rho}. \tag{5.13}$$

The F_1-measure is the completely dominant measure of effectiveness in the field of text classification. Sometimes it is used in conjunction with precision and recall as well.

5.4 Further Issues In Evaluation

Yang [39] note that many researchers draw far-reaching conclusions regarding their own methods superiority, even though their experiments are not comparable to others. This is a problem which we have also seen in the literature. It is not uncommon that a method is stated as being superior to another even though they differ by less than a percent.

As a remedy Yang give conditions for two different ways of comparing classifiers. *Direct comparison* means that Φ' and Φ'' can be compared if they are tested on the same set Ω, by the same researcher with the same preprocessing of the data and so on. *Indirect comparison* on the other hand means that the classifiers are tested on Ω' and Ω'', but are compared to one or more baseline classifiers $\Phi_1, \ldots, \Phi_n$ which have been tested on Ω' and Ω'' [39].

However, it should be noted that the results evaluated in these ways may depend on the representations used by the classifiers. Since the motive of this thesis is to compare different representations, rather than producing an optimal classifier, these issues are not of primary concern to us. But the results should of course be taken with a grain of salt.

One fundamental problem with all of the different measures discussed above is that they presuppose a dichotomy between correct and incorrect classification, i.e. correctness is treated as a boolean function. However, in practice it is often the case that some incorrect classifications are more damaging than others. When inter-indexer inconsistency is taken into account a document could both belong to, and not belong to, a certain class.

Furthermore, some misclassifications may be more damaging than others. For example in a hierarchical class structure a misclassification in which a document is classified as belonging to a parent, sibling or child class is probably a much better classification than a misclassification in which a document is classified as belonging to a different branch of the hierarchy. Thus, it should be possible to define measures of effectiveness which take into account some form of measure of distance between classes, e.g. based on topology if the class structure forms a tree or a graph. If no explicit topological structure is defined on the document domain, it could still be possible to define similarities between classes by for example taking the mean of the typical examples of the class. Unfortunately, this similarity measure would depend on the representation used.

A further possibility is that the estimated degree of inter-indexer inconsistency could be used to derive a threshold of acceptable topological distance of the misclassification in a hierarchy of classes, or even in a flat class structure. However, in huge class hierarchies such as the MeSH, see section 6.2, this would not be tractable.

An alternative to quantitative measures of consistency in classifications would be to use some form of *utility* measure, i.e. a measure of how well a certain text classification system serves the need of the users of the system. Of course, this would be an even more subjective measure. It is probably better then to apply this utility measure when deriving the class structure and use quantitative measures of the resulting systems performance.

Chapter 6

Method and Data Sets

In this chapter the data sets or *corpora* and methods used in the thesis are described. We briefly survey some standard corpora and the Medical Subject Headings class hierarchy. An account of the preprocessing of documents along with the weighting schemes used are given. We also describe the different parameters investigated and how parameters of the Support Vector Machine are optimized.

6.1 Corpora

Any collection of documents, e.g. written language or transcribed spoken language, could be said to constitute a *corpus* (lat. *body*). However, in the context of linguistics or other studies of language the following constraints generally apply: the collection should be representative of real world language use in some domain, it should be in machine readable form and it should be intended for scientific use.

In the case of text classification the corpus should be endowed with a set of classes and at least some of the documents should be marked with the classes they belong to. The corpus should be balanced, i.e. the number of documents belonging to a certain class should be proportional to the actual number of documents belonging to the class. Since new texts are constantly produced this will, of course, be a prediction of the future composition of the document domain. As we note in section 5.2, when comparing the results of different classification methods the way the corpus is split into training and test data is also a significant aspect.

There are several standardized corpora available for text classification tasks. The most important are the following:

- RCV1-v2[1] (Reuters Corpus Volume 1 version 2) – a collection of over 800,000 Reuters newswire stories from between August 20, 1996, and August 19, 1997, semi-automatically classified, i.e. the automatic classifications was checked by human classifiers, under three different controlled vocabularies: *Topics*, *Industries*, and *Regions*. The RCV1-v2 collection is intended to replace the Reuters-21578 collection described below [18].

- Reuters-21578[2] – a collection of 21,578 Reuters newswire stories from the year 1987 manually classified under four different controlled vocabularies of named entities: *Ex-*

[1] http://about.reuters.com/researchandstandards/corpus/ (2005-02-26)
[2] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html (2005-02-26)

changes, *Organizations*, *Places* and *People*; and one controlled vocabulary of *Topics*. For information on different versions of the Reuters Corpus see [39].

- TREC-AP[3] – a collection of 209,783 AP newswire stories from the years through 1998 and 1990 classified under a set of 20 classes.
- OHSUMED[4] – a collection of 348,566 abstracts and/or titles of articles published in medical journals in the years 1987 through 1991 manually classified under a subset of the MeSH controlled vocabulary (see below). The documents were collected from a subset of MEDLINE (see below). OHSUMED also contains 101 real world queries intended for testing of information retrieval systems [17].
- 20 Newsgroups[5] – a collection of approximately 20,000 newsgroup documents posted in the year 1993 collected from 20 different newsgroups.

Since we are particularly interested in classification in the domain of medical documents we were originally planning on using the OHSUMED collection as our primary corpora and the 20 Newsgroups collection for comparison. Unfortunately, this was not possible due to the large amount of parameters involved in the evaluation and the huge size of these corpora. Instead we chose to create our own corpus by downloading articles from PUBMED[6], an interface to the MEDLINE database which is a bibliographic database of medical literature maintained by the National Library of Medicine (NLM). In this way we could restrict the corpora to include only texts from a small sub-domain of MeSH, the *Cardiomyopathies* subtree which is a tree consisting of 12 classes structured as follows:

```
Cardiomyopathies (722)
- Cardiomyopathy, Alcoholic (17)
- Cardiomyopathy, Dilated (981)
- Cardiomyopathy, Hypertrophic (527)
- - Cardiomyopathy, Hypertrophic, Familial (40)
- Cardiomyopathy, Restrictive (39)
- Chagas Cardiomyopathy (82)
- Endocardial Fibroelastosis (6)
- Endomyocardial Fibrosis (29)
- Kearns-Sayer Syndrome (29)
- Myocardial Reperfusion Injury (240)
- Myocarditis (388)
```

where the numbers in brackets are the number of documents belonging to each class. See next section for a description of the MeSH vocabulary. Note the highly skewed distribution of the number of documents per class.

The Cardiomyopathies corpus contains 2900 documents, each consisting of the title and abstract of medical articles from the year 2000–2004. The corpus was constructed by posing the following query to PUBMED:

```
"cardiomyopathies"[MeSH Major Topic] AND hasabstract[text] AND
English[Lang] AND medline[sb] AND "humans"[MeSH Terms] AND
("2000/01/01"[PDAT] : "2004/12/31"[PDAT])
```

[3]http://www.daviddlewis.com/resources/testcollections/trecap/ (2005-02-26)
[4]http://www.ltg.ed.ac.uk/disp/resources/ (2005-02-26)
[5]http://people.csail.mit.edu/people/jrennie/20Newsgroups/ (2005-02-26)
[6]http://www.ncbi.nlm.nih.gov/entrez/ (2005-08-23)

We also created a second Cardiomyopathies corpus by posing the same query for the years 1995–1999. This gave us a corpus of 2241 documents. This corpus is used for pre-building the DOR and TCOR representations as described in section 2.2.2.

For reference, we created a corpus based on the 20 Newsgroups corpus by collecting a subset of 150 documents from each of the 20 document classes. This gave us a corpus consisting of 3000 Usenet messages. As with the Cardiomyopathies corpus, we also created a second 20 Newsgroups based corpus of equal size for use in prebuilding of the DOR and TCOR representations. Documents in the 20 Newsgroups corpora is classified under the following 20 document classes each class containing 150 documents per corpus:

```
alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc
```

6.2 Medical Subject Headings

MeSH is a controlled vocabulary produced by the United States National Library of Medicine.[7] Its main purpose is to aid in indexing and searching for biomedical and health-related information. It is a dynamic thesaurus which is revised once a year to capture actual language use in the field. This revision means that new terms are added to the vocabulary while old terms may be removed.

MeSH is structured in a hierarchy with broader terms near the root and more specific terms further down. Different paths through the hierarchy have different lengths, the longest consisting of 11 levels. All terms except for some of the broadest in the top can function as a leaf node, i.e. it can be assigned to a document. Each term resides in at least one position in the hierarchy, but it is quite often the case that a term resides in several different positions. When assigning MeSH-terms to a document the rule is to always use the most specific term. One exception to this rule applies when more than three terms in the same sub-tree are appropriate terms. Then a broader term, i.e. the parent term of the sub-terms, is used.

[7]This section is based on information available from the web site of the United States National Library of Medicine, see *http://www.nlm.nih.gov/mesh/meshhome.html* (2005-08-20).

Each position in the hierarchy is given a unique numerical identifier. This means that the MeSH hierarchy could be described as a *tree*, when nodes are identified with positions or as a *directed acyclic graph*, DAG, when nodes are identified with terms. Mostly the hierarchy is considered as a tree, but the DAG representation could be relevant for exploiting the hierarchical structure for text classification purposes.

The complete MeSH vocabulary consists of several different types of terms, which are used for different purposes. We use only the *descriptors* which are used to characterize the main content of documents. This is due to the fact that our main interest lies in the text-classification methods rather than in the MeSH itself.

As described in the previous section, only the small *Cardiomyopathies* subtree of MeSH is used. The 2005 edition of MeSH contains 22 995 descriptors, while the Cardiomyopathies tree contains only 12 descriptors. Using the complete MeSH would make evaluation of the text classification methods much harder since the amount of pre-classified data needed would be huge and execution times of the algorithms unacceptably long, considering the number of parameters investigated. This restriction means that we are unable to compare our results with other studies in which the complete MeSH-hierarchy is used.

6.3 Preprocessing and Weighting

Before creating document representations we filter all documents through the classical stoplist from the *SMART* information retrieval system, containing 567 common English words.[8] We further remove all documents with fewer than 5 words. The use of this number is not based on any theoretical considerations, but it is commonly used in the literature. The remaining documents are tokenized by removing all non-alphabetical characters and identifying word boundaries with white spaces. After stemming the tokenized words by use of the Porter stemmer we remove all terms that occur in fewer than 3 documents. Again, this number is merely heuristically selected. This preprocessing drastically reduces the number of term occurrences and unique terms in the lexicon, see table 6.1. In the Cardiomyopathies corpus the number of term occurrences are reduced to 323 067 occurrences after removing 264 876 occurrences and the 20 Newsgroups corpus contains 260 068 term occurrences after removing 462 521 occurrences. The language use in the two corpora is thus very different, with the 20 Newsgroups corpus containing much more varied language than the Cardiomyopathies corpus.

Table 6.1. Number of unique terms after preprocessing.

	Cardiomyopathies	20 Newsgroups
After stop word removal	17 445	62 052
After stemming	12 055	51 017
After document frequency threshold	3886	7507

As described in section 2.3, we use the weighting scheme of (2.2)–(2.4). In the BoW representation we simply weight the terms after preprocessing, while in the BoC representations we weight all elements of a term's concept vector by the weight of the term. See section 2.2.3 for a discussion of this heuristic.

When creating the TCOR representation with Random Indexing we use a sliding window as context. The sliding window is 8 terms wide on both sides of the focus term and the

[8] http://terral.lsi.uned.es/∼ircourse/examples/stoplist.html (2005-02-30)

random labels of the surrounding terms are weighted by $2^{(1-d)}$, where d is the distance to the focus term [19]. In creating the DOR representations and the TCOR with documents as contexts, we use simple unweighted term counts when creating the concept vectors.

6.4 Parameters Investigated

There are a vast amount of parameters to investigate. Due to limited time and resources, we are only able to investigate a handful. The parameters which we have investigated have not been chosen in the most structured manner. Rather, we have chosen those which are sufficiently straightforward to investigate with the computational resources, the software and the data sets at our disposal. The following parameters have been examined:

- Representations: BoW, BoC-DOR, BoC-TCOR-DOC, BoC-TCOR-SW
- Dimensionality reduction with BoC: Random Mapping, Feature Selection
- Random Mapping: $k \in \{30, 60, 90, 120, 150\}$
- Random Mapping: $e \in \{\sim k/10, \sim k/3\}$
- Feature selection by Gain Ratio: $\xi \in \{0.01, 0.05, 0.10, 0.15\}$
- Bag-of-concepts with/without pre-built DOR/TCOR representation
- Machine learning algorithms: SVM-LIN, SVM-RBF, AdaBoost, Decision Stump

where DOR denotes the document occurrence representation, TCOR-DOC denotes the term co-occurrence representation with whole documents as context and TCOR-SW denotes the TCOR representation with a sliding window as context. SVM-LIN and SVM-RBF denotes the use of linear and radial basis function kernels with SVM.

The ranges of these parameters where selected based on the results of initial experiments often in conjunction with heuristics appearing in the literature or based on the hypotheses underlying the thesis. Only the value $e = k/3$ is based on a sounder theoretical ground, see section 3.3.2. As described there the value of e is approximated with the smallest even integer $\leq k/10$ and $\leq k/3$, respectively. The values of the reduction factor used for feature selection are based on [41].

We have used Random Indexing to implement the random mapping. Unfortunately, due to restrictions in the Random Indexing implementation used we have only been able to investigate TCOR with sliding window when using RM for dimensionality reduction. When using TCOR with feature selection, document contexts are used instead. Thus the results for BoC-TCOR-SW with RI and BoC-TCOR-FS are not comparable on the basis of different dimensionality reduction techniques. We have not tried any dimensionality reduction methods with BoW, since the ML-methods we use are computationally efficient when data is sparse.

Optimal parameters for SVM are found using 3-fold stratified cross validation on Tr with F_1 as the objective function in a grid search. Ranges used in this search are, with linear kernel: $C \in [2^{-7}, 2^{7}]$ and with radial basis kernel: $C \in [2^{-2}, 2^{10}]$ and $\sigma \in [0.01 \cdot dims^{-1}, 100 \cdot dims^{-1}]$, where $dims$ is the dimensionality of the feature vectors. The parameter search is performed in two steps. First a coarse search on the parameter grid is performed. Then a finer search centered around the optimal value from the coarse search is performed. When using the linear kernel the coarse grid consists of the C-parameter interval divided into 10 steps, with size increasing exponentially. The fine grid consists

of the interval $[c_{opt-1} + 0.05 \cdot step, c_{opt+1} - 0.05 \cdot step]$ divided into 10 steps, where c_{opt} is the optimal step in the coarse search. When using the radial basis function the coarse grid consists of the C-parameter interval divided into 4 steps and the σ-parameter interval divided into 5 steps. The fine grid is constructed analogously with a division into 2 and 5 steps respectively. To speed up the search, if the last two steps have not changed the performance, the step size is temporarily doubled until performance changes. The values of C and σ are based on the values recommended in [8] but are somewhat modified based on initial experiments and computational considerations.

In addition to the above parameters we also tried different variants of *undersampling* and *oversampling* to reduce problems with the non-uniform size of classes. Since these methods did not improve performance, we only mention them here and we do not discuss the effect of different parameter choices related to these methods in the next chapter.

6.5 Software Used

We used an implementation of Random Indexing developed at NADA, KTH by Martin Hassel who also implemented the Porter Stemmer.[9] The Support Vector Machine implementation used was LIBSVM 2.8 by Chang et al.[10] We modified the source code to make it possible to perform a parameter search based on the F_1-measure instead of accuracy. However, the implementation still relies on the $L_{0/1}$ loss function in the optimization problem, F_1 is only used in the meta-search for good parameters. For feature selection, AdaBoost and Decision Stump we used the WEKA 3 machine learning package developed at the University of Waikato.[11]

[9] http://www.nada.kth.se/∼xmartin/java/ (2005-08-20).
[10] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/ (2005-08-20).
[11] http://www.cs.waikato.ac.nz/∼ml/weka/ (2005-08-20).

Chapter 7

Results and Discussion

In this chapter we present the results gained in the evaluation. The chapter is divided into sections based on the different parameters investigated. In each section we present the most interesting results followed by a brief discussion. For completeness we have put plots of the complete evaluation on the Cardiomyopathies corpus in appendix A.

Due to the strong dependence amongst the different parameters investigated, the results are hard to interpret and present. We have tried to separate the different parameters and tried not to draw too farreaching conclusions. All results were measured by the use of 3-fold stratified cross validation, which helps in reducing noise.

7.1 Dimensionality Reduction

7.1.1 Feature Selection

Feature selection by Gain Ratio proves to be a robust dimensionality reduction method, see figure 7.1. Since results for Cardiomyopathies showed no improvement in effectiveness using more than 10% of the features, $\xi = 0.15$ was dropped for the 20 Newsgroups corpus. As is clearly seen, there is no significant improvement on average when using more than 5% of the features. This is in line with the results of Yang et al. [41] who saw no degradation in effectiveness even when using as little as 1-2% of the features with the BoW representation. However, there is a substantial dip in microaveraged effectiveness, when using only 1% of the features with the Cardiomyopathies corpus.

In practice, feature selection is too computationally expensive for the dense BoC representations. This can be greatly alleviated if the feature selection is performed on a small validation set before building representations for the complete document collection. Then it is only necessary to use the selected features in the rest of the classification process. It is however possible that this will lead to a degradation in effectiveness since the feature selection is performed on incomplete data.

7.1.2 Random Mapping

As can bee seen from figure 7.2 and figure 7.3 random mapping performs better with higher values of k, although the difference between $k = 90$, $k = 120$ and $k = 150$ seems to be negligible in general. At the same time the dependence on the value of e is highly unpredictable. Even though 3-fold stratified cross validation is used, the variance across classes and runs is quite high even for the k-parameter, see appendix A.

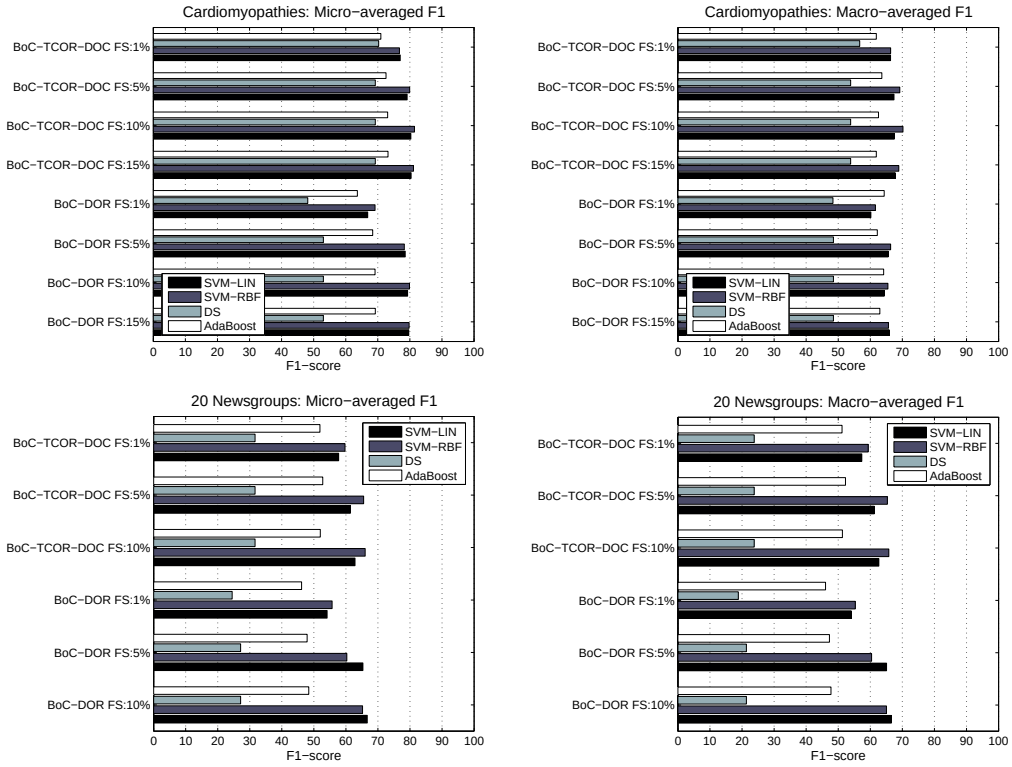


Figure 7.1. Feature selection by Gain Ratio proves to be a robust dimensionality reduction method. Top left: Microaveraged F_1-scores for the Cardiomyopathies corpus using feature selection by Gain Ratio with $\xi \in \{0.01, 0.05, 0.10, 0.15\}$. Top right: Same as top left, but macroaveraged. Bottom left: Same as top left, but for the 20 Newsgroups corpus with $\xi \in \{0.01, 0.05, 0.10\}$. Bottom right: Same as bottom left, but macroaveraged.

It is surprising that the improvement in effectiveness flattens out so quickly with higher dimensionality in random mapping. Sahlgren et al. [26] report poor performance when using a dimensionality less than $k = 1000$ and improved performance with a dimensionality of up to $k = 5000$. One explanation for this is that they use a somewhat larger corpus. Another possible explanation is that they do not seem to optimize any parameters related to the SVM classifier. This is crucial for good performance, see section 7.3. As is discussed in section 4.2.2, when using a higher dimensional feature space the examples are linearly separable with higher probability. Thus when not optimizing the C-parameter, which controls the penalty of misclassifications a higher dimensionality might lead to better performance. Since the noise introduced by RM quickly approaches a normal distribution with variance $1/k$ it seems unlikely that such high dimensionality should be needed. The results for the 20 Newsgroups corpus supports this conclusion. While this corpus contains approximately twice as many terms as the Cardiomyopathies corpus, the number of extra dimensions needed in random mapping is quite small, see figure 7.3. The result for BoC-DOR is even more similar, since the two corpora contain roughly the same number of documents.

We have not been able to evaluate the BoC-TCOR-DOC representation with random mapping. The results for BoC-TCOR-SW which yield concept vectors of the same dimensionality as BoC-TCOR-DOC indicate that the dimensionality needed for BoC-TCOR-DOC will be in the same order.

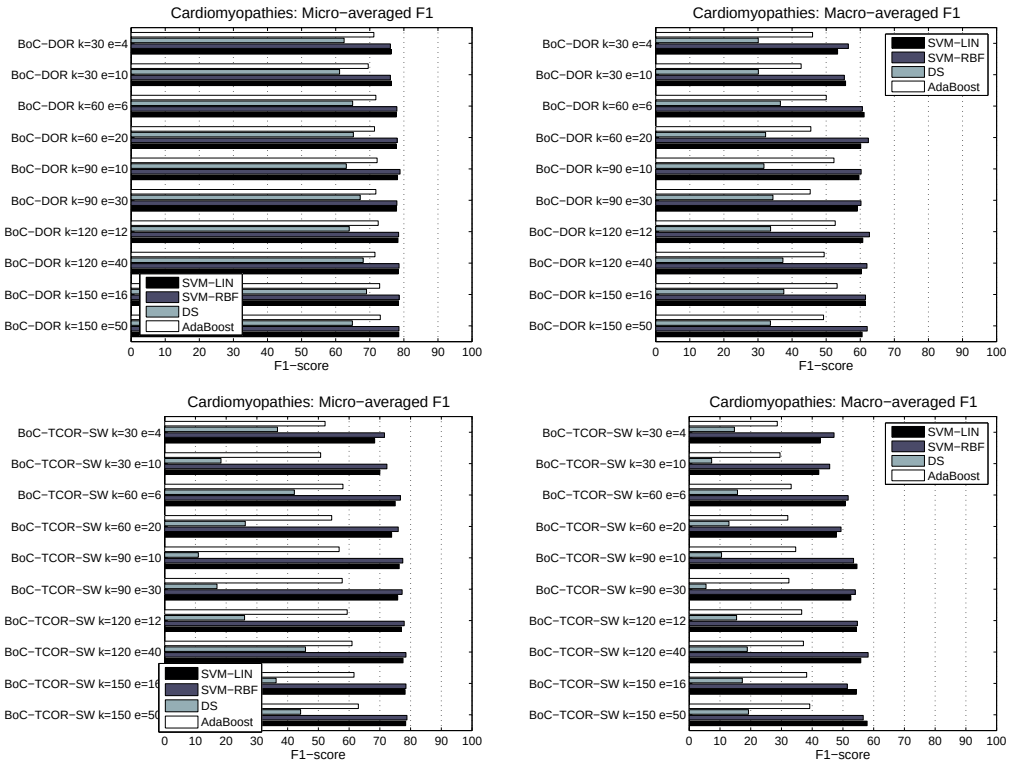


Figure 7.2. Random mapping proves to be robust as well, although at a higher variance. Top left: Microaveraged F_1-scores for the Cardiomyopathies corpus with BoC-DOR using random mapping with $k \in \{30, 60, 90, 120, 150\}$ and $e \in \{\approx k/10, \approx k/3\}$. Top right: Same as top left, but macroaveraged. Bottom left: Same as top left, but with BoC-TCOR-SW. Bottom right: Same as bottom left, but macroaveraged.

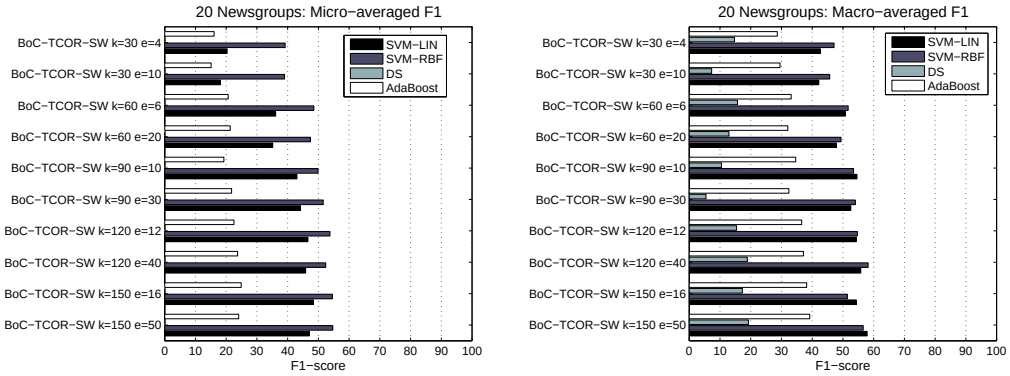


Figure 7.3. Random mapping shows similar behaviour with respect to the k and e parameters for the 20 Newsgroups corpus as for the Cardiomyopathies corpus, even though the original dimensionality of the former is approximately twice as high as the latter when using BoC-TCOR-SW. Left: Microaveraged F_1-scores for the 20 Newsgroups corpus with BoC-TCOR-SW using random mapping with $k \in \{30, 60, 90, 120, 150\}$ and $e \in \{\approx k/10, \approx k/3\}$. Right: Same as left, but macroaveraged.

The following is only speculations, but we believe that the seemingly completely contingent relationship between the value of e and the effectiveness of all the classifiers may prove to be more structured if we study the border region between the different classes. This is especially the case with SVM since the solution of the optimization problem solved by the SVM depend only on the support vectors, i.e. the examples closest to the separating hyperplane. This means that if the number of support vectors is small, then even small changes to the similarity between support vectors can change the solution substantially. If the number of support vectors on the other hand is large, then even a large change to some of the support vectors could give a small change to the solution. Given enough support vectors a small probability for larger amount of noise, i.e. a lower value of e, may be preferable to a lower probability for a smaller amount of noise, i.e. a higher value of e. Due to time constraints we have not been able to investigate this issue further.

7.1.3 Feature Selection versus Random Mapping

Beginning with the Cardiomyopathies corpus, when we compare random mapping with feature selection, the latter seems to give less distinctive individual features in conjunction with BoC-DOR. This is at least the case when using microaveraged F_1. When macroaveraging the opposite seems to be the case. This is however due to the fact that AdaBoost and DS work quite well on some classes on which SVM fails completely. With the other document classes, RM seems to give the most discriminating individual features as indicated by the fact that Decision Stump and AdaBoost perform better with RM than FS with these classes. Given that random mapping is a distributed representation which intuitively should give worse individual features this is an unexpected result.

At the same time RM in conjunction with BoC-DOR fails completely on one class, *Endocardial Fibroelastosis*, while FS give quite good effectiveness, see left of figure 7.4. Effectiveness with feature selection and BoC-DOR also gives substantially better effectiveness on the class *Endomyocardial Fibrosis*, see right of figure 7.4.

Turning to the 20 Newsgroups corpus, the difference between random mapping and feature selection is smaller on average but random mapping actually outperforms feature selection in some cases.

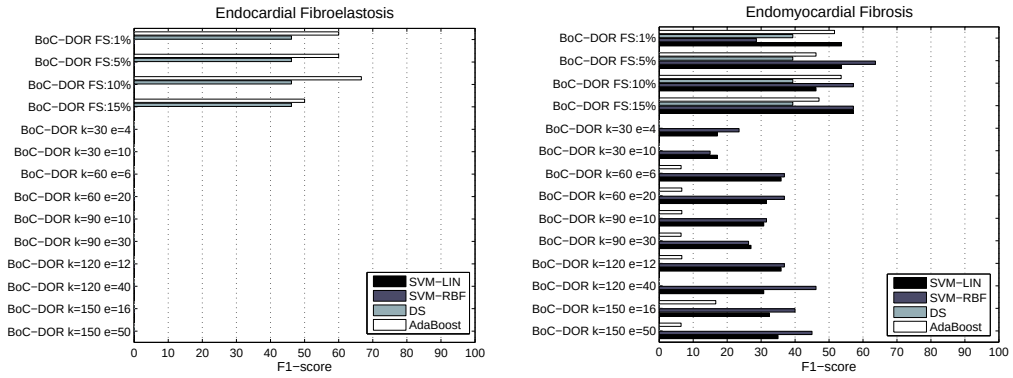


Figure 7.4. Random mapping fails completely or performs poorly in some cases with the Cardiomyopathies corpus. Left: F_1-scores for the *Endocardial Fibroelastosis* class. Right: F_1-scores for the *Endomyocardial Fibrosis* class.

Since feature selection is a completely deterministic dimensionality reduction method, it might be preferable to random mapping when evaluating different representations to reduce

variance. In real-world applications in which a low time complexity is often preferable, random mapping is probably the best choice. Even if we select only 1% of the features, the dimensionality will be in the order of thousands when working with real-world data. Thus if random mapping shows to be robust using only a few hundred dimensions, memory and time complexity will be lower by several orders of magnitude.

7.2 Pre-building

The use of pre-built DOR and TCOR, as described in section 2.2.3, do not improve results on the Cardiomyopathies corpus and it actually leads to worse performance in some cases. The only positive result for this representation is that it seems to improve performance of Decision Stump for some classes. Due to space constraints we do not include any details about the specific results for this corpus.

On the other hand, pre-building seems to give some improvement in effectiveness on the 20 Newsgroups corpus when using feature selection, see top of figure 7.5. When using random mapping pre-building is highly detrimental, see bottom of figure 7.5. Overall pre-building thus seems to be a bad idea, and the small improvement when using feature selection may actually depend on the fact that we select a certain *fraction* of the features. Hence, since the number of features is larger when using pre-building, the selected number of features will also be larger.

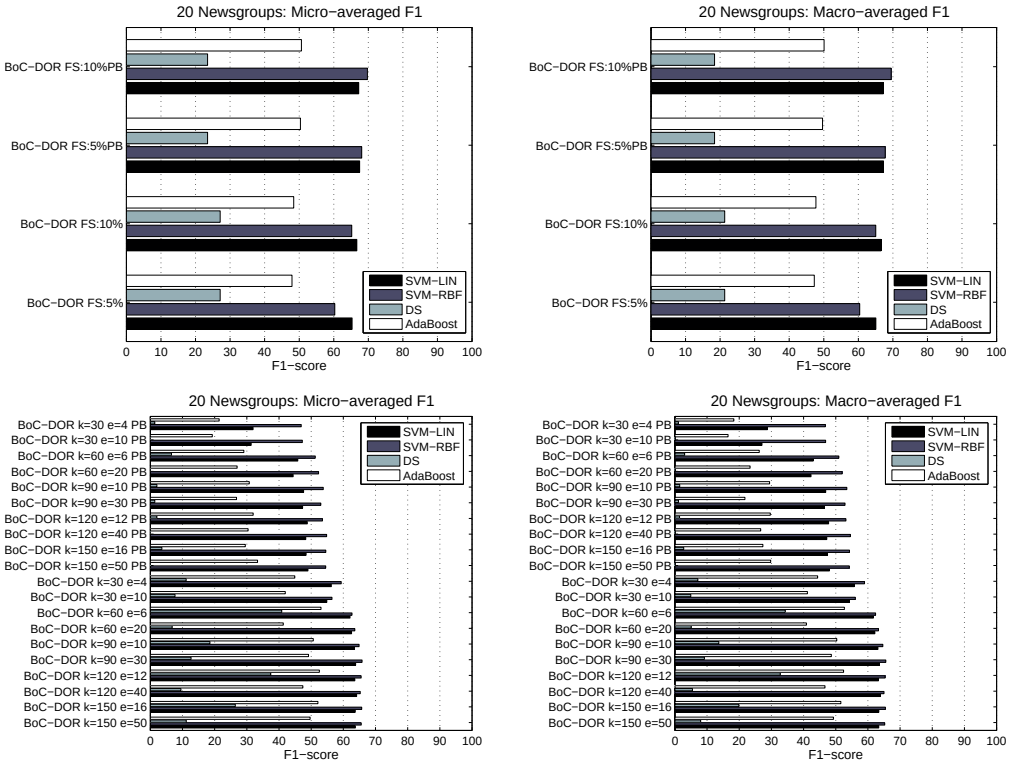


Figure 7.5. Top left: Microaveraged F_1-scores for the 20 Newsgroups corpus with BoC-DOR and feature selection with and without pre-building. Top right: Same as top left but macroaveraged. Bottom left: same as top left but with random mapping instead of feature selection. Bottom right: same as bottom left but macroaveraged.

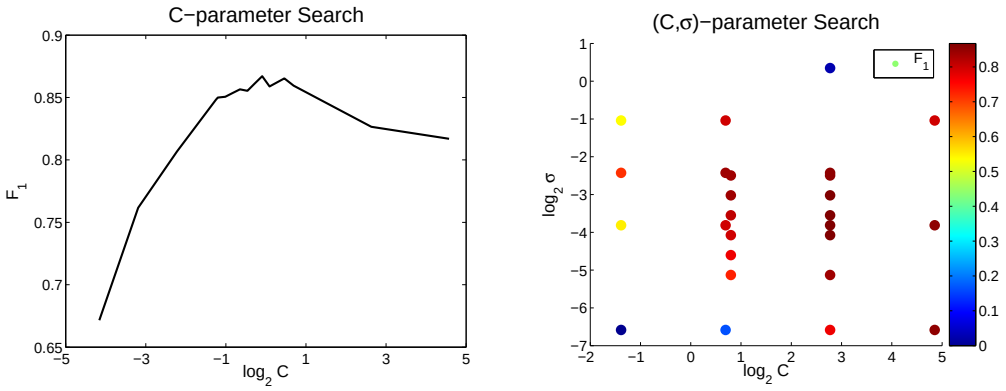


Figure 7.6. SVM is quite sensitive to parameter choice. Left: F_1-score as a function of the C parameter with linear kernel. Right: F_1-score (coloured) as a function of the C and σ parameters with radial basis function kernel.

Recalling the underlying hypothesis that bag-of-concepts representations better capture the semantics of documents and that the DOR and TCOR representations actually do capture the semantics of terms, one could expect that pre-building should be a good idea. The question then is whether the flaw lies in the hypotheses or somewhere else. One possible explanation for the method's poor performance is that too much noise is introduced when occurence and co-occurence statistics are collected on documents that are not used to train the classifier. This may be the case if these documents contain a high amount of new terms which are not present in the documents used in classification. However, since these documents were pre-processed the same way as the documents used in the classification, most of the noise should be filtered out already.

7.3 Machine Learning Algorithms

As expected SVM outperforms both AdaBoost and DS on almost all document classes. The only exceptions are *Endocardial Fibroelastosis* and *Kearns-Sayer Syndrome* from the Cardiomyopathies corpus. These are both really small document classes containing only 6 documents each and both seems to be highly keyword-dependent.

The difference in performance between the different machine learning methods is smallest in conjunction with BoW, where AdaBoost performs almost on par with SVM, with DS performing slightly worse than AdaBoost on average. When using the BoW representation there is only one class, *Cardiomyopathies* on which DS and AdaBoost fail completely while SVM show good performance.

As shown in figure 7.6 it is crucial to optimize the C and σ parameters when using SVM with linear and radial basis function kernels. As noted in section 6.4, we have not optimized the number of rounds used in AdaBoost. AdaBoost is however known to be quite stable with respect to this parameter and improvement usually drops quite fast with increased number of rounds.

There is no significant difference between the performance of SVM-LIN and SVM-RBF on average. This is especially the case with the Cardiomyopathies corpus. With 20 Newsgroups it seems like the linear kernel is better suited for use with BoC-DOR, while the RBF-kernel is better suited for use with BoC-TCOR-DOC when using feature selection. When using random mapping there is no significant difference between the two kernels.

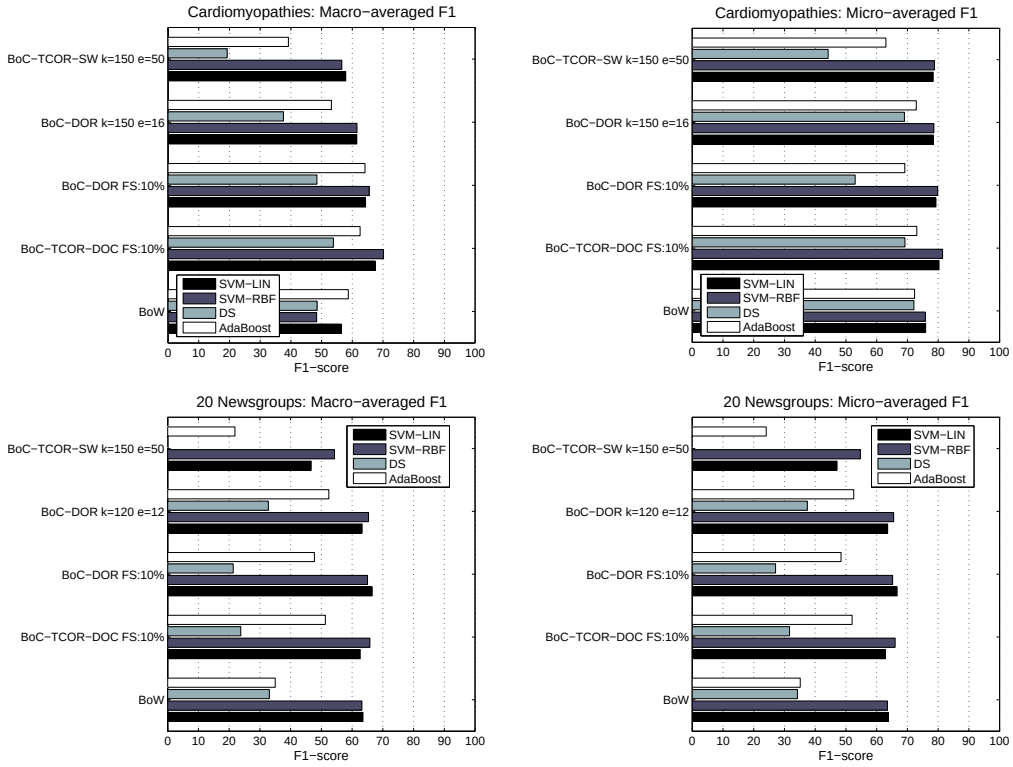


Figure 7.7. On average BoC improves effectiveness as compared to BoW. Top left: Best microaveraged F_1-score on the Cardiomyopathies corpus for each representation. Top right: Same as top left, but macroaveraged. Bottom left: Same as top left, but on the 20 Newsgroups corpus. Bottom right: Same as bottom left, but macroaveraged.

Given that the linear kernel has only one parameter to optimize and that each kernel evaluation has lower computational complexity, it is probably the best choice in practical applications.

7.4 Representations

Microaveraged results on the Cardiomyopathies corpus indicate that there is a small increase in effectiveness when using BoC representations as compared to BoW together with SVM, see left of figure 7.7. When using AdaBoost and DS, BoW outperforms BoC in all cases except for BoC-TCOR-DOC. The difference is however quite small and may be attributable to the fact that the parameter optimization of SVM is still quite coarse. Macroaveraged results on the Cardiomyopathies corpus on the other hand show a substantial increase in effectiveness when using BoC-DOR and especially BoC-TCOR-DOC, see top right of figure 7.7. When using feature selection both these representations outperform BoW in conjunction with all the machine-learning methods. BoC-TCOR-SW does not improve effectiveness at all as compared to BoW when using macroaveraging.

The BoW representation outperforms all the BoC representations on only one class, *Kearns-Sayer Syndrome*, and this is only when using Decision Stump or AdaBoost, see figure 7.8. The equal performance of DS and AdaBoost suggests that the best classification

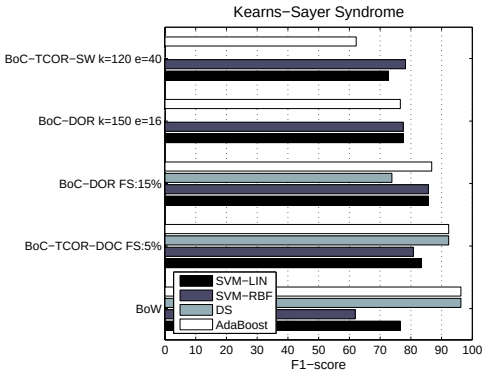


Figure 7.8. Sometimes it suffices to use only one feature for good classification performance. This figure shows F_1-scores on the *Kearns-Sayer Syndrome* document class. Results is only given for the best parameter choice of each representation.

in this case is based on only one keyword.

There are however some document classes on which the BoW representation fails radically while BoC works quite well, see figure 7.9. The classes are "*Endomyocardial Fibrosis*", "*Cardiomyopathy, Restrictive*", "*Cardiomyopathy, Hypertrophic, Familial*" and "*Cardiomyopathy, Alcoholic*". These are all classes containing few documents and the performance of BoC is not as good as on the larger classes, even though it is significantly better than BoW. While the noted *Kearns-Sayer Syndrome* contains very few documents, it seems clear that document classes containing few documents benefit most from the BoC representations with the exception for strongly keyword dependent classes.

This is interesting since Sahlgren and Cöster [26] report the opposite result, i.e. that BoC representations works best with the largest document classes. They use the Reuters-21578 corpus described in section 6.1, which may explain this difference. However, since they do not seem to have optimized any of the parameters related to SVM, their results might be misleading. Further, the improvement of effectiveness in their results is only in the order of a few percents, while we see a substantial improvement. Here it should be noted that there is a large difference in resolution of the measurements of performance on small classes compared to large classes.

Except for the small classes discussed above there is a smaller, but significant, improvement in effectiveness on the class *Cardiomyopathies* which contains 722 documents.

Comparing the different BoC representations also yields interesting results. For the large document classes there is no significant difference in effectiveness between the different representations. This is at least the case when using SVM, which gives the best performance on all these classes. The effectiveness of DS and AdaBoost is more varied, with performance slightly worse than SVM when used with TCOR-DOC. When used with DOR, their performance is somewhat worse than with TCOR-DOC. And when used with TCOR-SW, DS and AdaBoost perform poorly. This suggests that TCOR-SW yields more distributed representations than DOR and TCOR-DOC and that TCOR-DOC yields the best individual features. The reason for this behaviour is unclear. One reason might be that in the TCOR-DOC representation a word always co-occurs with itself, thus capturing some of the keyword-dependent aspects of the BoW representation. This is not the case with the random indexing implementation used with TCOR-SW.

Results on the small classes on which the BoW representation fails is more interesting, again see figure 7.9. Here TCOR-SW gives somewhat better results than BoW, while

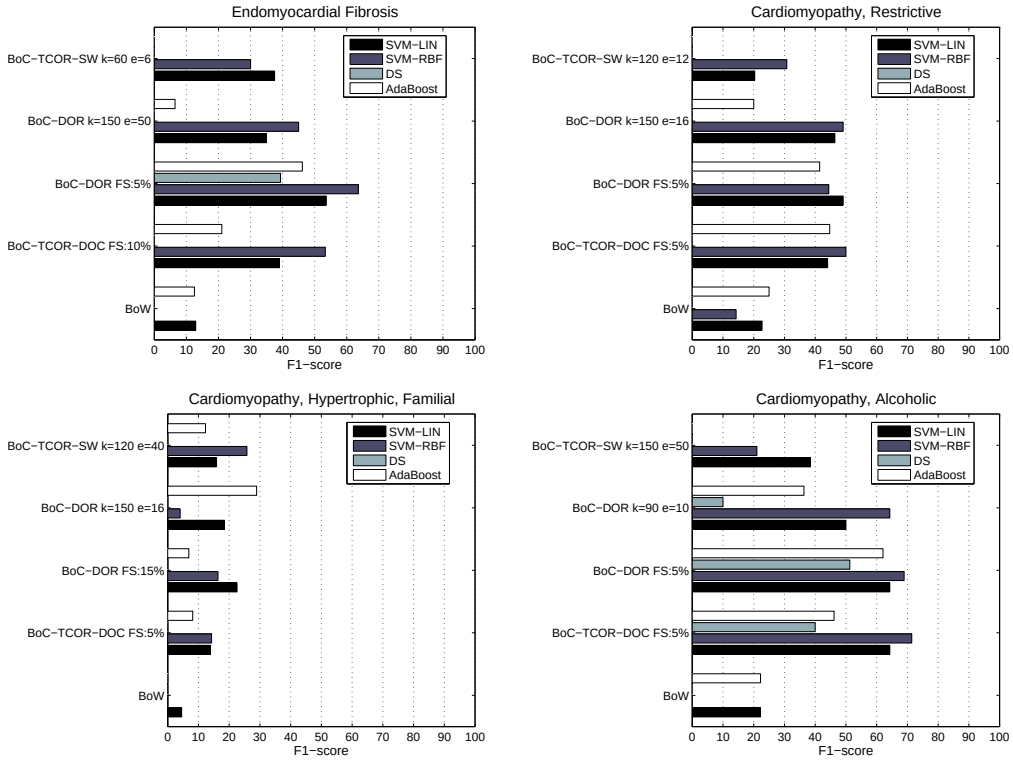


Figure 7.9. While the BoW representation fails completely on some of the small document classes, BoC gives acceptable and in some cases good effectiveness. Top left: F_1-scores on the *Endomyocardial Fibrosis* document class. Top right: F_1-scores on the *Cardiomyopathy, Restrictive* document class. Bottom left: F_1-scores on the *Cardiomyopathy, Hypertrophic, Familial* document class. Bottom right: F_1-scores on the *Cardiomyopathy, Alcoholic* document class.

TCOR-DOC and DOR show a large increase in effectiveness. There is a slight performance difference in favour of DOR, but since the resolution of the F_1 measure on these small classes is quite poor, this difference might just be accidental.

Results on the 20 Newsgroups partly confirm the appropriateness of BoC representations, found on the Cardiomyopathies corpus. On the 20 Newsgroups corpus, however, the improvement is much smaller for DOR and TCOR-DOC, while the TCOR-SW actually leads to worse performance, see bottom of figure 7.7. Interestingly, performance on the 20 Newsgroups is much more stable over classes as can be seen from the similarity between microaveraged and macroaveraged results. This is particularly the case for SVM, while DS and AdaBoost show some variation.

Chapter 8

Conclusions

In this chapter we give a brief summary of the insights reached in the thesis. We present the, in our view, most important positive and negative results. Finally we give some suggestions for future research.

8.1 Main Results

The most important positive result is in our opinion that BoC give a huge improvement in effectiveness on classes on which BoW fails. Another promising result is that random mapping is robust down to about 100 dimensions in conjunction with SVM, although one can expect that a somewhat higher dimensionality will be needed in real-world applications.

It is also positive that DOR and TCOR-DOC in general work better than TCOR-SW. This is in line with the intuition of Sahlgren [25] who states that a broader context captures what a word *is about* rather than what it *means*. Unfortunately we have not been able to find any significant difference in performance between DOR and TCOR-DOC in general, which was anticipated given the results of Lavelli et al. [16].

The biggest disappointment is the fact that pre-building the term meaning representations does not improve performance. However, we would not rule out the method altogether without pursuing some further research. It is our belief that if pre-building is shown never to be a good idea, then there is something wrong with the underlying hypothesis that DOR and TCOR captures the *meaning* of words, or with the hypothesis that linear combinations of word's meaning captures the semantics of documents.

The second biggest disappointment is that we are unable to give any clues on how the parameter e in random mapping affects performance.

8.2 Implementational and Practical Issues

When designing a text classification system for use in real world domains, we suggest that BoW is used as a primary representation and that BoC is used as a complement on document classes on which BoW fails. It is, however, not clear how to find these classes in advance. If there is a sufficiently large amount of pre-classified data, this could be used to evaluate the different representations. But it would of course be preferable if there were some more analytical methods available. It might also be possible to make use of machine-learning methods to predict the best representation for a given document class.

We believe that random mapping is the only practically useful dimensionality reduction method on real world text classification problems, since feature selection by measures such as

information gain or gain ratio is too computationally expensive. Although feature selection works better in some cases, random mapping is sufficiently robust especially in conjunction with SVM.

When it comes to machine learning methods, the support vector machine is in our opinion preferable. In theory the RBF-kernel should be better or as good as the linear kernel, given sufficient parameter optimization. However, when taking computational considerations into account, the linear kernel is probably the best choice. The SVM implementation used in this thesis works well, despite that it uses the $L_{0/1}$ loss function discussed in section 4.3. However, we recommend that an implementation, which uses the F_β function directly in the optimization problem, is used instead. See the paper by Joachims [11] for further information.

8.3 Suggestions for Future Research

We observe three fundamental questions related to the use of bag-of-concepts representations in text classification to be studied more thoroughly by future researchers:

- How does the choice of context affect the effectiveness of the resulting system?
- How should concept vectors be combined to document vectors to best capture the semantics useful for classification?
- Are there other models of word's meanings which better mimic the semantics used by humans as compared to the simple frequency count of the DOR and TCOR models?

All three questions are related to the question of how to best model the contextual aspects of meaning and they will probably have different answers depending on the document domain and the classification task at hand.

Specifically related to the medical domain is the question of how to utilize the vast knowledge accumulated in different ontologies and knowledge bases such as the Unified Medical Language System, UMLS. This knowledge might prove very useful in the construction of text-classification systems, if we could somehow combine it with the representations studied in this thesis.

A related question is whether a combination of BoW and BoC and possibly other representations may give better results than the best individual representation. There are several possible machine learning schemes for combining representations, e.g.*voting/ensemble* methods and *mixture of experts* methods.

Another important issue is the usefulness of the different representations with other languages than English. It is possible that some representations are especially suited for Swedish which has more combined forms than English. In the context of clustering and information retrieval, for example, stemming and lemmatization have been shown to improve performance more with Swedish than with English.

A more technical question related to SVM is how to best construct a hierarchical classifier. To reduce a flat hierarchy to binary classification is extremely computationally demanding when used in conjunction with large class structures and large document domains, such as the MeSH. One possibility is of course to use a simple divide and conquer approach. This will however not solve the problem for larger flat regions of the hierarchy. There exist solutions to this problem in the case of single-label classification, but to our knowledge not in the case of multi-label classification.

In hierarchical domains it would also be very interesting to study the graveness of misclassifications which different representations and algorithms lead to, as discussed in section 5.4.

Bibliography

[1] Dimitris Achlioptas. Database-friendly random projections. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, New York, NY, USA, 2001. ACM Press.

[2] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[3] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[4] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 78–87, New York, NY, USA, 2004. ACM Press.

[5] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[6] Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Scott C. Deerwester. Using latent semantic analysis to improve information retrieval. In *Proceedings of CHI'88: Conference on Human Factors in Computing*, pages 281–285, New York, USA, 1988. ACM.

[7] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.

[8] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification.

[9] Wayne Iba and Pat Langley. Induction of one-level decision trees. In *Proceedings of the ninth international workshop on Machine learning*, pages 233–240, Aberdeen,UK, July 1992.

[10] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[11] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML), 2005*, 2005.

[12] P. Kanerva, J. Kristoferson, and A. Holst. Random indexing of text samples for latent semantic analysis. In L.R Gleitman and A.K Josh, editors, *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036, Mahwah, USA, 2000. Erlbaum.

[13] Jussi Karlgren and Magnus Sahlgren. From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 294–308. CSLI Publications, Stanford, USA, 2001.

[14] Samuel Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of IJCNN'98, 1998 IEEE International Joint Conference on Neural Networks*, pages 413–418, Piscataway, USA, 1998. IEEE Service Center.

[15] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1/2):273–324, 1997.

[16] Alberto Lavelli, Fabrizio Sebastiani, and Roberto Zanoli. Distributional term representations: an experimental comparison. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 615–624, New York, NY, USA, 2004. ACM Press.

[17] William Hersh; Chris Buckley; T. J. Leone and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201, Dublin, Ireland, July 1994. ACM.

[18] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, April 2004.

[19] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28(2):203–208, 1996.

[20] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In *Advanced lectures on machine learning*, pages 118–183. Springer-Verlag New York, Inc., New York, USA, 2003.

[21] Tom Mitchell. *Machine Learning - International Edition*. McGraw-Hill, 1997.

[22] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Symposium on the Principles of Database Systems*, pages 159–168, Seattle, USA, 1998.

[23] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[24] Ross Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, Menlo Park, August 1996. AAAI Press.

[25] Magnus Sahlgren. Vector-based semantic analysis: Representing word meanings based on random labels. In *Semantic Knowledge Acquisition and Categorisation Workshop, ESSLLI 2001*, Helsinki, Finland, 2001.

[26] Magnus Sahlgren and Rickard Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004*, pages 487–493, Geneva, Switzerland, August 2004.

[27] Magnus Sahlgren and Jussi Karlgren. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering*, 11(3), 2005.

[28] Robert E. Schapire. Theoretical views of boosting. *Lecture Notes in Computer Science*, 1572:1–10, 1999.

[29] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based systemfor text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

[30] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

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

[32] Fabrizio Sebastiani. Text categorization. In Alessandro Zanasi, editor, *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*. WIT Press, Southampton, UK, 2005. Forthcoming.

[33] Fabrizio Sebastiani and Franca Debole. Supervised term weighting for automated text categorization. In *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing*, pages 784–788, Melbourne, US, 2003.

[34] Dagobert Soergel. Indexing and retrieval performance: The logical evidence. *Journal of American Society for Information Science*, 1994.

[35] Dawei Song and Peter D. Bruza. Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology*, 54(4):321–334, 2003.

[36] Victoria Uren. An evaluation of text categorisation errors. In *Proceedings of the One-day Workshop on Evaluation of Information Management Systems*, pages 79–87, London, UK, September 2000. Queen Mary and Westfield College.

[37] C.J. van Rijsbergen. *Information Retrieval, 2nd edition*, chapter Evaluation. Butterworths, London, UK, 1979.

[38] Vladimir Vapnik. *Statistical Learning Theory*. Springer-Verlag, Ney York, USA, 1998.

[39] Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):69–90, 1999.

[40] Yiming Yang and Christoffer G. Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 92)*, pages 447–453, 1992.

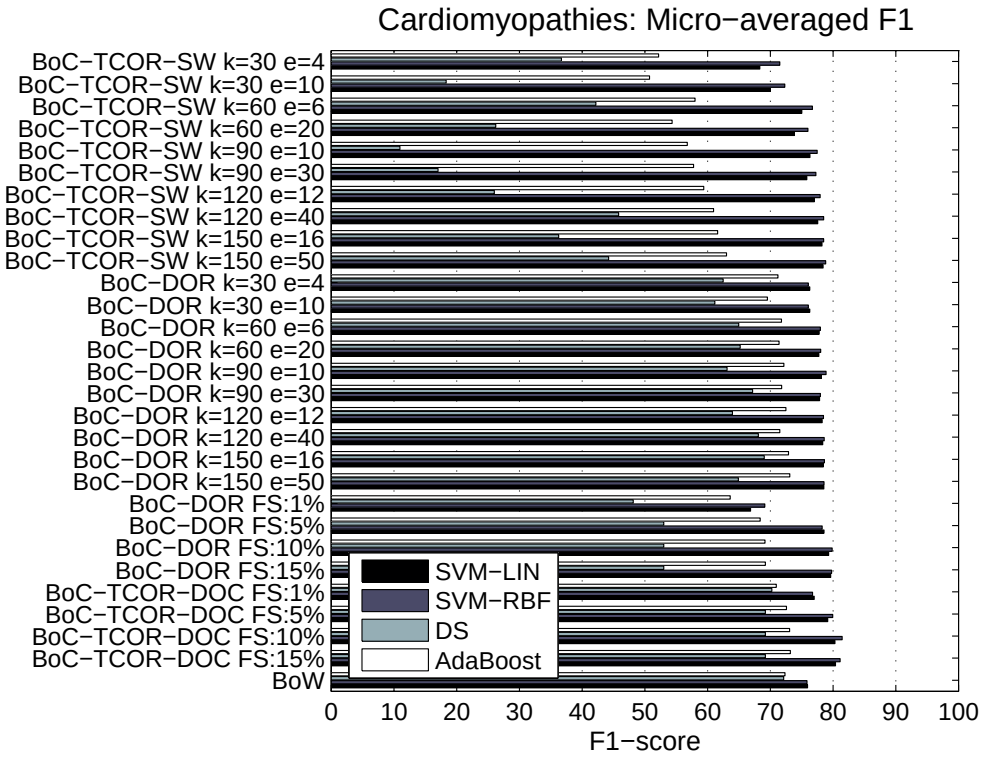
[41] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

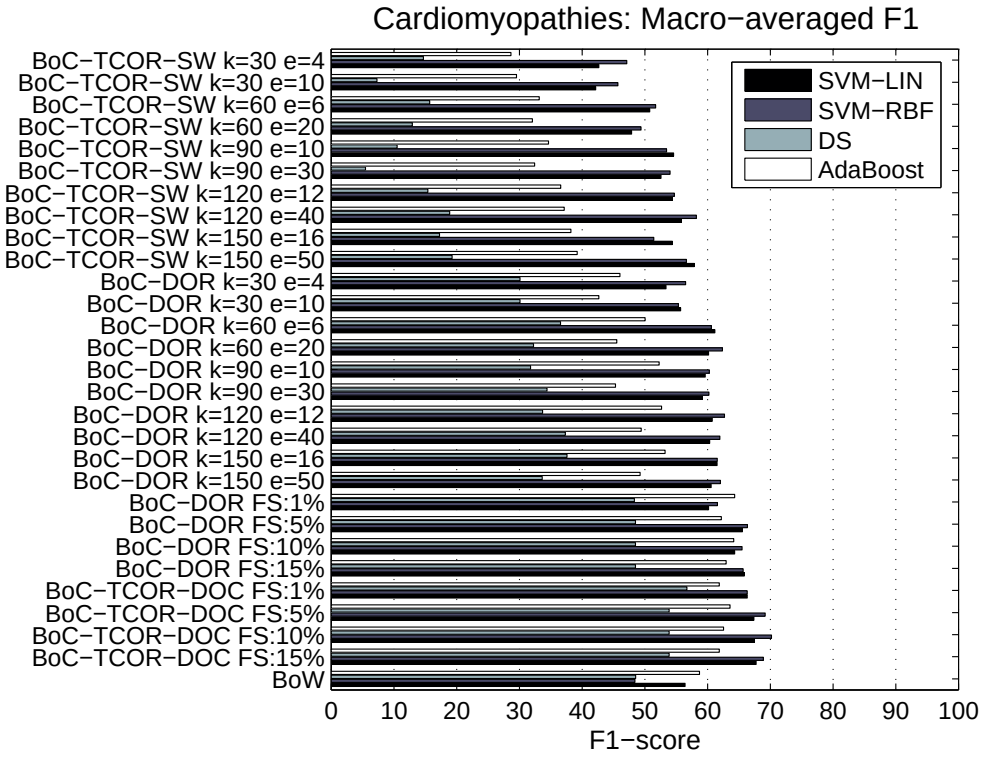
[42] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, New York, USA, July-August 2003. ACM Press.

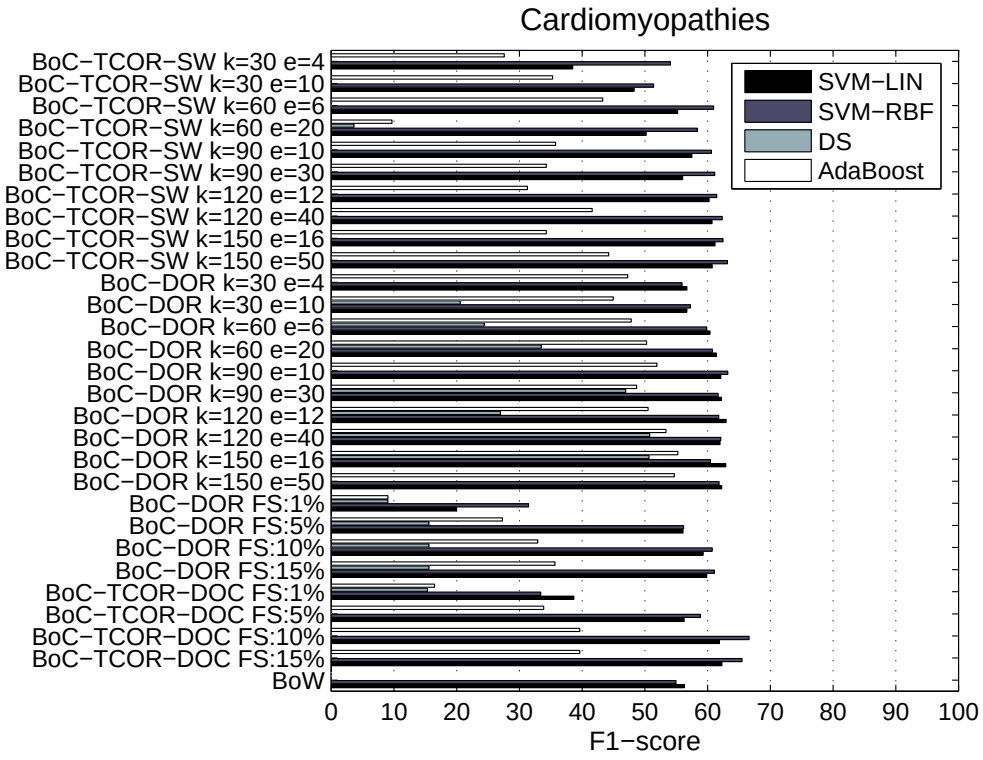
[43] Justin Zobel and Alistair Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.

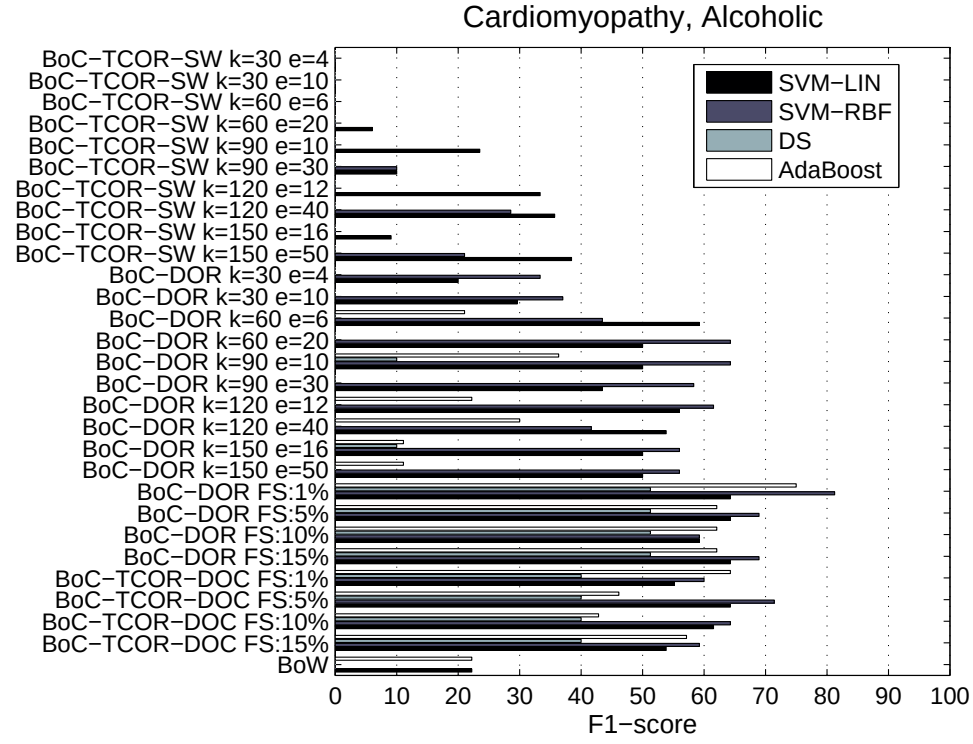
Appendix A

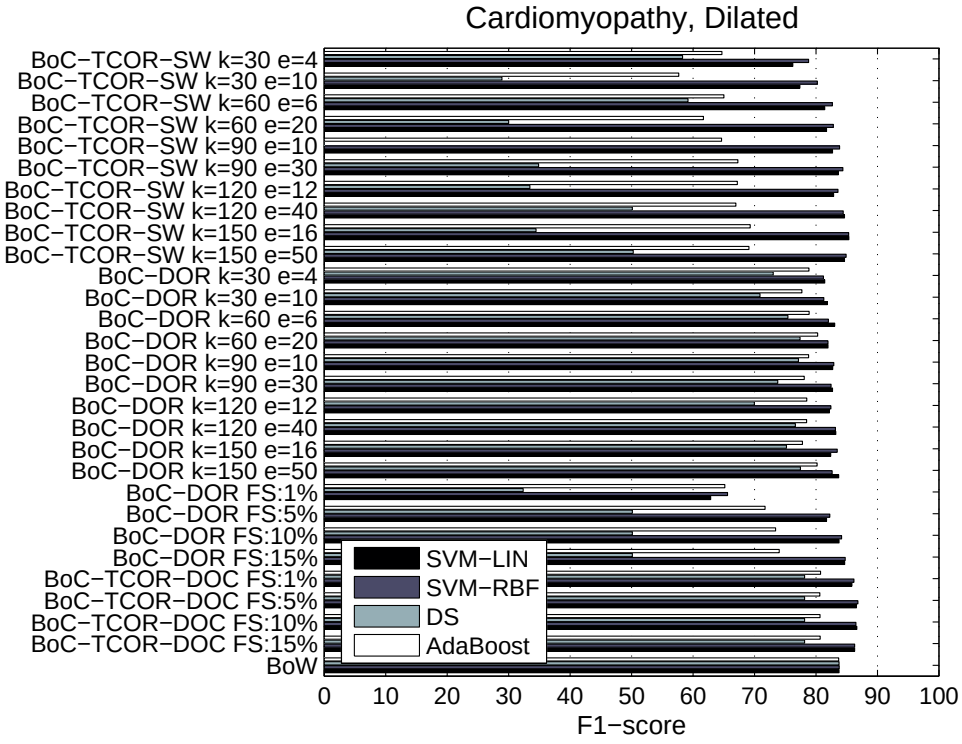
Cardiomyopathies Results

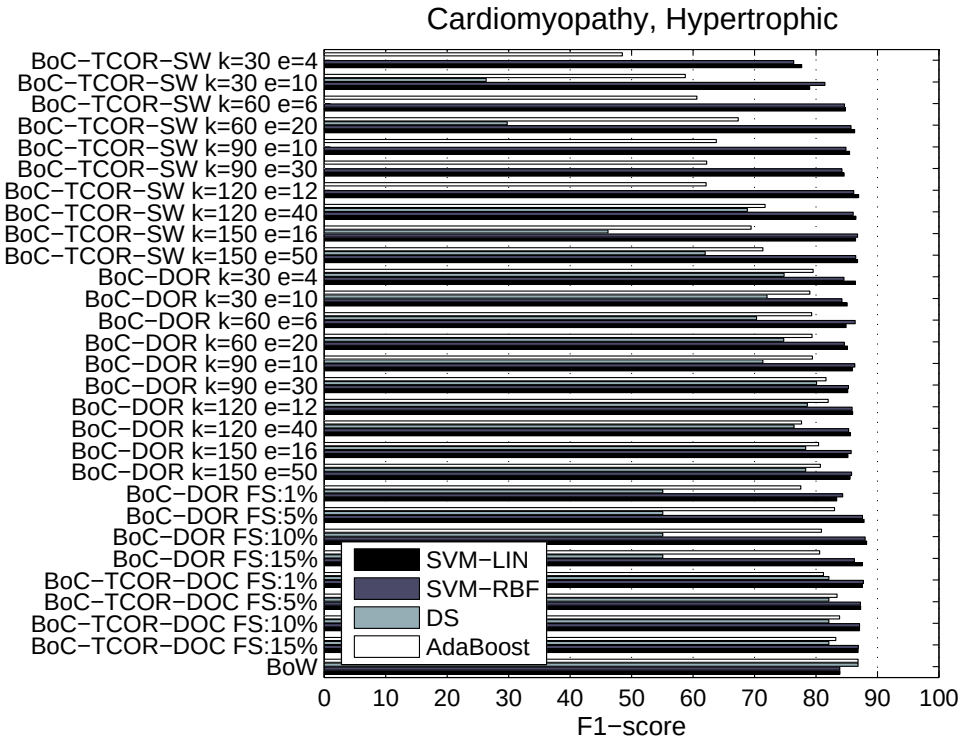


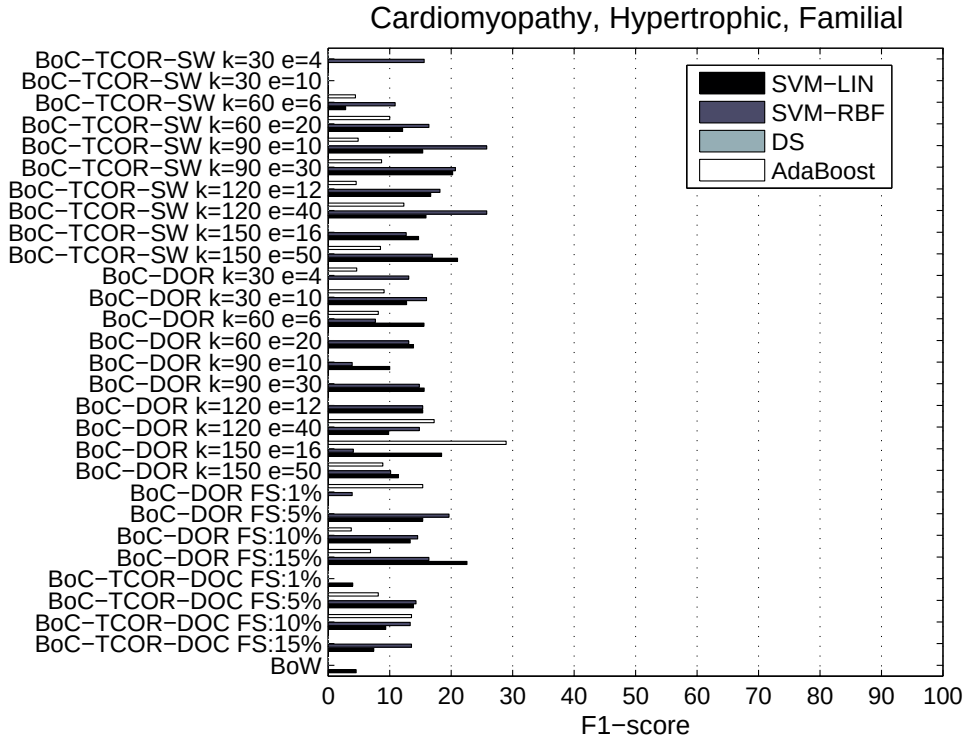

Cardiomyopathies: Macro−averaged F1
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

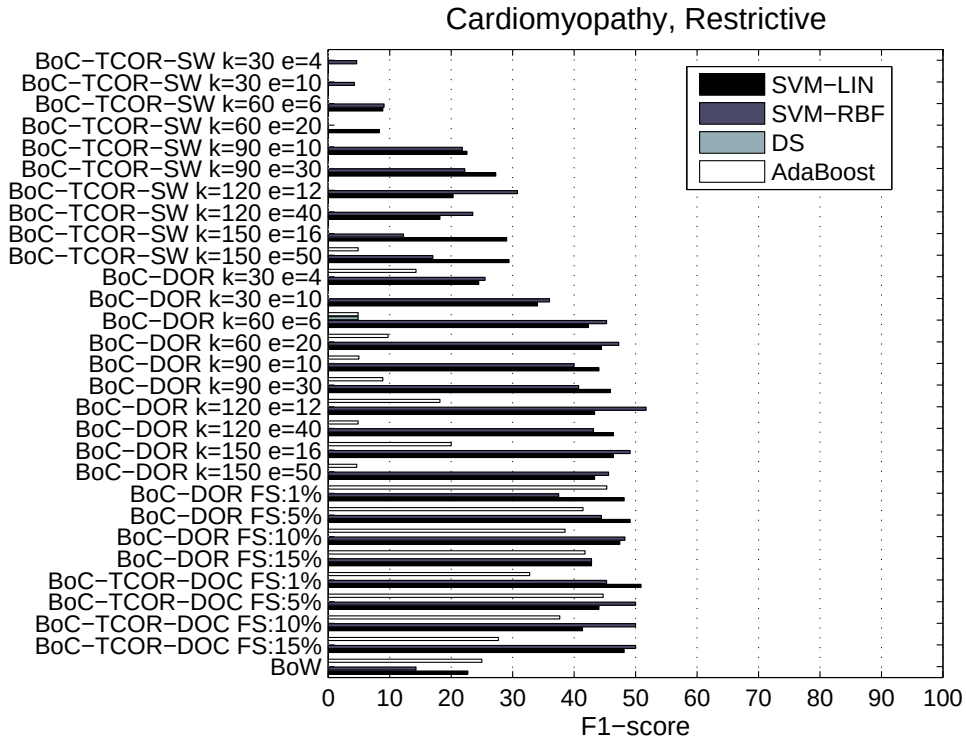

Cardiomyopathies
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

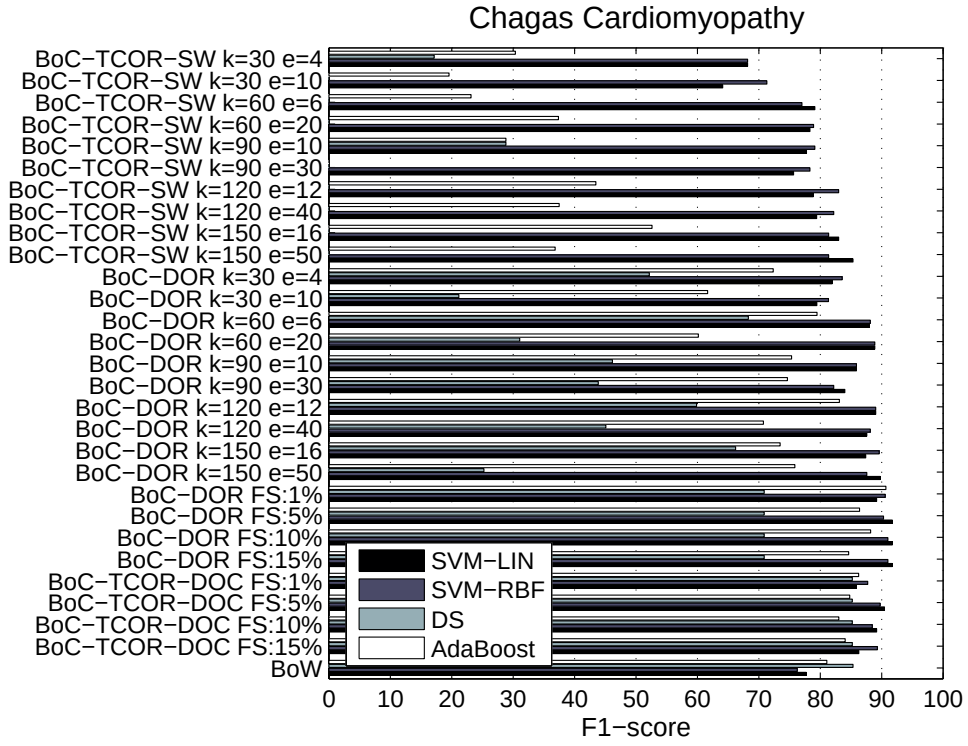

Cardiomyopathy, Alcoholic
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

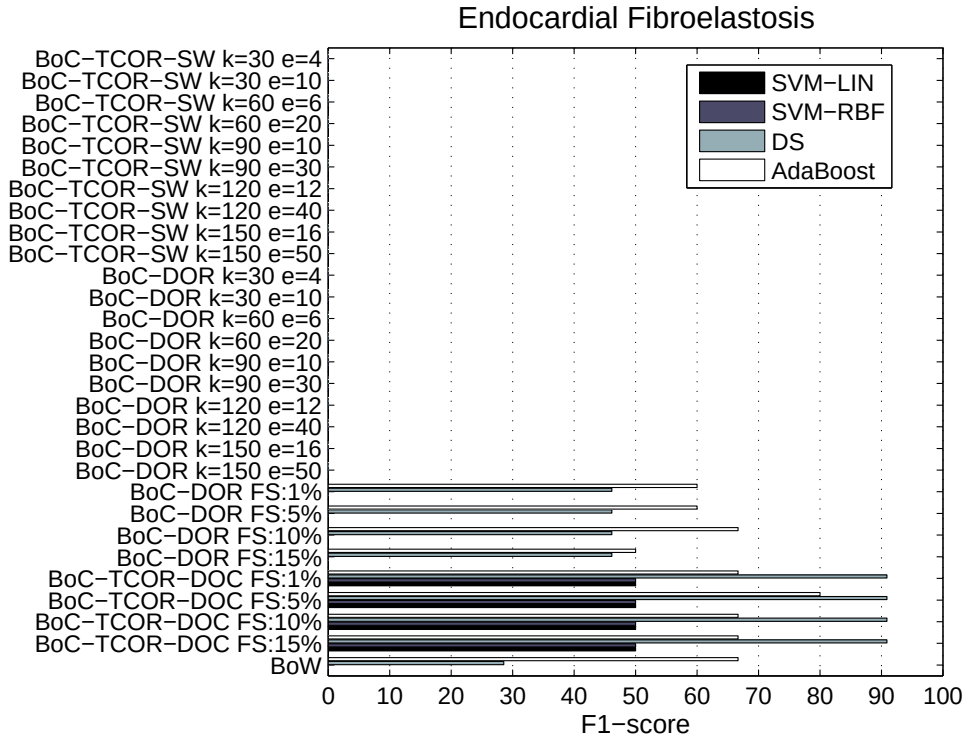

Cardiomyopathy, Dilated
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

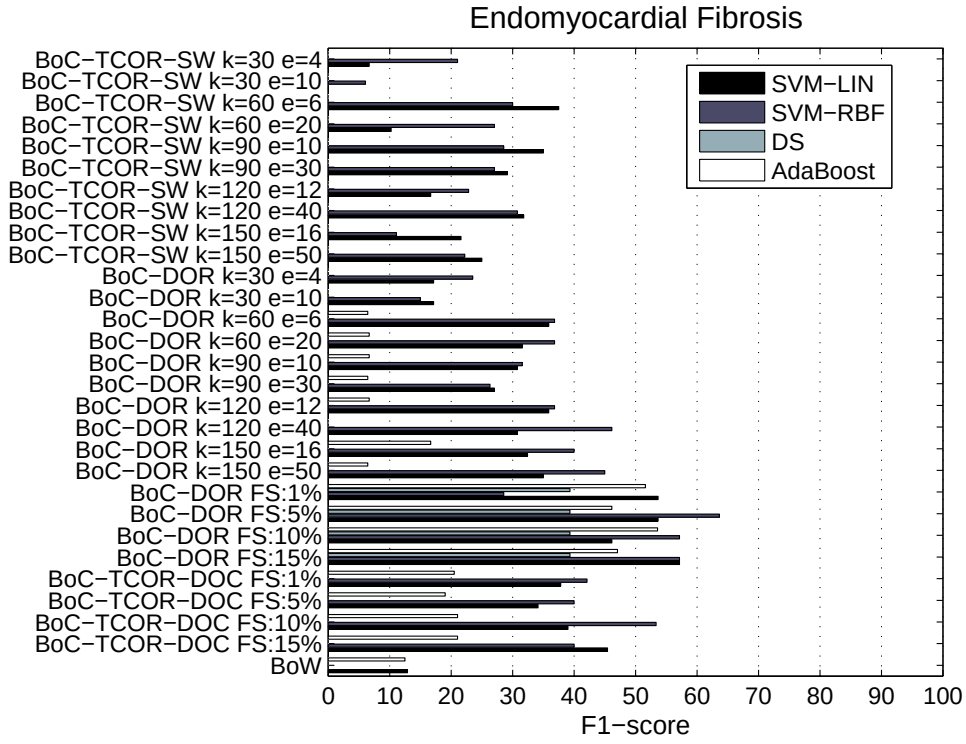

Cardiomyopathy, Hypertrophic
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

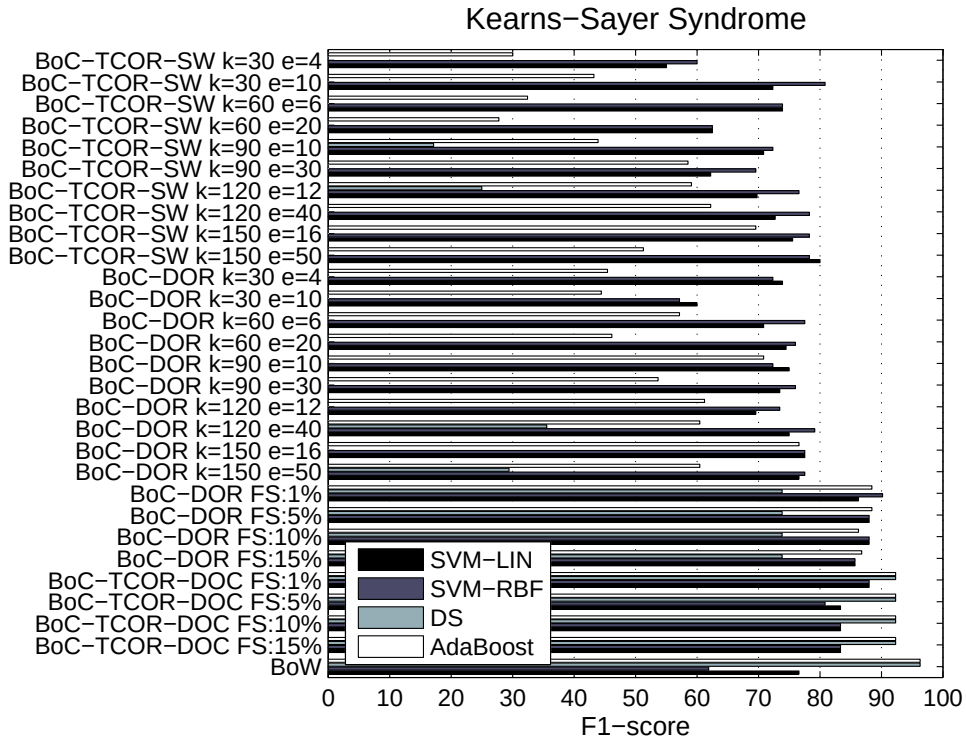

Cardiomyopathy, Hypertrophic, Familial
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

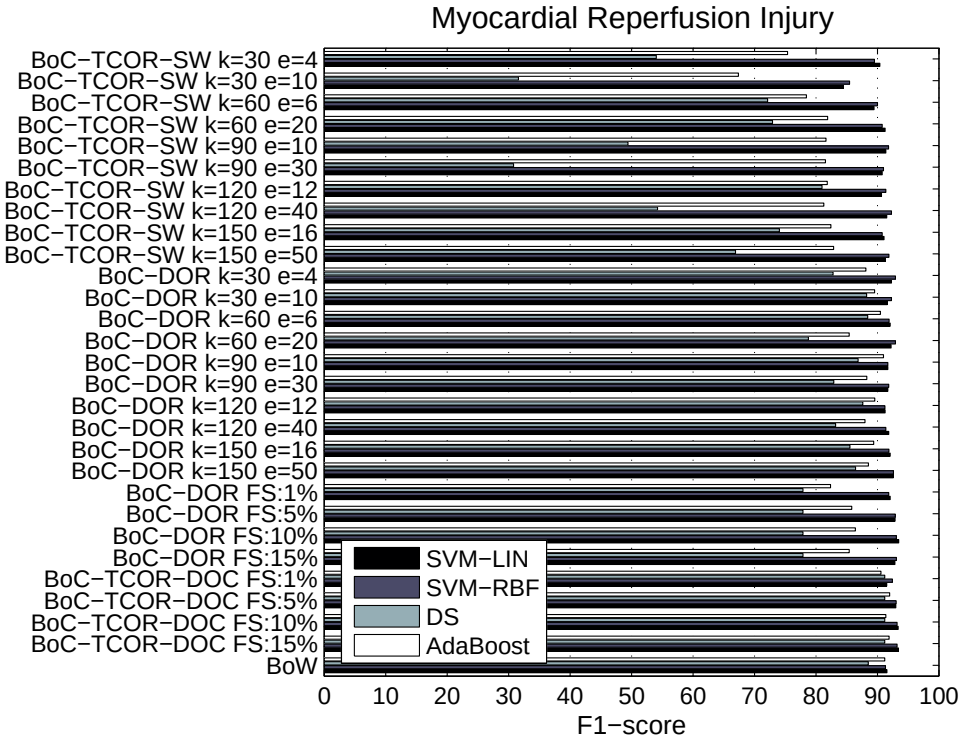

Cardiomyopathy, Restrictive
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score

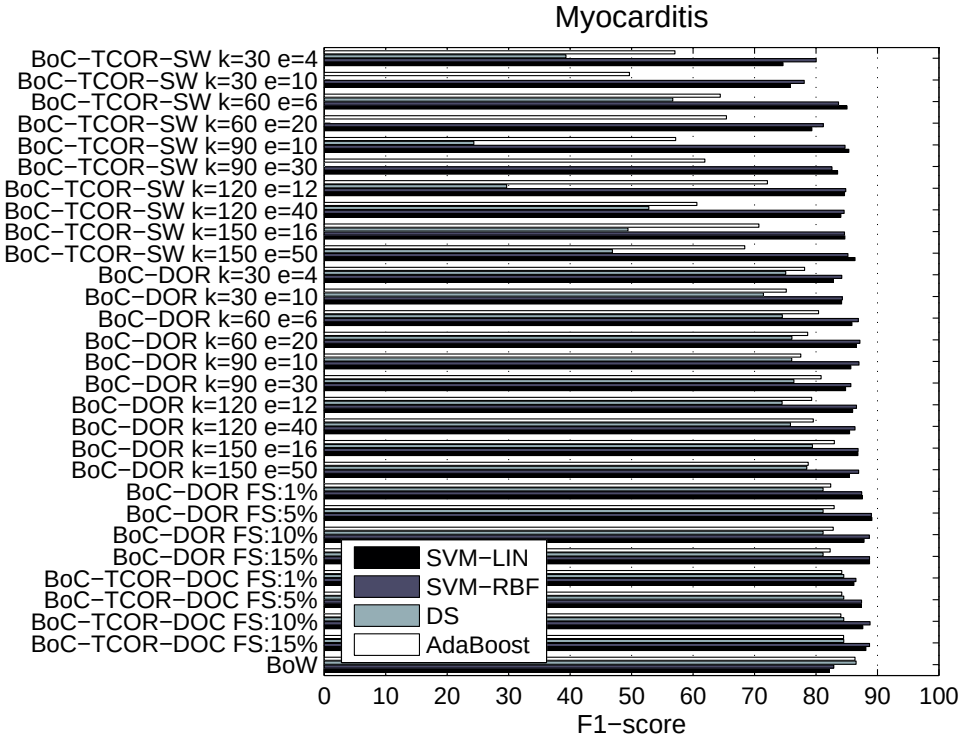

Chagas Cardiomyopathy
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score


Endocardial Fibroelastosis
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score


Endomyocardial Fibrosis
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score


Kearns−Sayer Syndrome
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score


Myocardial Reperfusion Injury
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score


Myocarditis
BoC−TCOR−SW k=30 e=4
BoC−TCOR−SW k=30 e=10
BoC−TCOR−SW k=60 e=6
BoC−TCOR−SW k=60 e=20
BoC−TCOR−SW k=90 e=10
BoC−TCOR−SW k=90 e=30
BoC−TCOR−SW k=120 e=12
BoC−TCOR−SW k=120 e=40
BoC−TCOR−SW k=150 e=16
BoC−TCOR−SW k=150 e=50
BoC−DOR k=30 e=4
BoC−DOR k=30 e=10
BoC−DOR k=60 e=6
BoC−DOR k=60 e=20
BoC−DOR k=90 e=10
BoC−DOR k=90 e=30
BoC−DOR k=120 e=12
BoC−DOR k=120 e=40
BoC−DOR k=150 e=16
BoC−DOR k=150 e=50
BoC−DOR FS:1%
BoC−DOR FS:5%
BoC−DOR FS:10%
BoC−DOR FS:15%
BoC−TCOR−DOC FS:1%
BoC−TCOR−DOC FS:5%
BoC−TCOR−DOC FS:10%
BoC−TCOR−DOC FS:15%
BoW
SVM−LIN
SVM−RBF
DS
AdaBoost
0
10
20
30
40
50
60
70
80
90
100
F1−score