A Comparison of Automatic Email Categorization Algorithms

Abstract

Email is more and more important in everybody's life. Large quantity of emails makes it difficult for users to efficiently organize and retrieve. Although there are many text categorization methods, email categorization is still challenging, because of the large quantity of its unique characteristics. Our paper managed to provide extensive analysis of email categorization using real-world email message dataset from former Enron employees. Challenges of email categorization are discussed, which do not exist in traditional text categorization. Various design choices of email categorization are presented and two automated classification methods are tested: Wide-Margin Winnow (WMW) and Supported Vector Machine (SVM). We showed experimental results using both classifiers, discussed the cause of the result and lessons we learnt, and confirmed that our pre-processing and wide-margin winnow classifier outperformed existing system in reference paper.

1 Introduction

Nowadays, emails are becoming more and more important pervasive in people's daily lives. People are usually overwhelmed by all kinds of email, some dealing with serious business, but also others, such as sign up information, ads, and spam, which is unexpected.

The explosion of emails becomes a big burden for users to processs organize and retrieve them efficiently. To deal with this problem, users folders emails by categorize emails based on their topics and originals. Doing this manually is time consuming and not human friendly. Among all solutions, email automatic categorization is one of the most promising one.

Whereas there are a lot great work on general text mining in the literatures, for example, categorize documents into different se-mantic topics (Lewis, 1992). More recently, Cohen(1996) considers a number of binary classification problems of one folder vs. all the other folders. Two classifiers, RIPPER and a tfidf. Naive Bayes are used in Provost (1999) and Rennie (2000). SVM is first adopted in Kiritchenko and Matwin (2001) for email foldering task. (Bekkerman et al., 2004), is the first work that presents an extensive benchmark test on four classifiers: Maximum Entropy, Naive Bayes, SVM and Winnow.

The email foldering task has many challenges that distinguish it from traditional topic-based categorization. Here we list some of the challenges. First, almost all custom folders are created when a new message belongs to this topic arrives. While users keep creates new folder, old ones fall out of use. For messages belong to the same folder, the topics can vary as well. For example, for a graduate student, a folder named "research" can contain messages sent by his/her supervisor discussing research, which can be very technical, or messages sent by research group administrators announcing weekly research meetings, or travel information of a conference, etc.

Second, email foldering strategies can be very different from one user to another. These differences arise from many facts, such as amount of preferred active folders, granularity of topic classification, classification criteria, folder structure (single level or nested structure). It might be interesting to see how different classifiers' performance differs for different

users.

In addition, emails arrive in a stream over time, which may span a long period. Topics in one folder may shift as time passes, which makes emails received long time ago not suitable for training classifiers. We did not find any existing analysis to this problem, but this could be an interesting problem to investigate in the future.

Last but not least, one problem in the email foldering area is that there is not much real-world, publicly accessible email dataset, which makes comparison of different classifiers very difficult.

In this paper we present a comparative study of two email categorization methods: Wide-Margin Winnow (WMW) and Support Vector Machine (SVM). Followed by a brief description of the target dataset (Enron email dataset), we present various design choices regarding to target problem representation, data pre-processing and feature construction. A detailed description is provided on how the WMW is implemented and the way SVM is applied. We have shown that our WMW outperforms not only the existing SVM package, but also the previous email categorization work that uses WMW on the same dataset. I describe my evaluation method designed particularly for email categorization task, which divides a dataset into time-based data trunk and test classification performance on each chuck based on all previous chunks.

The rest of the paper is organized as follows: Section 2 describes the data set and the way splitting the dataset into training and test set; Section 3 discusses various design choices for the experimental setup; Section 4 presents a detailed description of the classifiers adopted in this project, especially WMW; Section 5 reports and discusses the result obtained; and conclusions and future work are presented in the last section.

2 Data set description

Since one of the challenges of email categorization is the lack of standard, publicly accessible real-world dataset on which different classifiers can be tested and compared. (Bekkerman et al., 2004) performs a benchmark test on two public email dataset: one from former Enron employees, another from participants in an SRI research project. Although the Enron dataset is still available, we failed to find the SRI dataset online. The Enron dataset is provided after some clean-up and removal of attachments. The version of the dataset I uses is released on August 21, 2009.

3 Design choices

Several main design choices are presented here in order to set up the email categorization task. These design choices considered different aspects of pre-processing, cleaning and organization. Usually, email data is unstructured and messy. All these design choices are made in order to provide clean and representative training and test data for the classifiers.

3.1 Target Training Users

The dataset obtained from the web contains emails from 150 users, most of whom do not have enough messages to train the classifier. For experiment purpose, seven users who have most emails are selected: *beck-s*, *farmer-d*, *kaminski-v*, *kitchen-l*, *lokay-m*, *sanders-r*, *williams-w3*.

3.2 Removal of Non-topical Folder

Non-topical folders are folders where the emails stored do not share the same topic. Common non-topic folders in a typical email system are Inbox, Sent Mail, Drafts and Trash, etc. The reason to remove these folders in the training case is because the emails in these folders do not share common topic, which is precisely the criteria most users adopt to categorize the email. Also, because it is unnecessary to assist users classifying email into these folders, perform training on these folders is meaningless in the first place.

According to (Bekkerman et al., 2004), non-topic folders belong to three main categories: folders automatically created by email application (such as “inbox” and “sent”); folders that are standard for all the users of a certain organization (such as “all_documents” and “discussion_threads”); folders created by a particular user for archiving purpose. The first two types of non-topic folders are removed in the experiment. The third type of non-topic folder is remained mainly because the classifying strategy is different from one user to another. However, this type of personalization can be interesting potential future work.

3.3 Flatten Folder Structure and Removal of Small Folders

Most users have email folder hierarchical trees: sub-folders can be created under existing folders to store messages which logically belong to the existing folder but share more specific common topics, such as sub-folder “Japan” under folder “Travelling”.

The existence of sub-folder can make classification either easier or more difficult, depending on evaluation criteria. One may argue that if a message is misclassified, but into its parent folder or its sibling folders, fewer penalties should be enforced. However, from a user-centered design perspective, it is still expected that the message being classified exactly into the desired folder. Otherwise the user might be required to perform another manual classification so that future retrieval will be easier. Thus, I decide to flatten the folder structure so that all folders are treated equally.

Folders with s small number of messages are commonly seen, and it is expected that the classifier can deal with these folders correctly. However, training dataset that is too small will not be able to train the classifier effectively. In my experiment, folders containing less than three messages are removed from the dataset.

3.4 Feature Construction

In most cases documents are represented as distributions over features, where features may be words, sequences of words, part-of-speech tags, word clusters, and so on. Document representations usually make a transformation of dimensionality reduction.

In the pre-processing of dataset, I use Natural Language Toolkit (NLTK) as the main library for my flodering system. NLTK is a leading platform for building Python programs to work with human language data. It pro-vides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

There are a lot of design choices in the feature construction. In the paper, I used the bag-of-words document representation: messages are represented as vectors of word counts. The steps of pre-processing are as follows.

First, I consider words as sequences of alphabetic, digit and other punctuations that appear anywhere in the email header or body. Words are downcased, and Part-of-Speech tags method were applied on the words generated.

Second, in general, words with tags of noun, adjective, verb and adverb are reserved, because they represent more concrete meaning that might be related to the topic. Others e.g. pronoun, preposition, article, conjunction interjection and most punctuation marks, are filtered out.

In this step, I referred to Brown Corpus as the part-of-speech tags definition, and used Natural Language Toolkit (NLTK) 2.0 as our library for part-of-speech tagger, which is implemented in Python. It provides much finer classification mechanism. For example, a verb can be in its base form (VB), or can be in past participle (VBN). To be more specific, words with following tags are reserved: ABL, ABN, ABX, AP, CD, EX, FW, JJ, JJR, JJS, JJT, NC, NN, NN$, NNS, NNS$, NP, NP$, NPS, NPS$, NR, OD, PN, RB, RBR, RBT, RN, VB, VBD, VBG, VBN, VBP, VBZ.

Third, grouping words of different forms together further reduces the dimensionality. For example, "describes" is a 3rd singular present verb, but it's exactly the same as the base form "describe". So "describes" is transformed to its base form "describe". The same procedure is

applied for plural noun, past tense verb, morphologically superlative adjective, comparative adverb, etc. The library I used for this task is WordNet, also provided by NLTK 2.0.

At last, I removed the 100 most frequent words from the training set, and the words that appear only once in the whole training set are removed. So the remaining words are counted in each message to compose a vector.

Compared to the (Bekkerman et al., 2004), we applied part-of-speech tagger and WordNet to reduce the dimensionality of the training data, making the words more related. This extra step results in much better classification results compared to existing work, as can be seen in Section 5.

4 Classifier description

In this experiment, two classifiers are tested for the email categorization task: Support Vector Machine (SVM) and Wide-Margin Winnow (WMW), a variant of the Winnow classifier.

4.1 Wide-Margin Winnow

Winnow belongs to the family of on-line learning algorithms, which means it attempts to minimize the number of incorrect guesses during training as training examples are presented to it one at a time. It has several important theoretical properties, including the ability to easily handle domains with sparse data and high dimensionality, such as text classification. Winnow is similar to a perceptron because it attempts to construct a linear separator for each class. As a matter of fact, Winnow is guaranteed to find a linear separator if it exists. It's different from the perceptron by doing multiplicative updates instead of additive updates of the weight vectors during training.

My multi-class implementation of the Winnow algorithm is called Wide-Margin Winnow is learned from the reference paper (Bekkerman et al., 2004). The parameters used in my Winnow algorithm are as follows. Let c be the number of folders, m be the size of the feature space, and n be the number of training examples. I keep $(m+1)$-dimensional weight vectors w_1, w_2, …, w_c, all of which are initialized to contain all 1's. Given a new example (x, y), Winnow guesses its label to be $j = \text{argmax}_{i=1,\ldots,c}\{w_i \bullet x\}$. I performed t iterations through the training set, adjusting the weight vectors when its guesses are incorrectly (i.e., $j \mathrel{!}= y$). I adjusted the weight vectors by increasing w_y and decreasing w_j at those coordinates where x has non-zero feature values. The amount by which we adjust the weights depends on how many iterations we have made through the training data: during the p-th iteration, we increase weight $w_{i,f}$ by setting $w_{i,f} = w_{i,f}(1+\varepsilon\alpha^p)$ or decrease it by setting $w_{i,f} = w_{i,f}(1-\varepsilon\alpha^p)$.

If there is not only one best j (i.e. tied for first place), I still increase the true weight vector, but decrease all the wrong guessed weight vectors. This is different from the original Wide-Margin Winnow algorithm designed in (Bekkerman et al., 2004).

In addition, as an attempt to disambiguate all training examples, I use the following heuristic to keep wide margins between classes: instead of only adjusting weight vectors after an incorrect guess during training, I also adjust the weights when it barely gets the correct answer (i.e., when j = y but the ratio between the second and first largest dot products is bigger than δ.

Another detail is that when we compute dot products, the vector x is augmented with an additional $(m+1)$-th feature whose value is always set to 1.0. This is used to make Winnow classify test examples as the largest class seen during the training time in the degenerate case when there are no relevant features.

In this experiment, I set $t = 5$, $\varepsilon = 0.5$, $\alpha = 0.5$, and $\delta = 0.5$. The pseudo-code of the core implementation of Wide-Margin Winnow training algorithm is presented as below. The Winnow classifier is implemented in Python v2.7.

Algorithm 1: The Wide-Margin Winnow algorithm

Input: $\{(x_k, y_k)|k = 1, \ldots, n\}$ is training set
Output: $w_1, \ldots, w_c$ are $(m+1)$-dimensional weight vectors for each category
ε is weight adjustment rate
α is the cooling rate
δ is the confidence measure

Initialize vectors $w_1, \ldots, w_c$ to all 1's
for p from 1 to t
for k from 1 to n
$j = \text{argmax}_{i=1, \ldots, c}\{w_i \bullet x_k\}$
$j' = \text{argsecondmax}_{i=1,\ldots,c}\{w_i \bullet x_k\}$
if $j \neq y$
$w_{y,f} = w_{y,f}(1+\varepsilon\alpha^p)$ at those coordinates f where x_k is non-zero
$w_{j',f} = w_{j,f}(1-\varepsilon\alpha^p)$ at those coordinates f where x_k is non-zero
else if $w_y \bullet x_k < \delta w_{j'} \bullet x_k$
$w_{y,f} = w_{y,f}(1+\varepsilon\alpha^p)$ at those coordinates f where x_k is non-zero
$w_{j',f} = w_{j,f}(1-\varepsilon\alpha^p)$ at those coordinates f where x_k is non-zero

Given an unseen example x, guess its label to be $\text{argmax}_{i=1, \ldots, c} w_i \bullet x$

4.2 Support Vector Machine

The Support Vector Machine (Boser et al., 1992) is a very popular vector-space classification method widely used in areas such as text classification. The basic SVM takes a set of input data and predicts, for each given input, which of the two classes this data belongs to, making it non-probabilistic binary linear classifier. The goal of the two-class SVM is to find a maximum-margin hyperplane that separates training instances into two classes. By introducing various kernels, SVM can be used to perform nonlinear classification as well.

In multiclass settings, the classification problem is decomposed to multiple binary sub-problems by observing one class vs. all the others.

In this study, due to limited time range, instead of implementing m own SVM, one of the existing SVM packages, SVM *light* (version 2.20) by Thorsten Joachims (Joachims, 1999) is applied. There is existing work (Bekkerman et al., 2004) which uses the same package to perform email classification on the same dataset by using a two-class classifier. Instead, a multi-class classifier is used to see if it outperforms existing work.

5 Experimental results and discussion

5.1 Training/test set splits

In many classification settings, the standard training/test split is done uniformly at random. However, actually email datasets are constantly growing over time, so random splits may create unnatural dependencies of earlier email on later email. In this way, a more reasonable split would be the one that is based on time: train on earlier email, and test on later email.

In addition, a serious problem of this approach can arise when the test set is large: topics discussed in email far in the future may have nothing in common with the email the classifier was trained on. Another problem occurs when some of the folders in the test set did not exist at training time.

In this study, we propose an incremental time-based split similar to (Bekkerman et al., 2004): after sorting the messages according to their time stamp, we train a classifier on the first N messages and test it on the following N messages, then train on the first 2N messages and test on the following N messages, until we finally train on the first (K - 1) N messages and test on the rest of the messages. This approach provides a practical and intuitively clear split

of the data that allows us to monitor the classifiers' performance as the number of messages increases over time. If a test set contains messages of a newly created folder, so that no messages of this folder have been seen in the training data, then such test messages are ignored in the accuracy calculation. We use N = 100 for Enron datasets.

To address the problem of achieving statistically significant classification results using a time-based evaluation method, I average the results over all the training/test set splits. Intuitively, such averaging might appear inadequate, because we expect the system to improve its performance as the training set grows, but in practice this improvement is unlikely to occur, which in turn approves the applicability of the averaging approach.

I use traditional classification accuracy as our evaluation method, which is getting the ratio of correctly classified instances to the total number of instances in the test set. And the correctly classified instance means the folder that has the largest possibility to be the correct one is just the actual correct folder. In this study, due to limited time range, instead of implementing m own SVM, one of the existing SVM packages, SVMlight (version 2.20) by Thorsten Joachims (Joachims, 1999) is applied. There already exists work (Bekkerman et al., 2004) which uses the same package to perform email classification on the same dataset by using a two-class classifier. Instead, a multi-class classifier is used to see if it outperforms existing work.

5.2 Experimental Results

Figure 1 shows the results for each Enron user with both SVM and Winnow classifiers, *X* axis is the training dataset and *Y* axis is the accuracy. Each figure depicts the accuracy over the timeline for one user. Table 1 presents more detailed numerical results, with both mean and standard deviation available.

5.3 Discussion

From both Figure 1 and Table 1, it is obvious that Winnow outperforms SVM significantly, even for every user we tested. The performance of multi-class version of SVM *light* is worse than the binary version for the email foldering task on the Enron dataset. One of the reason may be that there are a bunch of free parameters needed to be tuned in SVM *light*. One important parameter, trade-off between training error and margin, was extensively tested with a wide range of values (from 0.01 to 0.15), but we failed to find any significant difference.

The Winnow classifier in this study is less accurate than the one presented in (Bekkerman et al., 2004), but not much (the average accuracy of seven users is 5% lower than their result). The reason can been speculated from the graph: although there are many data points that are higher than 60%, there are still quite a lot of data points whose values are very low (lower than 40%), and some of which reaches zero. Data points with low values can occur because of various reasons. Below are some of my understandings worth mention.

One reason is that email foldering task itself is challenging especially because different emails can have quite different words distribution, while semantically they are talking about similar topics. This indicates that using word count as features may not be adequate.

The second reason might be the way that the dataset is split in each training/test case. It is actually very common for users to create a folder and classify incoming emails into this folder. What I found in the experiment is that for many test cases, while one hundred emails are presented, most of which belong to folders that do not exist in the training cases. Recall that these emails are not included in the actual test case, which results in very few test messages. For these cases, a small number of incorrect errors will decrease the accuracy dramatically. To fix this problem, a solution that might make more sense will be collecting legitimate emails until one hundred testing emails are available.

6 Conclusion and future work

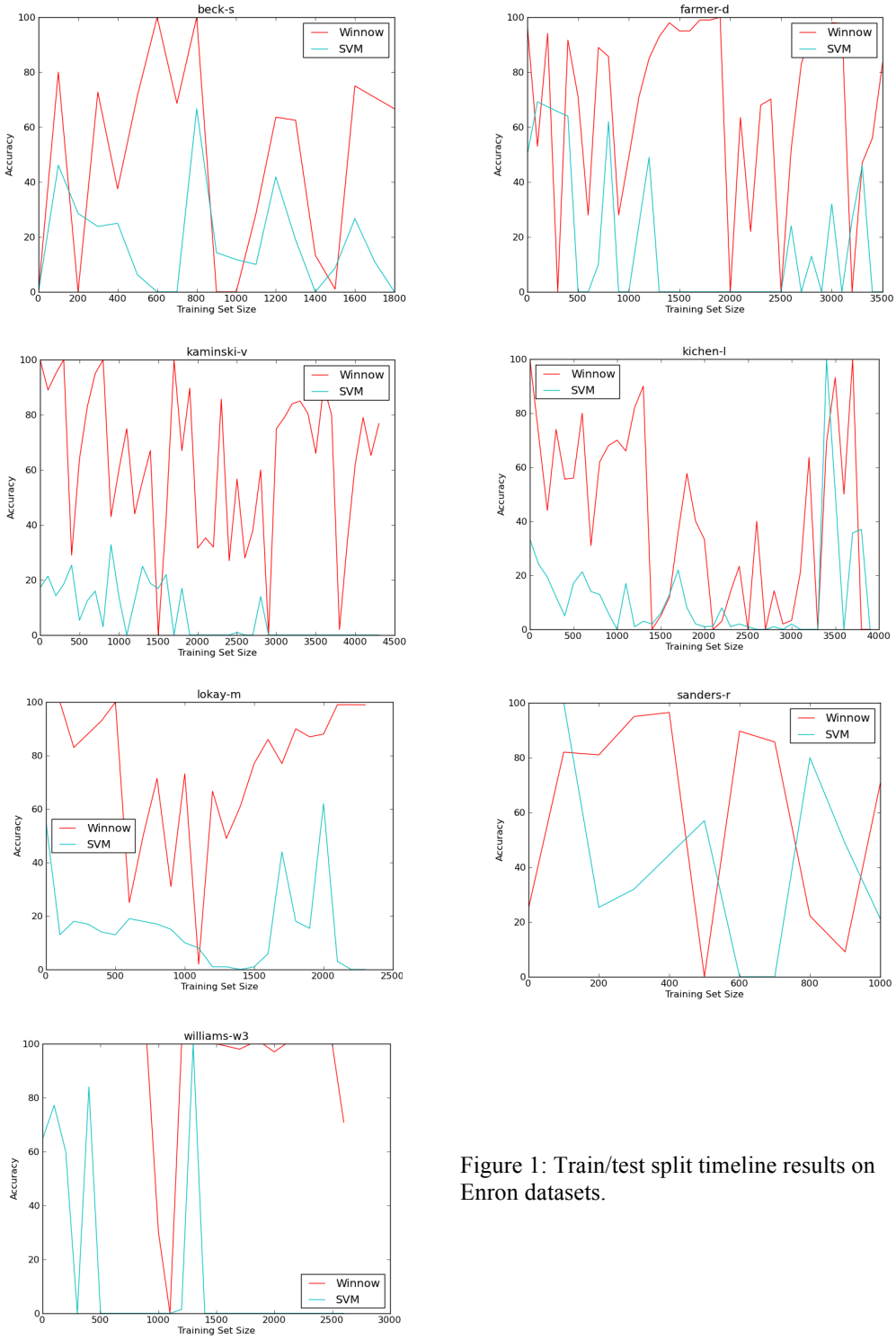

Figure 1: Train/test split timeline results on Enron datasets.

Table 1: Train/test split statistical results on Enron datasets.

User	beck-s	farmer-d	Kaminski-v	kitchen-l	lokay-m	sanders-r	williams-w3
SVM	17.9±18.3	15.0±24.0	7.0±9.5	12.0±18.9	15.3±16.5	40.5±34.3	14.3±31.2
WMW	46.7±35.5	68.0±32.4	62.5±27.8	40.0±32.7	74.8±26	59.7±35.6	92.4±22.9

In our study, the problem of email foldering is analyzed in detailed, especially various challenges which make it more difficult than general text categorization tasks. Various designed choices are discussed, including introducing Part-of-Speech Tagger and WordNet for pre-processing. Two methods, Wide-Margin Winnow and Support Vector Machine are tested on seven users of the Enron email dataset. The experimental results are analyzed in detail, and some observations from the results are presented and discussed.

For future work, there are some other classifiers we can apply to perform email foldering task, such as Maximum Entropy, Random Forest, etc. It would be valuable to test different classifiers on the same dataset to test the performance and accuracy compared with the Winnow and SVM.

We can also try some other feature construction method besides bag-of-words, such as word clusters. Word clustering is a different approach to the induction of word senses consists of clustering words, which are semantically similar and can thus bear a specific meaning. In this way, the semantic information of the email header and body could be remained, which to some degree, could improve the performance of our classifiers.

Another point worth mentioning is that in this study we did not provide any information regarding to training and testing efficiency of different classifiers. In real-world application, this can be very important criteria for engineers to select different classifiers.

Acknowledgments

I appreciate University of British Columbia, CPSC 540 Machine Learning Professor Nando de Freitas and fellow students for their valuable comment and suggestion towards our project.

References

D. D. Lewis. Representation and learning in information retrieval. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

W. W. Cohen. Learning rules that classify e-mail. In *Proceedings of AAAI Spring Symposium on Machine Learning and Information Retrieval*, 1996.

R. Bekkerman, A. McCallum, and G. Huang. Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. CIIR Technical Report IR-418 2004.

J. Provost. Naive-bayes vs. rule-learning in classication of email. Technical Report AI-TR-99-284, University of Texas at Austin, Articial Intelligence Lab, 1999.

S. Kiritchenko and S. Matwin. Email classification with co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, 2001.

J. Rennie. ifile: An application of machine learning to e-mail filtering. In *Proceedings of KDD'2000 Workshop on Text Mining*, 2000.

B. Klimt and Y. Yang. The enron corpus: A new dataset for email classi¯cation research. In *Proceedings of ECML'04, 15th European Conference on Machine Learning*, pages 217-226, 2004.

T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods – Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144-152, 1992.