

Master of Science Thesis

Multi-modal Classification in a One-class Setting

Matthijs Hovelynck

October 2009

Supervisors:

Dr. Virginia Dignum (Utrecht University)

Dr. Boris Chidlovskii (Xerox Research Centre Europe)

Prof. Dr. Albert Visser (Utrecht University)

Dr. Yannick Toussaint (Université Nancy 2)

Utrecht University

Faculty of Humanities

Cognitive Artificial Intelligence



Universiteit Utrecht

Université Nancy 2

Sciences Cognitives et Applications

Traitement Automatique des Langues



Contents

Contents	i
1 Introduction	3
1.1 Research question	4
1.2 Contribution	5
1.3 Relevance to CAI and SCA	5
1.4 Outline	6
I Theoretical Background	7
2 Statistical Machine Learning	9
2.1 Learning	9
2.2 Minimizing errors	10
2.3 Maximum Likelihood Principle	11
2.4 Model selection	12
3 Support Vector Machines	15
3.1 Maximum margin classifiers	15
3.2 Formal description	16
3.3 The kernel trick	17
3.4 Soft-margin classifiers	18
II Experimental Framework	21
4 The Enron Corpus	23
4.1 The Enron Corpus	23
4.2 Responsive documents	25
4.3 Pre-processing	26
4.4 Resulting data set	27
4.5 Document representation	28

5	One-class classification	33
5.1	Related work	34
5.2	Mapping Convergence	35
5.3	A sequence of hypotheses	38
5.4	Generating hypotheses from unlabeled data	41
5.5	Combining classifiers	42
III	Results	45
6	Results and Analysis	47
6.1	Implementation	47
6.2	Classifying Enron	48
6.3	Combining classifiers	49
7	Conclusion	51
7.1	Discussion	51
7.2	Future research	52
7.3	Main conclusions	53
	Bibliography	55
A	Database schema	61
A.1	Tables	61
A.2	Views	62
B	Reference corpora: classification results	63
B.1	INEX 2008 XML mining corpus	63
B.2	Letter recognition data set	64
C	Enron Corpus: classification results	67
C.1	Text-based features	67
C.2	Social network-based features	68
	List of Figures	69

Acknowledgements

The thesis that lies before you marks the end point of my studies in Artificial Intelligence. I am grateful that I have had the chance to compose a less than ordinary curriculum. Having finished the bachelor program in Cognitive Artificial Intelligence at Utrecht University, I continued in a joint program by Utrecht University and the Université Nancy 2, in *Cognitive Artificial Intelligence* and *Traitement Automatique des Langues* respectively. I would like to thank everyone in Nancy that made my stay there immensely worthwhile.

The second half of the year I spent in France I have enjoyed the opportunity to work in the stimulating environment of Xerox Research Centre Europe. First of all, I would like to thank Boris Chidlovskii for his continuous support and his ability to ask the right questions. Furthermore I greatly acknowledge the help I received from so many of the skilled researchers in the lab, especially Thierry Jacquin, Guillaume Jacquet, Caroline Privault, Jean-Michel Renders and Jean-Baptiste Faddoul. In general I am thankful to all the people of the Xerox community that made my stay there incredibly pleasant both in and out of the office.

I am grateful to my first supervisor, Virginia Dignum, who has read several versions of this thesis from start to end supplying me with all the suggestions and support I needed. Also I thank my supervisor at Nancy 2, Yannick Toussaint, and the third reviewer, professor Albert Visser, for the time and effort they invested.

Finally I thank my parents, Rob and Hetty, and Eefje. I am certain that without their incredible and relentless support neither my stay in France, nor this thesis would have been possible.

Chapter 1

Introduction

In a world where information becomes available in ever increasing quantities, document classification plays an important role by preselecting what we get to see and in what order. Obvious applications range from search engines to spam filtering, but also more specialist tasks can be approached with the same techniques. A particular problem, that we focus on in this thesis, is document review done by legal experts in large corporate litigation cases.

In common law judiciary systems, during the pre-trial process of discovery, litigants are commanded by means of a subpoena to bring any relevant documents to the court. In cases involving large corporations, this means that lawyers have to go through the records that those are obliged to keep and produce any *responsive* (i.e. potentially of importance to the case) documents. The number of documents under review could easily run in the millions.

The review of documents has to be done by expensive legal experts and is very time-consuming. Besides, even for human annotators the accuracy is not very high: a large mismatch usually exists between the annotations of different people. It turns out that both speed and accuracy of reviewers can be improved dramatically by grouping and ordering documents.

Technology has been developed to support human annotators by discovering structure in the corpus in order to present documents in a natural order. Usually this kind of software takes into account the textual contents of documents only (see e.g. [32]). It is our intuition that other levels of description, for example the group of people that worked on the document or the visual layout, could be of importance in distinguishing relevant items. Of course this idea, if proven fruitful, could be applied in a multitude of different tasks, but the intuition that not just textual content, but many different properties play a part in the annotation with responsiveness makes it an ideal testbed.

Conveniently, a large corpus has been around for some time now, that might provide the possibility to put our ideas to the test. The Enron Corpus consists of about 250.000

e-mails from the accounts of 150 top managers of Enron Corporation around the time of the collapse of this U.S. energy giant. E-mail also allows us to construct an alternative representation of documents, because it can be considered as documents that live in a community. The social network that is implicit in the group of people communicating offers a second level of representation, that is complementary to the textual level. Combining different levels of description is what we will call *multi-modality*.

The development of document review systems has traditionally been severely impeded by a lack of real-world annotated data. As we will see later on, corpora resembling the data that is typically encountered in corporate litigation is scarce and its annotation is virtually impossible to obtain. We circumvent this problem by obtaining a small set of labels from the context of a specific litigation trial. This leaves us with a number of responsive documents, while examples of non-responsive documents are still not available, a sub-optimal situation with respect to the development of a machine learning algorithm. For this reason we test our hypothesis with respect to multi-modality in a framework for *one-class classification*.

1.1 Research question

The Enron Corpus enables us to investigate the following central research question: ***can we improve classification performance in a one-class setting by combining classifiers of different modalities?*** Being specifically applied to the problem of distinguishing responsive documents in a corpus of e-mails, we hope that it will become clear that the same principles might be successfully applied to other classification problems.

To find an answer to our main research question, we have to define a data set that allows us to implement and evaluate our approach. While we are lacking any large set of corporate litigation data with sufficient annotation, we have to consider *how to annotate a real-world data set to represent a classification towards responsiveness?*

Given the fact that we hypothesize that a multi-modal approach will improve overall performance on classification, the second subquestion is how to construct multiple views of different modality on the documents. In the case of e-mails, more specifically we are looking at *how to construct a powerful description of documents from the social network that is implicit in a network of exchanged communication?*

We adopt a semi-supervised approach from [51] to one-class classification. A first problem we have to solve is that it assumes a clear separation between positive and negative data and performance degrades dramatically when the positive data is under-sampled. The third subquestion is therefore: *how to implement and evaluate the Mapping Convergence framework when very little positive training data is available?*

Finally we then turn to the question *how to extend the one-class framework to enable multiple levels of description to support the classification?* With this problem solved, we will be able to answer the main research question.

1.2 Contribution

This thesis' contribution can be divided in three facets. Firstly, with respect to one-class classification problems:

- We propose an alternative evaluation method for the Mapping Convergence framework [51], that allows for broader applicability by dismissing the prerequisite that positive and negative items must be naturally separable.
- We propose an extension that allows for it to be used and evaluated even when very little positive training examples are available by introducing a cross-validation step. We evaluate this extension on different corpora.
- We propose an extension based on co-training principles [6] that enables us to take advantage of the availability of multiple redundant views on the data. We evaluate this extension on the Enron Corpus, classifying towards responsiveness.

Secondly, with respect to representation of documents for classification purposes:

- We propose a way to turn the social network that is implicit in a large body of electronic communication into valuable features for classifying the exchanged documents.
- We then show that a combination of text-based features and features based on this second extra-textual modality improves classification results.

Finally, with respect to the Enron Corpus and research aimed at developing tools for supporting document review:

- We present a new, cleaned version of the Enron Corpus in relational database format. Using the e-mails on the exhibit lists published by the U.S. Department of Justice [48] we construct a dataset that can be used for training and evaluating tools for classifying towards responsiveness. We are unaware of any other attempt to bring this data into the realm of research.

1.3 Relevance to CAI and SCA

This thesis is written as a final exercise for my studies in *Cognitive Artificial Intelligence* at Utrecht University and *Sciences Cognitives et Applications* at Université Nancy 2. Cognition science is inherently multi-disciplinary, incorporating fields ranging from linguistics to computer science and from philosophy to psychology. While for students this can be both a blessing and a curse, I feel that in this project I have greatly benefited from this versatile background.

Central to both programs, I have been enrolled in for the last few years, is the relationship between human and artificial intelligence. Machine Learning, which is at the

core of this thesis, offers promising insights in exactly this field of research by allowing a cross-fertilization of ideas. The other two main pillars this thesis is constructed upon, Natural Language Processing and Social Network Analysis, also are strongly linked to the questions approached by cognition scientists.

For me, personally, the combination of insights from different fields of research is one of the most interesting of challenges. This is where progress can be made in this diverse and fascinating field.

1.4 Outline

This thesis is divided in three parts. The first, starting in the next page, aims to give an overview of the theoretical background of the Machine Learning in general and more specifically Support Vector Machines. Machine Learning (ML) is a vast and diverse field with extensive foundations in mathematics. These will be touched upon and followed by a more in-depth discussion of Support Vector Machines, one of the most prominent ML technologies available and at the basis of our approach.

Secondly we turn to our experiments. After describing the data we work with and defining the classification problem, we come to talk about the different ways we propose to represent the documents in chapter 4. Chapter 5 is devoted to describing the algorithms. We adopt a semi-supervised approach to one-class classification and develop a way to implement and evaluate it with only very little positives available. We test this method on two reference corpora. We also develop an extension to the algorithm that combines different modalities.

Finally in the third part, we present results, opportunities for further research and conclusions.

Part I

Theoretical Background

Chapter 2

Statistical Machine Learning

This chapter aims to give a short and concise introduction to the central notions of Statistical Machine Learning and the general design choices to be made when building a learning system. Hopefully this will enable the reader to continue into the next chapter without too much trouble, where we will introduce the specific Machine Learning algorithm used in the remaining part of the thesis: Support Vector Machines (SVM). Since we do not intend to be complete, but only to convey the intuitions to pave the way for chapters to come, the reader is asked to consult [29], [21] or one of the many other good text books on Machine Learning for more thorough accounts.

2.1 Learning

There are a lot of things that computers do better than us, humans. They are very good at processing huge amounts of data, they do it extremely fast and don't object to doing boring work, just to name a few.

Getting a machine to perform a task generally comes down to exactly specifying that task beforehand, although this is where trouble might come in. Sometimes the environment in which the machine will have to work is simply too dynamic or unpredictable to anticipate all peculiarities, in the extreme case it might even be completely unknown what it might look like. On another note, essential knowledge might be buried in an enormous amount of data or available only implicitly by means of domain experts. If a system that has to operate in an environment that fits to any of the aforementioned characteristics, we would prefer it to be adaptive. In other words: we would like to have a machine that learns from experience.

From Mitchell [29] we obtain a rather broad definition of Machine Learning:

A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks

in T as measured by P , improves with experience E .

In a way designing a learning machine is about putting this definition into practice. The first question the definition raises is what it is exactly that we are trying to learn. While there are no a priori limits as to what kind of structure this could be, in the case of classification we are trying to find a function $f : X \rightarrow Y$ taking objects x and producing labels y such that $y = f(x)$.

Another important aspect is the kind of information our learning machine is supposed to learn from. We can distinguish supervised and unsupervised learning. In the first case, an algorithm fits its hypothesis to a training set, that specifies pairs of input and output, trying to come up with a function that will generalize to unseen examples. We will see an example of this in the next section. On the other extreme, unsupervised learning amounts to finding structure in unlabeled data. The middle road is semi-supervised learning, where part of the training data is labeled, but the model is also supported by unlabeled examples. In any case the goal is inducing some kind of knowledge from raw data.

Learning a solution to a Machine Learning problem thus can be viewed as searching a hypothesis space for a function mapping inputs x to outputs y , as to optimize a performance measure on that mapping. Note that, unless specified otherwise, in this thesis we will maintain this notational convention. The key to stepping from this somewhat abstract definition to a practically implementable paradigm, is to explicitly define how to represent inputs, outputs and hypotheses, and specifying the performance measure.

2.2 Minimizing errors

Typically, inputs are vectors \mathbf{x} of numerical or binary represented properties of objects. Let's take an example of a supervised learning problem to get some clear intuitions about Machine Learning (example adapted from [31]).

Suppose we are trying to predict whether houses are likely to be sold within a number of months based on the asking price and the living area: a classification problem. The input space can be represented as a vector \mathbf{x} of features (e.g. number of rooms, living area, etc.).

The next step is to define the hypothesis space h . To keep things simple, we pick h to be a linear combination of the feature values in \mathbf{x} , parametrized by θ :

$$h_{\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \quad (2.1)$$

We can now clearly distinguish the role of the parameters θ . Note that the hypothesis space limits the level of agreement between the prediction of h and the actual values. If we are, as in the example, classifying from a linear function of number of bedrooms and living area only, we build quite a crude model that might leave some crucial effects unmodeled. On the other hand, deploying a high order polynomial, that matches the

training data closely, might result in a hypothesis that generalizes badly too. We will come back to this trade-off later in this chapter.

For classification we could take the following approach: when h returns a positive value, we classify the item as positive and similarly for negative values. The decision plane is then represented by:

$$h_{\theta}(\mathbf{x}) = 0 \quad (2.2)$$

We now have to find parameters θ that optimize some evaluation metric over our predictions. To represent how close the predicted values are to the actual ones, we define a cost function C over our ℓ training examples. For example we could implement the ordinary least squares model, summing over the difference between realized and expected values:

$$C(\theta) = \frac{1}{2} \sum_{i=1}^{\ell} (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})^2 \quad (2.3)$$

The goal is to find the parameters that minimize the cost function. Several algorithms exist to perform this minimization. A well-known algorithm that guarantees convergence to a minimum is gradient descent. Apply the following update rule for all components of θ until convergence, where x_0 is defined as 1 (i.e. θ_0 is the intercept) and α determines the size of the convergence steps:

$$\begin{aligned} \theta_j &:= \theta_j + \alpha \frac{\partial C(\theta)}{\partial \theta_j} \\ &= \theta_j + \alpha \sum_{i=1}^{\ell} (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}_j^{(i)} \end{aligned} \quad (2.4)$$

What we have seen so far illustrates what a typical supervised machine learning problem might look like. Obviously many other representations, evaluation functions and search algorithms are possible, but for a complete overview we again point to the references mentioned in the introduction of this chapter. The goal of this section is to use these simple examples to bring across some intuitions that will prove useful in later chapters, but first we will take a look at some other considerations concerning Machine Learning in general.

2.3 Maximum Likelihood Principle

The algorithm described in the previous section has the particularly pleasant property that it's fairly intuitive: it explicitly minimizes the difference between realized and expected values. A different perspective as to what hypothesis to pick is defined by the Maximum Likelihood Principle, stating that we should search for the hypothesis that maximizes the posterior probability density of the training examples.

An important point to take is that our model might not be perfect. Random noise might pollute our data, and there might be some unmodeled phenomena in our input

space. Let's assume that the input and output variables are related as $y^{(i)} = \theta^T \mathbf{x}^{(i)} + \epsilon^{(i)}$, where $\epsilon^{(i)}$'s are normally distributed error terms with $\mu = 0$ and unknown σ^2 .

We consider a set of ℓ observations $D = \langle y^{(0)}, \dots, y^{(\ell)} \rangle$. Since the data points are independent given the hypothesis, we are maximizing over the product of the probability densities of the separate training examples.

$$h_{ML} = \operatorname{argmax}_h p(D|h) = \operatorname{argmax}_h \prod_{i=1}^{\ell} p(y_i|h) \quad (2.5)$$

We now rewrite h_{ML} as normally distributed with mean y and the same σ^2 as ϵ :

$$h_{ML} = \operatorname{argmax}_h \prod_{i=1}^{\ell} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} (h(\mathbf{x}^{(i)}) - y^{(i)})^2} \quad (2.6)$$

Maximizing the logarithm of this function yields:

$$h_{ML} = \operatorname{argmax}_h \sum_{i=1}^{\ell} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} (h(\mathbf{x}^{(i)}) - y^{(i)})^2 \quad (2.7)$$

$$= \operatorname{argmin}_h \sum_{i=1}^{\ell} (h(\mathbf{x}^{(i)}) - y^{(i)})^2 \quad (2.8)$$

The resulting objective function is exactly the same as we picked earlier based on the principles of Least Mean Squares model. When assuming a random noise component in the training examples, normally distributed with $\mu = 0$ and independent for every example, this Bayesian approach supports the Empirical Risk Minimization formulation presented previously.

2.4 Model selection

In Machine Learning, the goal is to learn the characteristics of the underlying distribution from a limited number of training examples. Despite the fact that our hypothesis is based on a partial view of the data, how can we make sure that it will generalize well to unseen examples? As in the example earlier in this chapter, to find an optimal hypothesis we have to fix the hypothesis space. In the example we chose to search for a linear function. We could also have selected a high-order polynomial to fit the data as exactly as possible, thus obtaining a much lower training error. The three images in figure 2.1 illustrate the behavior of different hypothesis classes in a one-dimensional setting.

In the first case, we see a linear regression to the data points. The hypothesis is extremely simple, but might miss some of the characteristics clearly present in the data (i.e. the slightly quadratic distribution). The rightmost image illustrates the case of a high-order polynomial, where all data points are modeled exactly. When asking ourselves which of those models is the best for the data to be modeled, we have to ask which one would generalize best to new examples.

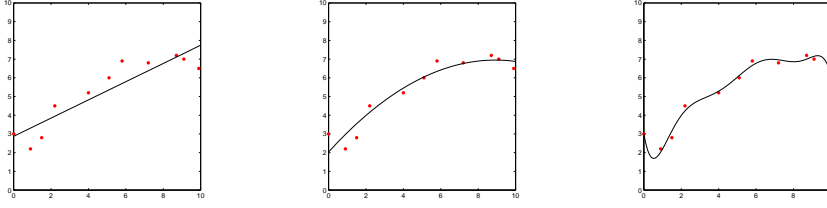


Figure 2.1: Bias-variance trade-off

Where the linear model might miss some characteristics of the distribution, it is said to have a high bias, the polynomial model might be overfitting, it has a high variance. The quadratic hypothesis (i.e. the center figure) seems to capture all relevant aspects of the data, while abstracting over the peculiarities of the specific training examples. Of course, the question is how to select the hypothesis class that yields the best generalizing models.

Under some assumptions, Learning Theory [21, 49] offers us a way of formalizing a bound on the real error based on the error realized on the training examples and the capacity of the learned hypothesis. This bound will also enable us to understand more formally the relative performance of different classes of hypotheses and thus the dynamics of bias and variance.

For a learning machine, defined as a function $h_\alpha(x), \alpha \in \Lambda$, where Λ is the set of possible parameter settings, we want to minimize the test error. Define the risk R (test error) of a hypothesis over examples of an unknown distribution P as the integral over some loss function L :

$$R(\alpha) = \int L(y, h_\alpha(x)) dP(x, y) \quad (2.9)$$

Since we do not know the distribution P , we can't calculate this value. We do have a set of ℓ training examples, that we will assume to be selected i.i.d. (independent and identically distributed) from the same distribution as the test set. The empirical risk (training error) is defined as follows:

$$R_{emp}(\alpha) = \frac{1}{\ell} \sum_{i=0}^{\ell} L(y_i, h_\alpha(x_i)) \quad (2.10)$$

As mentioned above, minimizing the empirical risk is equivalent to selecting the hypothesis with maximum likelihood. We can also use the empirical risk to estimate the risk, since the following bound has been shown to hold with probability $1 - \eta$ [49, 11]:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{d_{VC}(\log(2\ell/h) + 1) - \log(\eta/4)}{\ell}} \quad (2.11)$$

The second term in the right-hand side is called the confidence term. As can be seen, the confidence term will converge to 0 as ℓ goes to infinity. The second element

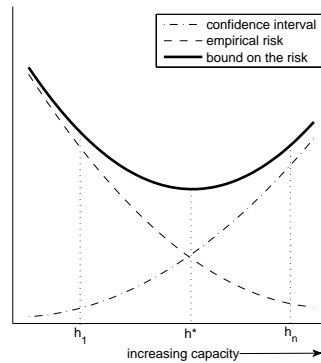


Figure 2.2: Structural Risk Minimization

that determines the size of the bound is d_{VC} , the Vapnik-Chervonenkis dimension (VC-dim), that represents the capacity of the hypothesis (i.e. its position in the bias-variance trade-off).

[21] defines the VC-dim of a function $h_\alpha(x)$ to be the largest number of points that can (in some configuration) be shattered by members of $h_\alpha(x)$. This means that any binary assignment of the points in the configuration can be perfectly assigned by the function. For a straight line in \mathbb{R}^2 , the VC-dim is 3, because three points can be shattered, while 4 cannot. In general, a linear plane in \mathbb{R}^n has VC-dim $n + 1$ [49].

When performing Empirical Risk Minimization, we can get a real positive outlook by getting the training error down to zero. However if this requires a hypothesis with a large VC-dim (i.e. with high variance) this might still result in a large test error, overfitting the data.

In figure 2.2 [49] the trade-off is illustrated. To obtain the hypothesis h^* that optimizes the prediction for the test error, we construct classes of hypotheses ordered by VC-dim. By minimizing the training error and selecting the hypothesis with the lowest VC-dim, we can select the optimal h^* . This strategy is called Structural Risk Minimization and, as we will see in the next chapter, is at the basis of Support Vector Machines.

Chapter 3

Support Vector Machines

Support Vector Machines are considered to be among the most powerful off-the-shelve learning techniques available. They have a solid foundation in Learning Theory [49], perform well on a range of learning problems [28] and are easily implemented using freely available software (e.g. [10]).

In a comparative investigation by [28], SVMs show to perform consistently well on a wide range of classification and regression problems. Even though on different problems some other algorithms are reported to achieve better performance, SVMs perform very well across-the-board. An additional advantage they offer is the little tuning of parameters that is necessary to get good performance.

Support Vector Machines have yielded good results in a large number of studies on classification [24, 35, 53]. Special attention has been given to text classification. As we will discuss more in detail in section 4.5, the specific properties of text make SVMs especially suited for the task [24].

In this chapter we will introduce the basics of Support Vector Machines, basing our account mainly on [9, 11, 21, 49].

3.1 Maximum margin classifiers

In Support Vector Machines, Structural Risk Minimization is realized as margin maximization. Still having vectors \mathbf{x} to describe objects and scalars y denoting their target classification, assume we have a linearly separable set of points in \mathbb{R}^n that we would like to classify. We define the a hyperplane in this space with the formula $\mathbf{w} \cdot \mathbf{x} + b = 0$. The weights vector \mathbf{w} is orthogonal to the hyperplane and $|b|/\|\mathbf{w}\|$ is its orthogonal distance from the origin. We now define the distance d_+ (d_-) to be the shortest from a positive (negative) point to the hyperplane. The margin between positive and negative classes is then $d_+ + d_-$.

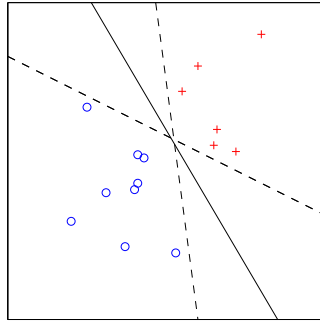


Figure 3.1: Maximizing the margin. The data can be linearly separated in many ways, but the solid line uniquely maximizes the margin.

We would now like to find the values for \mathbf{w} and \mathbf{b} that satisfy the following conditions:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1 \quad (3.1)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1 \quad (3.2)$$

Scaling \mathbf{w} and b , such that there are data points for which the equality holds, these are on hyperplanes parallel to the separating hyperplane with a distance:

$$\frac{1}{2}(d_+ + d_-) = \frac{1}{2} \left(\frac{1-b}{\|\mathbf{w}\|} - \frac{-1-b}{\|\mathbf{w}\|} \right) = \frac{1}{\|\mathbf{w}\|} \quad (3.3)$$

The larger the margin of a hyperplane, the smaller its capacity. It can be proven that the VC-dim of a maximally separating hyperplane in \mathbb{R}^n with $d = d_+ = d_- = 1/\|\mathbf{w}\|$ is bounded as

$$h \leq \min\left(\frac{R^2}{d^2}, n\right) + 1 \quad (3.4)$$

where all data points \mathbf{x} are in a sphere with radius R [49]. This means that finding the unique hyperplane that maximizes the margin is equivalent to following the Structural Risk Minimization principles.

3.2 Formal description

As becomes clear from the intuitions in the previous paragraph, learning an optimal separating hyperplane is maximizing the margin. This is equivalent to solving the following optimization problem:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (3.5)$$

$$\text{s.t. } y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 \quad \forall i \quad (3.6)$$

To transform this into a convex quadratic programming problem, we convert it to the equivalent Lagrangian formulation that we also minimize with respect to \mathbf{w} and b and

the positive Langrange multipliers α :

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^{\ell} \alpha_i \quad (3.7)$$

Setting the derivatives for \mathbf{w} and b to zero, we obtain:

$$\nabla_{\mathbf{w}} L_P = \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i = 0 \quad (3.8)$$

$$\frac{\partial}{\partial b} L_P = - \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad (3.9)$$

Substituting these back into the primal, we obtain the dual formulation, that we maximize in the positive orthant:

$$\max_{\alpha} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (3.10)$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad (3.11)$$

$$\alpha_i \geq 0 \quad \forall i \quad (3.12)$$

It has been proven that the Karush-Kuhn-Tucker conditions (3.6), (3.8), (3.9), (3.12) and

$$\alpha_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \quad (3.13)$$

guarantee that the values for \mathbf{w} , \mathbf{b} and α that optimize the primal and the dual problem are the same [9].

We can now retrieve b from one of the formulas (3.13) for which $\alpha_i \neq 0$. From the same formula also follows that for points that are not at the margin, at $1/\|\mathbf{w}\|$ of the separating hyperplane, the α is necessarily zero. This way the calculation of \mathbf{w} depends solely on the points that are on the margin, the support vectors. Resulting in a decision function

$$y^* = \text{sgn} \left(\sum_{i \in SV} \alpha_i \mathbf{x}_i \cdot \mathbf{x}^* + b \right) \quad (3.14)$$

3.3 The kernel trick

The biggest advantage of the reformulation of the original optimization problem as the dual Langrangian, is that it at the same time allows us to formulate the interaction of data points and the weight vector as dot-products. For this reason, the kernel trick enables us to transfer our search problem from the n -dimensional input space to a potentially infinite dimensional feature space. We define a transformation Φ that performs the map into some feature space \mathcal{H} .

$$\Phi : \mathbb{R}^n \mapsto \mathcal{H} \quad (3.15)$$

Instead of explicitly mapping all vectors in our equations to this high-dimensional feature space, we substitute the dot-products with a kernel function K such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (3.16)$$

An example kernel, mapping from \mathbb{R}^2 to \mathbb{R}^3 , could be $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$. This results in the same polynomial expansion as the inner product of the resulting vectors of the transformation function Φ :

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \quad (3.17)$$

The big advantage of having the kernel function working on the dot-product of vectors, is that the intermediate mapping by Φ is done only implicitly. A mapping is possible, without actually calculating all mappings of vectors into the feature space \mathcal{H} . We keep the convenient properties of a linear maximum margin classifier in the feature space, but realize a non-linear decision plane in the input space.

Some often used kernel functions are the following [21]:

$$\text{Linear :} \quad K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \quad (3.18)$$

$$\text{Polynomial :} \quad K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i \cdot \mathbf{x}_j) + 1)^d \quad (3.19)$$

$$\text{Gaussian (RBF) :} \quad K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|} \quad (3.20)$$

We will see some of the properties of these three kernels in the following chapters. It is however in principle possible to freely design kernels, as long as it is positive semi-definite (Mercer's theorem). For details see e.g. [40].

3.4 Soft-margin classifiers

C -SVM To this point, we have only considered linearly separable data. However in practice this might be too strong an assumption. In the literature some extensions have been proposed that allow for data to be not linearly separable. A first, intuitive possibility is to introduce slack variables. The cost factor C constrains the amount in which a data point can contribute as a support vector [13]. The optimization problem for this kind of soft-margin classifiers is:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} \xi_i \quad (3.21)$$

$$\text{s.t. } y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i \quad \forall i \quad (3.22)$$

$$\xi_i \geq 0 \quad \forall i \quad (3.23)$$

For the dual Langrange formulation only the second constraints changes, limiting the value of each α and influence of each support vector:

$$\max_{\alpha} \sum_{i=1}^{\ell} \alpha_i - \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (3.24)$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad (3.25)$$

$$0 \leq \alpha_i \leq C \quad \forall i \quad (3.26)$$

The parameter C controls the trade-off between the two goals of keeping both the number of training errors and the capacity of the classifier low.

ν -SVM An alternative formulation is described in [11]:

$$\min_{\mathbf{w}, b, \xi, \rho} \frac{1}{2} \|\mathbf{w}\|^2 - \nu \rho + \frac{1}{\ell} \sum_{i=1}^{\ell} \xi_i \quad (3.27)$$

$$\text{s.t.} \quad y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq \rho - \xi_i \quad \forall i \quad (3.28)$$

$$\xi_i \geq 0 \quad \forall i \quad (3.29)$$

$$\rho \geq 0 \quad (3.30)$$

By formulating the primal Langrangian, setting the partial derivatives for \mathbf{w} , b , ξ , ρ and the dual variables to zero and substituting them back, we obtain the following dual optimization problem (cf. [11] for the full deduction):

$$\min_{\alpha} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (3.31)$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad (3.32)$$

$$\sum_{i=1}^{\ell} \alpha_i \geq \nu \quad (3.33)$$

$$0 \leq \alpha_i \leq \frac{1}{\ell} \quad \forall i \quad (3.34)$$

In ν -SVM, an additional variable ρ represents the margin around the separating hyperplane. In the literature it has been noted that ν -SVM has an especially intuitive interpretation: ν is an upper bound for the fraction of margin errors and at the same time a lower bound on the fraction of support vectors.

Relation The important difference between the two formulations is the interpretation of the parameters C and ν . It has been shown that when ν -SVM leads to a positive ρ , it is equivalent to C -SVM with $C = 1/\ell\rho$ [11].

Part II

Experimental Framework

Chapter 4

The Enron Corpus

As presented in the introduction, the main research question we would like to answer in this thesis is whether a multi-modal analysis of documents can aid in classification. A particular example of a classification task that seems to require different levels of description to be taken into account is the one towards responsiveness.

Large bodies of documents, resembling the ones that are encountered when reviewing documents for corporate litigation, are very rare to appear in the public domain. For both privacy-related and legal reasons such collections are, in the rare case that they are assembled at all, usually not released. Consequently, the Enron Corpus (EC) is, with respect to its sheer size and completeness, unique in its kind. Containing all e-mails sent and received by some 150 accounts of the top management of Enron and spanning a period of several years, the total number of messages reaches about 250.000. Almost no censorship has been applied to the contents, resulting in a vast variety of subjects ranging from business related topics to strictly personal messages. It is no wonder that the EC has proved to be very attractive to the research communities concerned with fields like Machine Learning, Information Retrieval and Social Network Analysis.

In this chapter we will take a close look at the available data (i.e. the Enron Corpus) and available annotations. In section 4.3 we discuss the pre-processing done on the raw data and in section 4.4 the resulting data set is discussed. Finally we present the representations for documents in the collection. The complementary nature of a text-based representation and one based on the implicit social network in the corpus is crucial to our investigation into combining multiple modalities to aid classification.

4.1 The Enron Corpus

On 25 May, 2006 Kenneth Lay and Jeffrey Skilling, the former CEO's of Enron were convicted on a number of charges including securities fraud, insider trading and other financial crimes [33]. It was the result of a five month trial, that followed investigations

into the 2001 bankruptcy of the company. The investigation, initiated by the Federal Energy Regulatory Commission on February 13 2002, led to the publication of 2.2 terabytes of information, including a huge amount of e-mails from the top-ranked employees of the company [17], now known as the Enron Corpus.

A short history In the mid-eighties Enron started out as one of many energy companies in the United States, mainly specializing in natural gas. During the nineties Enron slowly reoriented to other commodities and during what would be known as the dot-com bubble it developed a completely new core business in energy trading. Turning electricity, water, gas and even weather into securities, Enron was elected to be U.S. most innovative company several times in a row by Forbes Magazine. Stock prices soared and the company was considered to be one of the biggest and most powerful of the United States.

It was in the summer of 2001 when one of the CEO's and mastermind behind the successes, Jeff Skilling, decided to leave the company, that the first cracks became visible in Enron's façade. In the months that followed one after the other rumor came up, causing the stocks to plummet and leading to the company's bankruptcy in December 2001. The scandal that followed revealed the shady constructions that had been invented to be able to cope with the need to keep the stock prices going through the roof. Enron's accountancy firm and several banks were involved and had to take heavy losses. Enron had shamelessly manipulated the recently deregulated California energy market, leading to the west coast energy crisis in 2001. In the meanwhile the top managers cashed out millions while reassuring the market that everything would be all right.

Enron turned into one of the biggest corporate scandals the world has ever seen. Dozens of executives have been convicted to sentences of up to 24 years in prison [1].

The corpus In the course of the trials, a great deal of the data that has been gathered by the investigating agencies has been made public. A mildly preprocessed version of the e-mail data became available in 2004, thanks to the work of William Cohen of Carnegie Mellon University [12]. He transformed the raw data into a more usable format, that has since been used by many around the world for research in a range of fields.

The original data set consists of about 600.000 text files, ordered in 150 folders, each representing an e-mail account. In these folders any original structure the users created has been preserved. Even though the files are raw text, each contains separately the body and several of the e-mail headers. Some preprocessing has been done on the headers (e.g. on some places e-mail addresses have been reconstructed). Attachments are not included.

The Enron corpus is unique in combining real-world social data with a huge scale. It has opened up a completely new realm of possibilities to the research community. A selection: subject classification [27] and folder detection [5], social network analysis [41] and hierarchy detection [38], identity modeling [15], reconstruction of threading [50] and large (e-mail) network visualization [22].

Our approach is complementary to previous work in that we explicitly use the modality of e-mail for classification. E-mail does not only consist of text, but it also implicitly instantiates a social network of people communicating. We will construct document representations modeling these distinct levels and combine them for classification. We chose to classify towards responsiveness, whether or not the message is of interest with respect to a particular litigation trial. Intuitively such a decision requires an integration of different aspects: not only the topic of an e-mail, but also the sender and receivers are relevant.

Several versions of the corpus are available online. Besides the raw text corpus by William Cohen [12], also a relational database version is available [41]. In this existing database, however, some important information regarding the social network has been discarded. Since this information is crucial to our efforts, we have decided to construct a database from scratch. In paragraph 4.3 we will describe the steps taken to clean all data.

4.2 Responsive documents

When large corporations get involved in litigation and receive a subpoena to produce evidence regarding the case, this usually means they have to go through enormous amounts of data to be able to obey the court order. Since companies are required by law to keep records of all documents that are created internally, the number of documents that has to be taken into account can run in the millions. Review has to be done by legal experts working at a rate of 400-1000 documents per day, leading to enormous costs. The economic interest of speeding up and improving the process of finding documents that are responsive (i.e. relevant) to the subpoena is huge.

Studies have shown that tools that group and order documents yield good results. In order to develop and test a strategy to support human document review we would ideally dispose a large set of pre-annotated data. Several attempts have been made to manually annotate parts of the Enron Corpus (e.g. [5]). All of these are relatively small (several thousands of messages) and typically annotated with subject, emotional tone or other primary properties. Annotation with responsiveness is tedious and expensive, requires legal expertise and is for those and other reasons usually not publicly available.

The solution we came up with is to use the lists of government exhibits published on the website of the United States Department of Justice (DOJ). More specifically we use the set of e-mails that has been used in the trials against the two CEO's of Enron, Ken Lay and Jeff Skilling. The assumption is that this set of documents is constructed by legal experts to represent all aspects of the entire data set with respect to the case. Of course, a drawback could be the fact that it is just a subset of potentially relevant items.

From the DOJ we obtain a list of 186 e-mails. The fact that this is just a very small set compared to the size of the entire corpus could be a major drawback. Also, while

we expect that the set that made it to the court somehow covers all types of e-mails found to be relevant, we also expect that a lot of tokens did not make it there. The corpus potentially contains many practically identical messages that do not appear on the exhibit list.

The fact that responsiveness is an ideal target for multi-modal classification, as argued in the introduction, and this set does indeed reflect this property are strong enough reasons to use the DOJ set. We added the messages to the corpus, some of them were already in there others were completely new. Any duplicates will be resolved by the pre-processing discussed in the next paragraph.

Remark that the decision to work with the DOJ set means that we have to work in a one-class setting, because we have only obtained labels for the positive set. The algorithm we will develop in the next chapter will be specifically aimed at classifying items based on a very small set of positive training examples and a large amount of unlabeled data explicitly using multiple views of the objects.

4.3 Pre-processing

Due to its relational nature and enormous size, we convert the raw data to a relational database. In the database we store all data that is directly related to the original corpus. We have tables for messages, correspondents, files, actors, words and word frequencies. For the database schema see appendix A.

The Enron Corpus contains a large number of duplicate messages, ambiguous references to actors and other inconsistencies. The first problem is partly due to the fact that some of the e-mail folders are automatically generated (e.g. *all documents*, *sent items*). Also e-mails may appear both in the sender's *sent items* and in the receiver's *inbox*. The first step of preprocessing after putting all 500.000 files in the database, is unifying identical messages based on title and a digest of the body, immediately reducing the number of messages by a little over 52%.

The second problem is the existence of ambiguities and inconsistencies among the references in sender and receiver fields. These references are often just names, coming from the personal address book of the user. To be able to extract the implicit social network, we will have to identify these references. As a result we have to find out which ones are in fact pointing to the same person. Consider that things like 'Richard Buy', 'Rick Buy', 'Richard B. Buy', 'richard.buy@enron.com', 'rbuy@ect.enron.com', 'Buy-R', etcetera probably all refer to one and the same person.

Using regular expressions we extracted firstname, lastname and, when available, e-mail address from all references. Then, as a first step, we reassemble references that occur in the headers of the same message. The idea is that often both a name and a e-mail address occur in the header and the knowledge that we have an agent 'Mark Fisher' that has the e-mail address 'mf@enron.com' could be valuable information. Within the

limited scope of a message header field we can be quite liberal in these unifications.

Having recombined these different references, the next step is to relate them to the references in other messages. An actor is a collection of references that have been identified as pointing to one and the same person. We use the email address as a primary cue: if we assume that if two references share an e-mail address, they surely refer to the same actor. Secondly, for each yet unidentified reference we search for possible matches in the set of actors with the same last name based on first name, prefix, middle name and/or nicknames (from a list of common US English nicknames). Provided a group of similar references refer to at most one actor in the network, we assume all yet unidentified references refer to that actor. As a result of these two steps, all different shapes of Richard Buy's name are in fact discovered to refer to the same person. In total we find that 167.274 different references can be grouped as 114.986 actors. This includes a large number of references that occur only once and a small number of more central actors that is referred to in many different ways.

To be able to easily construct bag-of-words representations later on, we finally generate the lexicon making up the total collection of messages. we constructed a word list and word frequency table from the bodies of the messages. We filter strings of length smaller than 3, apply a Porter stemmer and bring all words to lower case.

4.4 Resulting data set

Cleaning the Enron Corpus reduces noise considerably. Messages have been deduplicated and references to actors have been resolved. In the resulting dataset, based on the 517.617 original files, we have 248.155 messages and a network of 114.986 different actors (but setting even a low threshold on activity will dramatically decrease this number). Even though we can conclude that ambiguity and inconsistency have been reduced, some may still exist. For example, a considerable number of messages are found to have more than one sender. For our purposes this kind of errors is rare enough to be treated as noise.

Some characteristic plots are displayed in figure 4.1. Starting with the first plot, we see that, as predicted by Zipf's law, the frequency of words is inversely proportional to their rank based on frequency. The second plot shows the distribution of message sizes. The main peak is around 11 words, with most mass for lengths between 10 and 300 words. Important to see is that the average e-mail is a relatively short text, one more reason to try to use other properties in classification. The third plot tells us that even though there are some very active actors in the network, most are sending very few e-mails. Just as well the number of receivers per message shows a Zipf-like distribution: there are some e-mails with an enormous amount of receivers (up to 1000), but most communication is aimed a small group of recipients. Finally the fifth plot displays the number of e-mails per week that are sent over the timeframe of interest to the Enron Corpus. Vertical lines indicate that a message in the DOJ subset is in that week. We

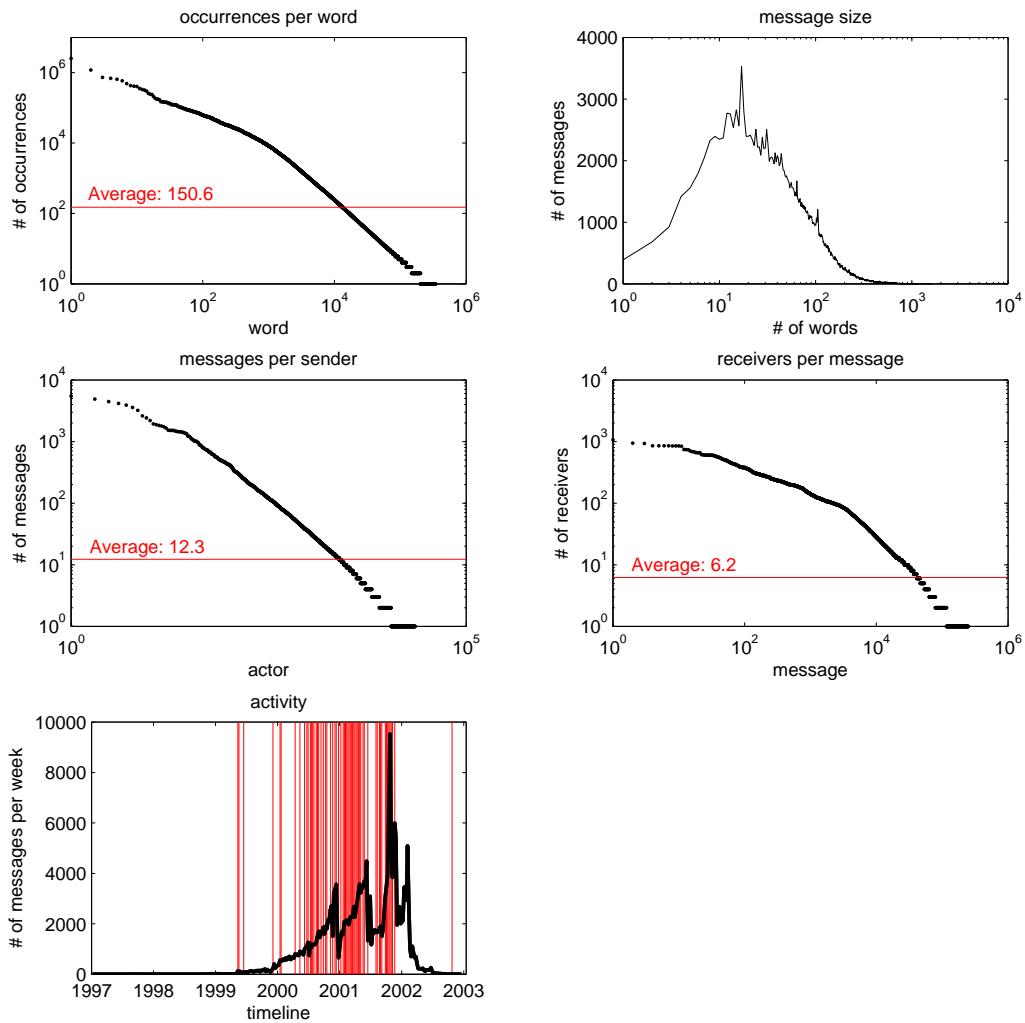


Figure 4.1: Characteristic plots of the resulting data set

can see that the timestamps of e-mails on the exhibit list are clustering around crucial dates like the California energy crisis (mid-2000 until mid-2001), the period leading up to the departure of CEO Jeff Skilling (mid-2001) and the bankruptcy (end 2001).

4.5 Document representation

With the data cleaned and processed to a more usable format, we now turn to the representation we use for e-mails. We will show that combining multiple complementary representations, that both are discriminative enough to successfully classify the collection, is beneficial to the overall performance.

The most obvious level of representation, and surely relevant to the distinction responsive/non-responsive, is the textual content of the documents. We include the contents of body and subject fields in a bag-of-words representation, that has proved

to be suitable for classification purposes. The idea is to construct a vector where each feature represents a word of the lexicon and the feature values express some weight.

Secondly we propose a representation based on the role of senders and receivers of the message in the network of people exchanging information. Intuitively this second level of description also plays a role for human annotators in deciding whether or not a document is responsive to a subpoena.

Other views of the documents are imaginable, but we leave those to future research. This section will describe the feature sets we constructed in detail. First we turn to the more traditional text-based representation, than we discuss the representation we build based on the social network that is implicit in the network of communication.

Text Recall the generation of the lexicon described in section 4.3. We impose a number of criteria on the terms: a minimum occurrence of 4 in the entire corpus and a minimum length of 3. Also, all keywords are stemmed with a Porter stemmer and lowercased.

From this initial lexicon we construct vectors with each feature representing one word. For the values we experiment with tf-idf, frequency and binary values, because mixed results have been reported in the literature. The feature values for a word w in a document d are calculated as follows. The document frequency df of a word w is the number of documents in occurs in, the term frequency tf is the number of times it occurs in a document d , N is the number of documents.

$$\begin{aligned} \text{binary}(w,d) &= \begin{cases} 1 & \text{if } w \text{ occurs in } d \\ 0 & \text{otherwise} \end{cases} \\ \text{frequency}(w,d) &= tf \\ \text{tf-idf}(w,d) &= tf \cdot \log(N/df) \end{aligned}$$

The feature set based on a bag-of-words representation has some properties that make it particularly suitable for SVM classification with a linear kernel [23, 24]. First of all, the feature is very high-dimensional (around 105.000 in this case). Also the vectors are very sparse, because only a small number of the words actually occur in each respective document. Thirdly, the terms that do occur in documents may have a huge overlap, even if the documents belong to different categories. And finally there is a lot of redundancy in the feature vector: a document typically has a large number of cues that signal a particular classification. For these reasons we believe that also with one-class based SVM algorithms, the linear kernel will perform best.

A problem that has been noted with extremely high-dimensional feature spaces for one-class classifier is the so-called *curse of dimensionality* [16]. The observation is that noise hurts performance significantly, especially in the case of learning from unbalanced data. Intuitively the distance across a multi-dimensional space grows exponentially with the number of dimensions, obscuring details. In practice this means that the behaviour of one-class SVM algorithms becomes more and more unpredictable.

To investigate this effect with respect to our algorithm we constructed two alternative sets of text-based document representations. First we select a subset of words that have the highest document frequency. The intuition is that the ones that occur in almost no documents are unlikely to carry much information. This allows us to select any number of features from the entire set.

The other way of reducing the dimensionality is by clustering features semantically. We use a soft clustering approach proposed in [3]. Maximal cliques in the co-occurrence graph are clustered to obtain a score indicating the probability a word belongs to each cluster. Using this approach we obtain feature vectors of 6522 features, each representing a deduced semantic field. Ambiguous words contribute to multiple features.

Implicit social network Network structures have received a lot of attention in scientific literature and have proven to be useful in diverse fields like sociology, biology, engineering, computer science and epidemiology [43, 30]. With the advent of the power to handle huge graphs and the emergence of huge, real-life and easily accessible linked structures on the internet, structure in social networks has become to be a much researched and well-understood phenomenon [19].

Classifying objects based on the relations that exist among them, is known as Link Mining [18]. The idea is to base a description of data on the structure it lives in. Typical tasks include for example subgraph discovery (i.e. graph related), link prediction (i.e. link related) or classification (i.e. object-related). Obviously this last application has our specific attention. For a survey of previous work on Link Mining, see [18].

The structure of a large corpus of e-mail, like the Enron corpus, clearly is not homogeneous. The lines of communication implicitly instantiate a network of actors. By setting thresholds on the number of e-mails on a connection, it is possible to generate versions of the graph with different levels of connectedness. We set this threshold to 2, making sure to take the majority of the traffic into account while discarding any accidental links with no or little meaning. Thanks to the Zipf-like distribution of the connection strenghts this reduces the number of actors to take into consideration considerably, without losing much relevant information. We take the largest connected subgraph (95% of the remaining actors) of this correspondence graph, resulting in a network of 30.094 actors.

Intuitively we would like our social network feature set to represent the position of correspondents in the corporate network. It has been shown that certain properties of nodes in a communication graph can serve very well to automatically detect the social role of actors in the network [38, 47].

We adopt a set of commonly used features to represent key properties of actors [38, 7]. The communication network is a directed weighted graph $G = \langle V, E \rangle$, with n vertices V and edges E . The first of twelve feature representing an actor is the activity of the actor in the network:

1. the number of e-mails sent.

In hypertext classification two features, that represent the authority that is assigned to nodes by its peers, have proven to be very valuable. Nodes with a high number of incoming edges from hubs are considered to be authorities, nodes that link to a large number of authorities are hubs [25].

2. hub score;
3. authority score.

A range of different centrality measures have been proposed to model the position of the node in the network. These depend on a undirected unweighted version of the communication graph. We calculate the shortest paths in the graph with a Matlab library for working with large graphs [20]. We obtain the distance from node s to t d_{st} and the number of paths from s to t σ_{st} . The number of paths from s to t via v is denoted as $\sigma_{st}(v)$.

4. mean centrality: $C_M(v) = \frac{n}{\sum_{s \in V} d_{vs}}$;
5. degree centrality: $\deg(v) = |s| : (v, s) \in E$;
6. betweenness centrality: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$;
7. closeness centrality: $C_C(v) = \frac{1}{\max_t d(v, t)}$;
8. stress centrality: $C_S(v) = \sum_{s \neq v \neq t} \sigma_{st}(v)$.

One feature characterizes the connectedness in the direct neighborhood of the node:

9. clustering coefficient: $CC(v) = \frac{2|(s, t)|}{(\deg(v)(\deg(v)-1))} : (v, s), (v, t), (s, t) \in E$.

For the final group we calculate all cliques in the graph with a Matlab implementation of [8]. As features we adopt:

10. the number of cliques an actor is in;
11. a raw clique score: a clique of size n is assigned a weight 2^{n-1} ;
12. a weighted clique score: a clique of size n , with w the sum of activities of its members, is assigned a weight $w \cdot 2^{n-1}$.

All twelve scores are scaled to a value in $[0, 1]$, where 1 indicates a higher importance. Each node can be assigned a social score [38]: a linear combination of these features. We take all features to have equal weight.

Crucially, we are not classifying the nodes in the social network, the actors, but messages that have been sent and received by these actors. To translate the properties of actors to properties of e-mails, we construct a set of 37 features to represent each message. A message is represented by three sets of 12 features, the first is the properties

of the sender, the second the average of the properties of all receivers and the third is the properties of the most prominent receiver (i.e. with the highest social score). The last feature is the number of receivers. This set of features based on the social network implicit in the corpus represents a quantification of the sender and receiver characteristics of the message.

Chapter 5

One-class classification

We are now in the position to clearly state the classification problem and identify potential bottlenecks. Recall that our main research question is whether a combination of classifiers that cover different aspects of the documents under review will aid their classification. We use a large e-mail corpus, aiming for a classification towards responsiveness to a subpoena. This finds its justification in the fact that it intuitively is defined by multi-modal characteristics (e.g. the textual content and the role of the correspondents in the organizational structure of the company). A big disadvantage that this results in is the fact that a gold standard classification for litigation responsiveness is, due to legal, economical and personal interests of the parties involved, is very hard to get hold of.

The Machine Learning problem that follows from this description and the available data is discussed in the previous chapter is very typical for a particular class of Information Retrieval problems. A small set of positive training examples is usually relatively straightforward to obtain (e.g. the medical records of patients that are diagnosed with a particular disease or the e-mails that occur on the exhibit list published by the DOJ). However it is often very difficult (if not impossible) to develop a set of objects that completely reflects the complement of the positive class. Illustrative is our case, where we have a clear intuition about what documents would be responsive, but the negative items are, more than anything, everything else.

To sum up, these are the key characteristics and assumptions of the problem we are approaching:

- In a very large set of documents, we have a small set of truly positive examples, leaving the rest to be unlabeled;
- The ratio between positive and negative items in the corpus is unknown, but assumed to be unbalanced where the positives are the minority class;
- The positive items are assumed to be drawn from a certain distribution, whereas the negatives are everything else.

Traditional Support Vector Machines depend on multi-class training data (i.e. labels are available for more than one of the target classes). The first section of this chapter is devoted to an attempt to define and SVM-based algorithm that is capable of constructing hypotheses based on positive trainings examples only. It turns out that in practice those are quite sensitive to choice of features and parameter settings and for that reason hard to implement. Underfitting and overfitting are always close [16, 26, 37].

We then turn to a semi-supervised framework, Mapping Convergence by [51], that not only looks at the positive examples, but is also supported by the huge amount of unlabeled data that is available. This will be the basis for our own approach that is described in sections 5.3, 5.4 and 5.5. The core idea is to confidently obtain artificial negative examples from the set of unlabeled objects that will support a process of convergence towards an optimal hypothesis. We adapt the interpretation to make it possible to start with a very sparse set of positives and still obtain a good estimate of the performance of the generated hypotheses. As a second step, in section 5.5, we extend the algorithm implementing ideas from co-training [6] to combine different and complementary classifiers.

5.1 Related work

A few extensions of the SVM algorithm have been proposed that enable learning in a one-class setting. Both the Support Vector Data Description algorithm (SVDD) [45] and One-class Support Vector Machines (OC-SVM) [39] are shown to be equivalent when data vectors are scaled to unit length [44]. We will take the path of OC-SVMs, enabling us to formulate the optimization problem as in ν -SVM and keeping the intuitivity of that parameter.

OC-SVM The main idea behind One-class SVMs is to create a hyperplane in the kernel space that separates the projections of the data from the origin with a large margin. The data is in fact separable from the origin, if there exists a \mathbf{w} such that $K(\mathbf{w}, \mathbf{x}_i) > 0, \forall i$. For the special case of a Gaussian (Radial Basis Function, or RBF) kernel, the following two properties guarantee this:

$$\text{For } K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|}: \\ K(\mathbf{x}_i, \mathbf{x}_j) > 0 \quad \forall i, j \quad (5.1)$$

$$K(\mathbf{x}_i, \mathbf{x}_i) = 1 \quad \forall i \quad (5.2)$$

Property 5.1 ensures that the dot-product of any two projections in kernel space is positive, so the value of the components of any projection vector is positive and we are looking at the positive orthant. From 5.2 it follows that the length of any projection is equal to 1, we are on the unit sphere. Therefore the data is separable from the origin.

Also in [45] the Gaussian kernel is shown to give a tight fit to the data, much tighter than for example a polynomial kernel. For these two reasons it seems to be very well

suited for use in a one-class setting. However there is no principal reason why other kernels would not work.

As is pointed out in [39], the connection between One-class Support Vector machine and binary classification is fairly strong. Supposing we have a parametrization (\mathbf{w}, ρ) for the supporting hyperplane of a data set $\{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$, we also have that $(\mathbf{w}, 0)$ is the parametrization of the maximally separating hyperplane for the labeled data set $\{(\mathbf{x}_1, +1), \dots, (\mathbf{x}_\ell, +1), (-\mathbf{x}_1, -1), \dots, (-\mathbf{x}_\ell, -1)\}$.

Also, supposing that we have a maximally separating hyperplane parametrized by $(\mathbf{w}, 0)$ for a data set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$ and with a margin $\rho/\|\mathbf{w}\|$, we know that the supporting hyperplane for $\{y_1\mathbf{x}_1, \dots, y_\ell\mathbf{x}_\ell\}$ is parametrized by (\mathbf{w}, ρ) . For the non-separable case, margin errors in the binary setting correspond to outliers in the one-class case.

To find the supporting hyperplane for a distribution, we solve the optimization problem of ν -SVM. Slightly extending the interpretation in the previous chapter, in a one-class setting ν represents an upper bound on the fraction of outliers (margin errors) and a lower bound on the number of support vectors.

Applications One-class classifiers can be used to estimate the distribution for a class of training examples. There are two types of Machine Learning problems for which one-class learning is very attractive [53]. In the first case we have a majority set that is well-sampled, but we are unable to get any reasonable sample for the minority class. This is the case in [35] and [34] where an ensemble of one-class classifiers is used to detect malicious traffic on a computer network. The basic assumption is that most traffic is normal and attacks are atypical. This problem is known as outlier detection.

With the second type of problems the target class can be accurately sampled, but the data is extremely unbalanced. In regular multi-class classification, this would result in a bias towards the majority class, by rebalancing (of which one-class classification is the extreme) this problem can be circumvented [37]. A particular example of a situation like this is document classification [37, 52, 26].

In a comparative study by Manevitz [26] the performance of OC-SVM is compared to other algorithms that are applicable in a one-class setting, such as Rocchio, Nearest Neighbour and Neural Nets. It turns out that OC-SVM and Neural Networks are the two most powerful algorithms. Their main conclusion with respect to OC-SVM is that its performance is very sensitive to choice of parameters and features. A big advantage however is that it is computationally much less intensive than Neural Networks.

5.2 Mapping Convergence

The small number of positive training examples and the sensitivity of OC-SVM seem to suggest that applying it to our classification problem will lead to poor performance.

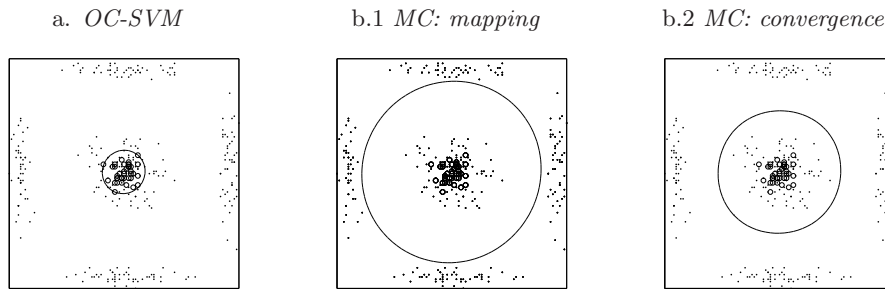


Figure 5.1: Artificial data in \mathbb{R}^2 . Pluses are positive data, dots unlabeled.

Fortunately, besides a small number of positives, we also have a very large set of unlabeled data. If we can use some sort of semi-supervised framework that can benefit from the additional knowledge we have about the distribution of the unlabeled data, we might be able to bootstrap the classification. This is where the Mapping Convergence framework [51] comes in.

The basic idea behind Mapping Convergence (MC) is to exploit the gap between negative and positive data [52]. What is meant by this, is best illustrated by figures 5.1, reproduced from [51], that show how the distribution of unlabeled data is used to obtain a boundary with good generalization performance.

Figure 5.1.a shows the OC-SVM tightly fitting the positive data (circles), ignoring the underlying distribution of unlabeled data (dots). Since we have no further information, the safest assumption is that the true distribution of positive data is that of the subset we know.

A crucial intuition for Mapping Convergence is that, given a hypothesis h , items that are further from the decision boundary are classified with a higher probability. In other words: if we have a description of the volume where the positive training examples live, we can take items that are furthest away from it to be most dissimilar to the positive items. Figure 5.1.b.1 shows the mapping stage, where artificial negative items are created by labelling the ones most dissimilar to the positive training examples.

This first approximation of the negative distribution serves as input for the converging stage to move the boundary towards the positive training examples. We train an SVM on the positive training set and the constructed negatives. The resulting hypothesis is used to classify the remaining unlabeled items. Any unlabeled items that are classified as negative are added to the negative set. The converging stage is iterated until convergence, reached when no new negative items are discovered and the boundary comes to a hold. In figure 5.1.b.2 we see the result of the convergence, that places the boundary between clusters in the unlabeled data. In algorithm 1 is a formal representation of the Mapping Convergence framework as it is proposed in [51].

Especially instructive is figure 5.2 reproduced from [51], showing seven clusters in one

Algorithm 1 Mapping Convergence**Require:**

- positive data set P
- unlabeled data set U
- negative data set $N = \emptyset$
- OC-SVM: C_1
- SVM: C_2 .

Ensure: boundary function h_i

- 1: $h_0 \leftarrow \text{train } C_1 \text{ on } P$
- 2: $\hat{N}_0 \leftarrow \text{strong negatives } (\leq 10\%) \text{ from } U \text{ by } h_0$
 $\hat{P}_0 \leftarrow \text{remaining part of } U$
- 3: $i \leftarrow 0$
- 4: **while** $\hat{N}_i \neq \emptyset$ **do**
- 5: $N \leftarrow N \cup \hat{N}_i$
- 6: $h_{i+1} \leftarrow \text{train } C_2 \text{ on } P \text{ and } N$
- 7: $\hat{N}_{i+1} \leftarrow \text{negatives from } \hat{P}_i \text{ by } h_{i+1}$
 $\hat{P}_{i+1} \leftarrow \text{positives from } \hat{P}_i \text{ by } h_{i+1}$
- 8: $i \leftarrow i + 1$
- 9: **end while**

dimensional space. In reality the fourth cluster from the left is positive, but all data is unlabeled except for the dark subset of the positive cluster. The optimal boundary is represented by the dashed lines, but a generic one-class algorithm would end up tightly fitting the positive training data on (b_p, b'_p) .

Mapping Convergence is able to take advantage of the underlying structure of the unlabeled data and give a good approximation of the optimal boundary. The initial hypothesis places the boundary on (h_0, h'_0) , generating the first approximation of N with data that is far away from the one-class boundary. The convergence step that follows moves the boundary to (h_1, h'_1) , adding the unlabeled data that is recognized as negative by h_1 to N . This process is iterated until no new data is added and the boundary remains fixed.

Each new hypothesis h_{i+1} maximizes the margin between h_i and b_p . Remark that when the new boundary is not surrounded by any data, it retracts to the nearest point where the data does live (cf. where the boundary moves from h_2 to h_3). For the algorithm to converge, there must live no data in the volume covered by the final step. For this reason, to be sure that it does not over-iterate, MC depends on a gap between positive and

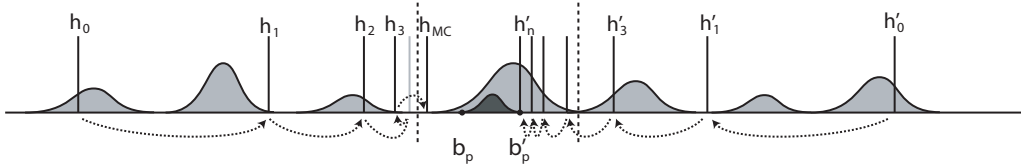


Figure 5.2: An illustration of MC in 1-dimensional space.

negative data of at least half the distance between h_{i-1} and b_p . Because this condition gets easier to meet as the boundary gets closer to b_p , the algorithm automatically is unlikely to select a boundary that is very far from the training data. On the right-hand side of image 5.2 we see that there is no gap that stops the convergence, resulting in the boundary being placed on b_p .

There is a trade-off in deciding how much data to label in the mapping stage. On the one hand, a bigger initial set of artificial negatives will better cover the distribution of the negatives. On the other, putting items in \hat{N}_0 that do not belong there results in poor performance because the effect will accumulate in convergence. Our experiments show that a proportion of 5 – 10% is enough to support the convergence and does not hurt the performance on known positives very much.

5.3 A sequence of hypotheses

Now that we have discussed the framework Yu [52, 51] proposes, it is time to turn to our own contribution. In the original papers, it is already noticed that the performance of MC dramatically decreases when the positive training set is severely undersampled. The explanation is that the algorithm is not able to detect the gap between positive and negative data in an environment where too much unlabeled items act as noise obscuring the border in the data. Iteration will not stop, causing the algorithm to overfit.

Given the nature of our problem, we are facing exactly such a situation where we have only a small amount of positive data and we are not able to enlarge this set in any way. To handle this, we will have to find a way to stop the iteration, before over-iteration occurs. We will consider the convergence to be a sequence of hypotheses, where each step is represented as a decision boundary in the feature space. The solution to the problem of over-iteration is finding the hypothesis that maximizes some performance measure.

We propose the following way of qualifying performance. Note that we don't know the size of the subset of the data that is to be labeled as positive. Notwithstanding, the key idea is to return a small part of the data that contains a large part of the positive items. In that way we will find a hypothesis that strikes a good bias-variance balance.

In this paper we use curves like the one in figure 5.3.f to reflect the distinguishing power of a classifier. On each convergence step of MC a classifier is produced. On the horizontal axis we plot the percentage of the entire data set returned, on the vertical axis the approximate percentage of the positives that is found within that space. The one-class classifier that is used in the mapping phase produces the first (unconnected) point in the plot. After this we construct the curve, starting from the right-top with the mapping hypothesis (i.e. selecting the objects furthest from the one-class hypothesis) and moving left and down with each step.

The resulting curve can be interpreted in a ROC-like fashion. The upper left corner represents a perfect classifier, points on the diagonal are random assignments. On each

iteration step of MC, a classifier is generated that gives a point on a curve like figure 5.3.f. The first point in the convergence will be close to the upper right corner: the mapping stage is about selecting a small part of the data set containing only near-certain negatives. From there each iteration will lead to a smaller selection (i.e. it moves left), but potentially also a lower performance on the true positives (i.e. it moves down). A large step leftwards corresponds to a large step in the convergence: a lot of unlabeled data is identified as negative. A large step downwards signals a big loss in performance.

The key is to identify the point in the curve that rids us of most of the data, while keeping a large part of the positive data to be classified correctly. The point on the curve that is closest to $(0, 100)$ is the best classifier. In practice we have to decide on this in retrospect. The distance measure can be weighted to assign more importance to recall or precision, in this paper we will use the euclidean distance.

Note that, contrary to a genuine ROC curve, the performance curve is not continuous. As a result it is not possible to calculate the area under the curve (AUC), nor can we take any point on the connecting line pieces to be a candidate. Assuming that we can identify where to stop the convergence, a curve is as good as its best (with respect to some criterion) point.

Example 1: random data in \mathbb{R}^2 To better understand the dynamics of the convergence, we create random data in \mathbb{R}^2 that also does not offer the large gap between positive and negative data that will stop it. We create the following dataset: 75 positives and 1225 unlabeled data points, of which 1000 are negatives and 225 are noise. We randomly generate clusters of positive data, surrounded by negative data. Some noise is added by switching the classification of 1% of the data. We use 80% of the data selected randomly for training and the remaining 20% for testing. Different ratios between the numbers of positives and negatives give similar results.

The classification of this data is shown in figure 5.3. The first image displays the distribution of negatives (dots), unlabeled positives (diamonds) and labeled positives (circles). Remark that the input for the algorithm only includes information about the labeled positives, as can be seen in the second image where all unlabeled data is represented as dots. The resulting ROC-like curve shows how the hypotheses are performing on separating out a small proportion of the entire data set, while maintaining performance on the positive subset.

In the upper right corner of figure 5.3.f, the mapping stage (iteration 0) creates a conservative hypothesis excluding only a small proportion of the unlabeled set. This corresponds to the decision boundary in figure 5.3.b. From that the convergence proceeds, with each iteration decreasing the number of unlabeled items that are selected. The performance on the positive set starts out to be good, until the point where overfitting takes place. This corresponds to the situation depicted in the 5.3.e. If we stop the convergence on the fifth iteration, giving us the classifier closest to the upper left corner

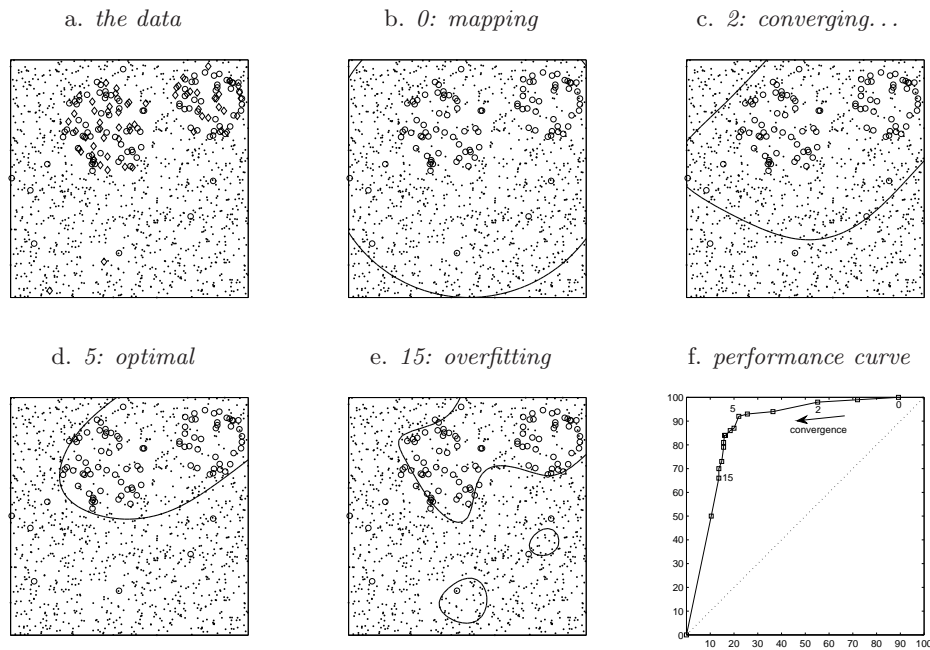


Figure 5.3: Random data in \mathbb{R}^2 : spacial representation of convergence and resulting performance curve.

of the plot, we obtain a fairly accurate description of the data.

Example 2: reference corpora In this section we will reproduce some experiments from [51, 52] to better understand the algorithm and to see how this behaviour is expressed using the performance curves we defined above. We look at what happens when positive data is undersampled by removing positive data and adding noise. Furthermore, for the linear kernel we examine the influence of reducing the number of features and for the Gaussian kernel we experiment with setting the γ parameter.

We use two reference corpora: the INEX 2008 XML mining corpus [2], containing text files with subject class labels, and the Letter recognition data set [42] from the UCI Machine Learning Repository [4], containing samples of handwriting classified with the correct symbol. The INEX08 corpus contains 11.437 objects described by 78412 features representing a bag-of-words representation with tf-idf values. The vectors are very sparse (at an average of 103 non-zero values per document), so we will use the linear kernel. We use classes 8 and 6 as positive, while all others are negative. The Letter recognition data set contains 20.000 instances, described by 16 features. Since these vectors are not sparse, the Gaussian kernel will be the best option. We use classes 0, 1, 2, 3 and 4 as positives. In both cases about 20% of the data is now labeled as positive.

For our experiments we use 20% of the data for testing. The remaining 80% of negatives is always included as unlabeled data. Of the positives we include specific

part as training data and another part as noise. Potentially this leaves some positive instances unused. To keep this discussion concise we will put the performance curves of our experiments in Appendix B.

In [52] it is noted that the algorithm tends to over-iterate when positive data is undersampled. To test this claim, our first experiments classify the data with different amounts of the positives as training data and without any noise. With both corpora the effect is clear (cf. figures B.1 and B.4): removing training data, even with no noise in the unlabeled data, severely hurts the performance. With less data steps are bigger and the curve in extreme cases even reaches (0,0). A potential explanation can be found in the performance of the OC-SVM that is used in the mapping phase: removing data moves the first unconnected point significantly towards (0,0).

Yu also claims that the addition of noise barely has any effect on the performance of the framework. As can be seen in figures B.2 and B.5, this is correct. Continuing with only 12,5% of the positive data for training, adding equal parts of noise does not make a big difference. We added up to 7 times more noise than trainings data (i.e. the remaining 87,5% of all positives) without clearly hurting the performance.

In both cases we now continue with 12,5% of the positives as training data and 50% as noise (ratio 1/4). For the INEX08 corpus, with the linear kernel, we test if removing features has a positive effect on the performance (cf. figure B.3). Actually we see that a reduction can be helpful (10.000 features) but if too much is removed performance deteriorates, ending with the overfitting and erratic results with 100 features. In the next chapter we will see the influence the number of features has for our attempts to classify the Enron corpus.

The other corpus is used with the Gaussian kernel, so we have an additional parameter γ (cf. figure B.6). It turns out that the best value for this data set is around 0.01. Larger values give larger steps, and ultimately an immediate jump from the mapping phase to the endpoint in (0,0). To small values for γ result in overfitting and erratic behaviour. Both observations comply with our intuitions about γ controlling the smoothness of the hypothesis. Apart from the two mentioned phenomena, the result for different values for this parameter don't change much for the overall result.

5.4 Generating hypotheses from unlabeled data

When applying the framework described above to the case of Enron, we encounter a serious problem: how to estimate the percentage of positive returned by each hypothesis when no labels are available for the largest part of the data set. To calculate the percentage of data that is returned, we can simply leave out part of the unlabeled data in the training phase and use a prediction on that to estimate the performance. Because we have only 186 positive training examples (whereas the entire data set contains over 250.000 items) and all of them represent a distinct part of the search space (legal experts

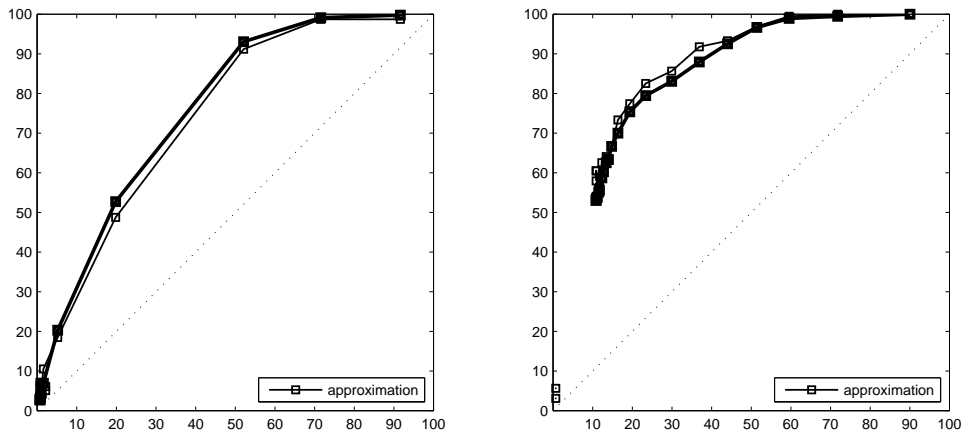


Figure 5.4: Approximation of performance with cross-validation. Left for the INEX08 corpus (linear kernel), right for the Letter recognition data set (RBF kernel).

have selected exactly this set to present the evidence) leaving out a part here would almost certainly hurt performance.

As a solution, we introduce a cross-validation step in the algorithm. Each step in Mapping Convergence process will now be carried out with a 5-fold split over the positive data. We train a hypothesis on 4 parts, and predict on the remaining fifth part and the entire unlabeled set. This results in exactly one prediction per item in the positive set, and after aggregating the five predictions for the items in the unlabeled set we get a single prediction there too. This allows us to obtain a solid estimate of the performance of the hypothesis.

For the two corpora introduced in the previous section we tested this approach by comparing the curves generated by the cross-validation approach with the curves that are generated in the same run of the experiment by testing on a part of the data that has been held out for that purpose. As can be seen in figures 5.4 the approximation is quite satisfactory: the approximation closely follows the results measured with a held-out test set.

Note that the introduction of a random five-way split in all stages of the convergence introduces some irregularities in the outcome of the algorithm. Using a higher cross-validation will solve this, but comes with a (computational) cost.

5.5 Combining classifiers

Now that we have a reliable way to estimate the performance of a classifier, even when very little labeled data is available, we can start looking at an extension that will allow for multiple views of the data to be combined. In chapter 4 we defined two complementary

views of documents in the Enron Corpus, one based on the text and the other on the implicit social network. The intuition is that a judgement based on the two should be more reliable than each one separately. In this section we propose to combine the MC algorithm with the idea of co-training, but first let's take a look at a more naïve way of combining predictions.

Combining different classifiers to improve overall performance is known as Ensemble Learning [14]. Essentially, the combination of multiple imperfect classifiers results in a superior performance. In our interpretation of the MC algorithm as producing a sequence of classifiers, we can generate predictions on a test set on each of the steps. In fact this is exactly what we will do to determine the position in the performance plane, using cross-validation for the positive examples.

Now suppose that we take classifiers based on different views. Both have classified all items in the test sets, but potentially have made errors. The idea is that when one of the two has made an error, it can be corrected by the second. Since we obtain for each prediction a number in $[-1,1]$ that represents the classification and confidence, we can average these predictions over multiple classifiers. The classifier that is most certain will 'win' and, in case of an error, correct the other.

An improvement of the performance will be easily recognized as a movement up or left (or both). When we move left in the performance plane this means that we select a smaller part of the data set, while a movement up can be interpreted as an increment in the percentage of positives that is retrieved. As we will see when we discuss the results of our experiments in chapter 6, combining classifiers in this way results in exactly these effects.

If we can establish that combining the predictions of multiple classifiers representing different modalities does indeed aid classification, it would be interesting to see if we can adapt the MC algorithm to take the different views into account on each of the steps. We will have different classifiers cooperating in a way that closely resembles the concept of co-training [6]. On each step of the iteration the more confident classifier will be able to overrule the other. Predictions are aggregated and a fixed percentage of the items is labeled as negative. The algorithm we propose to implement this idea is depicted as algorithm 2. Note that we are not necessarily labeling all data until convergence, only part of the data is needed: the ones that both classifiers agree on to be negative.

There are three differences with the original algorithm discussed in section 5.2. First of all we start not with one representation for the data items, but with n different views of them. An intuitive way to view the relation with the previous version is that we are processing n different convergences simultaneously. The interaction takes place only by means of the aggregation function, the second addition, that combines the predictions and thus creates a filter that can be used to select the items to label. A final difference is that in the convergence phase, not all items that are recognized as negative are added to the N . Only the ones that are agreed upon with the highest certainty are labeled,

Algorithm 2 Mapping Co-Convergence**Require:**

- n views of the positive data set $P^{(1)}, \dots, P^{(n)}$
- n views of the unlabeled data set $U^{(1)}, \dots, U^{(n)}$
- n views of the negative data set $N^{(1)} = \emptyset, \dots, N^{(n)} = \emptyset$
- OC-SVM: C_1
- SVM: C_2
- Aggregation function: Agg.

Ensure: boundary functions $h_i^{(1)}, \dots, h_i^{(n)}$

- 1: $h_0^{(k)} \leftarrow \text{train } C_1 \text{ on } P^{(k)} \quad \forall k \in [1, \dots, n]$
- 2: $\text{pred}_0^{(k)} \leftarrow \text{predict with } h_0^{(k)} \text{ on } U^{(k)} \quad \forall k \in [1, \dots, n]$
- 3: $\hat{N}_0^{(k)} \leftarrow \text{strong negatives } (\leq 10\%) \text{ in } U^{(k)} \text{ by Agg}(\text{pred}_0^{(0)}, \dots, \text{pred}_0^{(n)})$
- $\hat{P}_0^{(k)} \leftarrow \text{remaining part of } U^{(k)} \quad \forall k \in [1, \dots, n]$
- 4: $i \leftarrow 0$
- 5: **while** $\hat{N}_i^{(k)} \neq \emptyset \quad \forall k \in [1, \dots, n]$ **do**
- 6: $N^{(k)} \leftarrow N^{(k)} \cup \hat{N}_i^{(k)} \quad \forall k \in [1, \dots, n]$
- 7: $h_{i+1}^{(k)} \leftarrow \text{train } C_2 \text{ on } P^{(k)} \text{ and } N^{(k)} \quad \forall k \in [1, \dots, n]$
- 8: $\text{pred}_{i+1}^{(k)} \leftarrow \text{predict with } h_{i+1}^{(k)} \text{ on } \hat{P}_i^{(k)} \quad \forall k \in [1, \dots, n]$
- 9: $\hat{N}_{i+1}^{(k)} \leftarrow \text{strong negatives } (\leq 5\%) \text{ in } \hat{P}_i^{(k)} \text{ by Agg}(\text{pred}_{i+1}^{(0)}, \dots, \text{pred}_{i+1}^{(n)})$
- $\hat{P}_{i+1}^{(k)} \leftarrow \text{remaining part of } \hat{P}_i^{(k)} \quad \forall k \in [1, \dots, n]$
- 10: $i \leftarrow i + 1$
- 11: **end while**

limited to 5% of the entire data set.

In the next chapter we will see the results of classifying Enron with the MC algorithm proposed in this chapter. Also we will see the results of combining multiple classifier using both methods described in this section.

Part III

Results

Chapter 6

Results and Analysis

6.1 Implementation

We implement our framework in Python, and use the LIBSVM library [10]. As briefly discussed in section 4.5 we use the linear kernel for the text-based feature sets and the Gaussian kernel with the social network-based sets. An important parameter for both is ν , that bounds the number of margin errors or outliers. By picking a small value, we ensure that the algorithm is forced to come up with algorithms that stay true as much as possible to the training data we gave it. We pick $\nu = 0.1$ for all experiments. For the Gaussian kernel we also have the γ parameter, controlling the smoothness of the decision boundary, that we will discuss later.

Although Support Vector Machines are computationally not very expensive, for practical reasons we have decided to run our experiments on a subset of the corpus. In all experiments we use the same sets of 186 positive examples obtained from the DOJ exhibit list and 10,000 randomly selected unlabeled documents. For testing on the positives we implement 5-fold cross-validation, for testing on unlabeled items we hold out 500 items. This way we can construct the performance curves as discussed in the previous chapter.

LIBSVM can be set to produce probability estimates for each of the classes. These are based on a method by Platt [36, 46] that fits a logistic function to the output of an SVM to generate posterior probabilities. This algorithm assumes that the distribution of positive and negative data points in training and test sets are equal. Crucially this is not the case in a one-class setting, nor in the convergence steps where the distribution in the training set actually changes on each step.

Alternatively we slightly alter the code of LIBSVM to directly output the distance to the decision plane. This indeed is a measure of the confidence of the prediction. After scaling to $[-1,1]$ we can aggregate the predictions of multiple classifiers, in this case by taking averages. A more advanced way of capturing the confidence of predictions might be subject of future research.

6.2 Classifying Enron

In this section we present our results classifying the Enron corpus using the two document representations discussed in chapter 4.5. We try to get the optimal setting of parameters to obtain good classifiers for use in the next section, where we will try to combine their predictions.

We present the performance curves of the best classifiers for each type of features. The accompanying results that are discussed in the text are in appendix C.

Text-based representations For the text-based representations, we first of all have to decide what type of feature values to use. It turns out that both binary and tf-idf values give similar results, the latter being slightly better and more stable (cf. C.1).

Because we use a one-class classifier, the dimension of the feature space might be important (for reasons discussed in chapter 4.5). Even though with our reference corpus we had a slight increase in performance by reducing the number of features, no such thing happens in this case (cf. C.2). We even observe that the OC-SVM performs considerably worse with less features, but the convergence itself is not hurt. Also the reduction of features using semantic clustering does not seem to improve anything (cf. C.3).

Overall best and most stable performance is obtained using all features with tf-idf feature values. The performance curve is shown in figure 6.1. We can see that during the convergence performance degrades slowly, with a drop at the end. The key is to select a classifier that is just before the drop. Note also that the algorithm is clearly beating OC-SVM. The algorithm takes a huge first step in the convergence, yielding a hypotheses that separates out 75.8% of the positives in 16.8% of the data.

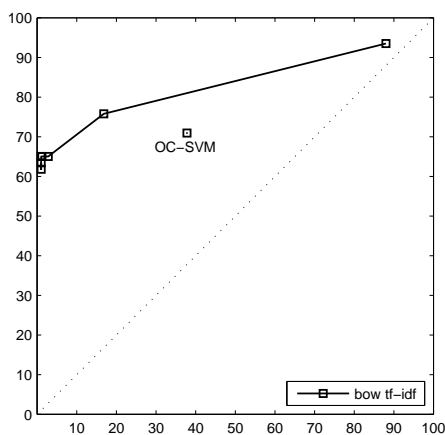


Figure 6.1: Enron: text-based.

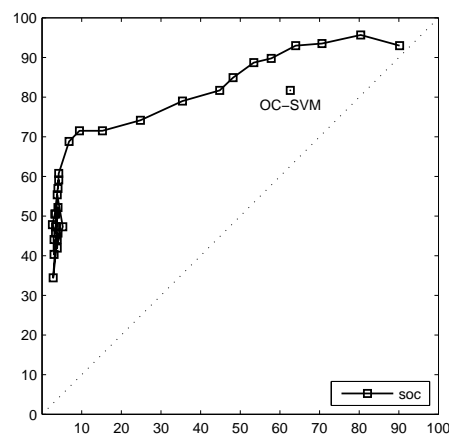


Figure 6.2: Enron: social network-based.

Social network-based representations Because of the different nature of the features used in the document representation based on the implicit social network, we have a lot less tuning to do. The feature values are fixed and reducing the number of features is also not an option. However, with the RBF kernel we have an additional γ parameter. Our observations largely correspond with what we noticed varying the value of this parameter for the Letter recognition data set in the previous chapter. The optimal value that we have found is 0.1. Bigger values lead to underfitting: large steps in the convergence. Smaller values lead to underfitting: good performance until a certain point, and erratic behaviour thereafter.

The best and most stable performance is obtained with $\gamma = 0.1$. The performance curve is shown in figure 6.2. Optimally we select 71.5% of the positives in 9.4% of the data.

6.3 Combining classifiers

In this section we present the results of combining the classifiers from the previous section. The two methods are described in detail in section 5.5. It turns out that in both cases best results are obtained using the bag-of-words representation with all features and tf-idf values with the social network feature set.

Naïve method The simplest way of combining two classifiers is to take their predictions, aggregate those (in some way) and measure the performance. This is exactly what we are doing in this approach. For an aggregation function we simply take the average of

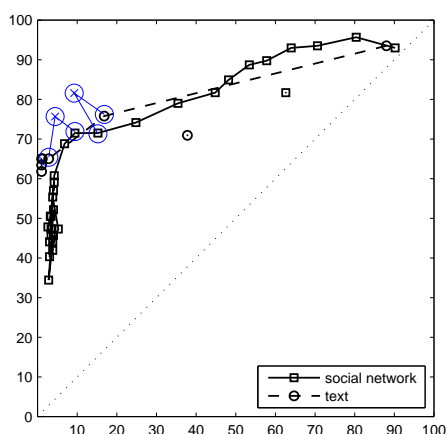


Figure 6.3: Enron: naïve combination of classifiers (examples).

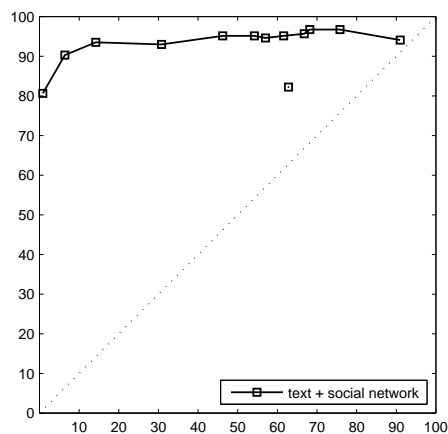


Figure 6.4: Enron: combination of classifiers using the Mapping Co-Convergence framework.

classifier	% data	% pos	distance
text (bow tf-idf)	75.8	16.8	29.5
social network	71.5	9.4	30.0
text + social (naïve)	81.6	9.2	20.6
text + social (MCC)	90.3	6.4	11.6

Table 6.1: Comparison of the different classifiers.

the predictions. This, of course, could be a point for improvement but we leave that to future research.

See figure 6.3. By means of example we combined the classifiers we get at the 12th and 13th steps on the social network curve with the 2nd and 3rd steps on the text curve respectively. The fact that we observe a movement towards the left-top means that we get a classification that takes less data while retrieving more of the positives. There is something to be gained by combining the predictions based on different representations of documents.

Mapping Co-Convergence The idea of this approach is to combine the different views on the data on each step of the conversion. Every time a limited number of objects that is classified as negative with the highest certainty by the two classifiers combined is labeled. The hypotheses appearing on the curve are based on a small part of the data. The performance is excellent, staying above 90% recall of the positives while discarding over 90% of the data.

As explained before, we can compare the curves by comparing their “best” classifier. We take the Euclidian distance to (0,100), the perfect classifier, to be the measure of comparison. Of course we can define other performance measure that take into account that we find recall more important than precision or vice versa. In table 6.1 are listed the best classifiers of the curves discussed in this chapter. We can see that the combination of social network-based and text-based feature sets does indeed yield very good results.

Chapter 7

Conclusion

7.1 Discussion

From the results presented in the previous chapter we can conclude that our approach to document classification by using multiple levels of description does yield excellent results. Implementing ideas from co-training within the Mapping Convergence framework has enabled us to dramatically improve classification results. First we discuss some crucial observations with respect to our framework.

Initialization of Mapping Convergence It appears that the mapping phase of the framework is crucial for a good result. Any bad classifications made will have a snowball effect and impede performance severely. It is however necessary to include enough data in the first approximation of the negative set to support convergence. From our experiments it seems that labeling 5-10% with a one-class classifier is usually enough to keep the convergence going.

Dependency on parameter selection Even though the number of parameters is not huge, the algorithms are somewhat sensitive to the selection of parameters. Some of them we have been able to select by looking at its properties, for others however this is not so easy to do. As can be seen from the test results on the reference corpora their influence is sometimes quite substantial.

Instability as a result of random cross-validation We introduced a cross-validation step to be able to work with a corpus which does not contain labels for much of the data. This is however another source of instability. The split is randomly made on every step in the convergence. After several runs we have picked average results, but there sometimes was a discrepancy of about 10% between runs. This could be reduced by using 10-fold crossvalidation or higher, at a computational cost.

Lack of training data From our tests on INEX08 and the Letter recognition data set it appears that the size of the initial positive set is of capital importance. Reducing the number of positive training examples has a striking effect on the overall results. In Enron we have a very small set of positively labeled items, that we used with a random subset of the rest of the corpus. Using it with the entire corpus might cause the unlabeled data to flood the positives after all. It might be that the combination of different views in Mapping Co-Convergence partly handles this problem, but that has to be verified.

Confidence measure for SVM Research on ways to assign a confidence measure to the predictions of Support Vector Machines is an active field. A well-known approach by Platt [36] is actually implemented in the LIBSVM toolkit [10] we used. Unfortunately, because of an assumption made by this approach it is not applicable in our case. We decided to use a scaled version of the distance to the decision plane as an indicator of confidence. This is definitely a point that can be improved.

Aggregation function Also the way we combine the prediction of different classifiers is quite naïve. In fact we take the average of the predictions. It is clear that this is an approximation and lacks solid theoretical support, but future research is necessary to come up with a better solution.

Verification on an annotated corpus In spite of the fact that cross-validation offers a fairly good approximation of results, we would still like to replicate the success of our Mapping Co-Convergence framework on a fully annotated corpus. However we depend on a classification target that justifies the use of multiple views. Also we would need a corpus that does have more than one complementary annotations available.

7.2 Future research

Our approach has displayed good results, but can be improved in many ways. We would like to list some of the opportunities here.

Additional views We use the textual contents and the implicit social network in the corpus as bases for our document representations. It would be interesting to see if other descriptions would be good candidates. One could think of things like the visual layout, attachments, etcetera. Theoretically adding more views would ever increase the power of the framework we defined.

DOJ data set Given the economical interest of research related to document review and the lack of any publicly available data sets that represent a classification towards responsiveness, we believe that the DOJ data set that we have introduced might prove useful for further research.

Applications to document review In document review technology the general strategy is to present documents in a natural order to the user to improve human performance. For example, usually potentially responsive documents are grouped together. Our approach has showed to be capable of accurately identifying a small part of a large data set containing almost all positives.

In that respect it would also be interesting to see if it is possible to use the additional input and corrections from users during the review process. For that we would have to look at ideas from active learning.

7.3 Main conclusions

Returning to the subquestions that we formulated in the introduction we reach the following conclusions:

- From the exhibit lists that are published by the U.S. Department of Justice, we obtain the e-mails that have been used in a particular trial. This proves to be a good basis for a classification towards responsiveness.
- We have constructed a representation of e-mail documents based on the social network that is implicit in the exchange of communication. Social network features of senders and receivers, that encode the role of the actor in the organization, can be used to classify e-mail. The principle has already been applied to detect the position of these actors in the hierarchy, but its application to document classification is, for as far as we are aware, a novelty.
- Our alternative evaluation method of the result of Mapping Convergence allows us to apply it to the more general case where no natural gap between positive and negative data exists. Using cross-validation we developed an approach to approximate performance when only a very small set of labeled (positive) data is available.
- We have extended the original MC framework using the concept of co-training. Combining different views and using them to generate predictions as to what is to be labeled as negative turns out to yield excellent results.

With respect to our main research question, whether we can improve classification performance in a one-class setting by combining classifiers of different modalities, we can answer affirmatively. A combination of social network-based features with a more traditional text-based representation improves overall performance in classification towards responsiveness. Especially the incorporation of co-training in the Mapping Convergence framework seems promising.

Bibliography

- [1] Enron: The smartest guys in the room, 2005. Jigsaw productions.
- [2] 2008 INEX XML Mining corpus. 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, 2008.
- [3] Julien Ah-Pine and Guillaume Jacquet. Clique-based clustering for improving named entity recognition systems. In *EACL*, pages 51–59, 2009.
- [4] A. Asuncion and D.J. Newman. UCI machine learning repository.
- [5] Ron Bekkerman, Andrew McCallum, and Gary Huang. Automatic categorization of email in folders: benchmark experiments on Enron and SRI corpora. Technical Report IR-418, CIIR, 2004.
- [6] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1998.
- [7] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
- [8] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.
- [9] Christopher J. C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for Support Vector Machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [11] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. A tutorial on ν -Support Vector Machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.
- [12] William Cohen. <http://www.cs.cmu.edu/~enron/>, March 2004.
- [13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [14] Thomas G. Dietterich. Ensemble methods in Machine Learning. In J. Kittler and F. Roli, editors, *Proceedings Multiple Classifier Systems, 2000*, volume 1857, pages 1–15. Springer Verlag, 2000.

- [15] Tamer Elsayed and Douglas Oard. Modeling identity in archival collections of email: a preliminary study. In *Conference on Email and Anti-Spam*, 2006.
- [16] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In A. Abraham, B. Baets, M. Koppen, and B. Nickolay, editors, *Advances in Soft Computing : Applied Soft Computing Technologies: The Challenge of Complexity*, pages 425–438. Springer Verlag, 2006.
- [17] Federal Energy Regulatory Commission. The western energy crisis, the Eron bankruptcy, and FERC’s response. <http://www.ferc.gov/industries/electric/indus-act/wec/chron/chronology.pdf>.
- [18] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7(2):3–12, 2005.
- [19] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 99, pages 7821–7826, 2002.
- [20] David Gleich. MatlabBGL: a Matlab graph library. http://www.stanford.edu/~dgleich/programs/matlab_bgl/, 2008.
- [21] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction (Springer Series in Statistics)*. Springer New York, 2 edition, 2009.
- [22] Jeffrey Heer. Exploring Enron: Visual data mining of e-mail. <http://www.jheer.org/enron/>, 2005.
- [23] Thorsten Joachims. Text categorization with Support Vector Machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag.
- [24] Thorsten Joachims. A statistical learning model of text classification for Support Vector Machines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 128–136. ACM Press, 2001.
- [25] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [26] Larry M. Manevitz and Malik Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.
- [27] Andrew McCallum, Xuerui Wang, and Andres Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research*, 30:249–272, 2007.
- [28] David Meyer, Friedrich Leisch, and Kurt Hornik. The Support Vector Machine under test. *Neurocomputing*, 55:169–186, 2003.
- [29] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

- [30] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
- [31] Andrew Ng. Machine Learning. Lecture notes with course CS229, Stanford University.
- [32] Jacki O’Neill, Caroline Privault, Jean-Michel Renders, and Gregory Bauduin. Disco: Intelligent help for document review. Workshop DESI at 12th International Conference on Artificial Intelligence and Law, June 2009.
- [33] Shaheen Pasha and Jessica Seid. Lay and Skilling’s day of reckoning. http://money.cnn.com/2006/05/25/news/newsmakers/enron_verdict/, May 25 2006.
- [34] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864–881, 2009.
- [35] Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an ensemble of One-class SVM classifiers to harden payload-based anomaly detection systems. In *ICDM ’06: Proceedings of the Sixth International Conference on Data Mining*, pages 488–498, Washington, DC, USA, 2006. IEEE Computer Society.
- [36] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
- [37] Bhavani Raskutti and Adam Kowalczyk. Extreme re-balancing for SVMs: a case study. *SIGKDD Explorations*, 6(1):60–69, 2004.
- [38] Ryan Rowe, German Creamer, Shlomo Hershkop, and Salvatore J. Stolfo. Automated social hierarchy detection through email network analysis. In *WebKDD/SNA-KDD ’07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 109–117, New York, NY, USA, 2007. ACM.
- [39] Bernhard Scholkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.
- [40] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- [41] Jitesh Shetty and Jafar Adibi. Discovering important nodes through graph entropy the case of Enron email database. In *LinkKDD ’05: Proceedings of the 3rd international workshop on Link discovery*, pages 74–81, New York, NY, USA, 2005. ACM Press.
- [42] David J. Slate. Letter recognition data set. <http://archive.ics.uci.edu/ml/datasets/Letter+Recognition/>.
- [43] Steven H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.
- [44] David M. J. Tax. *One-class Classification*. PhD thesis, Delft University of Technology, The Netherlands, June 2001.

- [45] David M. J. Tax and Robert P. W. Duin. Support Vector Domain Description. *Pattern Recogn. Lett.*, 20(11-13):1191–1199, 1999.
- [46] Hsuan tien Lin, Chih jen Lin, and Ruby C. Weng. A note on platts probabilistic outputs for support vector machines. Technical report, 2003.
- [47] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. In Marleen Huysman, Etienne Wenger, and Volker Wulf, editors, *Communities and Technologies*, pages 81–96. Kluwer, Deventer, The Netherlands, The Netherlands, 2003.
- [48] United States Department of Justice. Enron trial exhibits and releases. <http://www.usdoj.gov/enron/>, 2006.
- [49] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, 1999.
- [50] Jen-Yuan Yeh and Aaron Harnly. Email thread reassembly using similarity matching. In *Conference on Email and Anti-Spam*, 2006.
- [51] Hwanjo Yu. Single-class classification with Mapping Convergence. *Machine Learning*, 61(1-3):49–69, 2005.
- [52] Hwanjo Yu, ChengXiang Zhai, and Jiawei Han. Text classification from positive and unlabeled documents. In *CIKM*, pages 232–239, 2003.
- [53] Lin Zhuang and Honghua Dai. Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of computers*, 1(7):32–40, 2006.

Appendices

Appendix A

Database schema

A.1 Tables

We construct a database from the data from the Enron data set published by William Cohen [12]. All data is in the six tables displayed in white (file2email, message, correspondent, id_evidence, words, freq). Only the first four are concerned with representing the corpus. The latter two contain the lexicon.

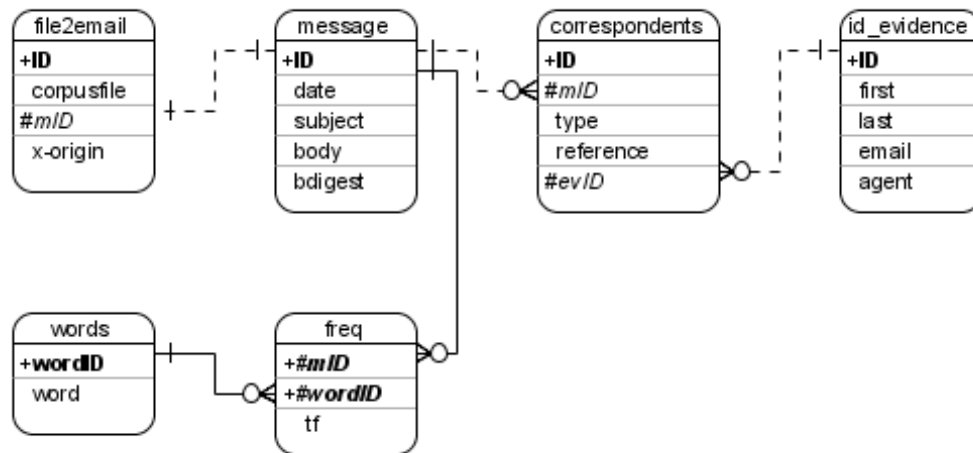


Figure A.1: Database schema of Enron DB.

file2email contains an ID, the original datafile (corpusfile), a reference to the corresponding message (mID), a header field added by Cohen containing a reference to the account it comes from (x-origin).

message contains the actual messages. An ID, the date (normalized for timezones), subject, body and an MD5 digest of the body (bdigest).

correspondent contains the information from To/From/etc fields. An ID, a reference to the message (mID), type (TO/FROM/CC/BCC), the reference string (i.e. the email address/name), a reference to the reconstructed evidence (evID).

id_evidence contains reconstructed identity evidence: an ID, first name (and/or initial(s)), last name, email address and the agent ID. Each row represents a reference in one message (potentially by multiple strings, e.g. Mark Fisher and mf@enron.com in one message). The agent field links several rows of this table together, so all Mark Fishers in different messages refer to the same agent.

A.2 Views

We also define four views to simplify access to the data.

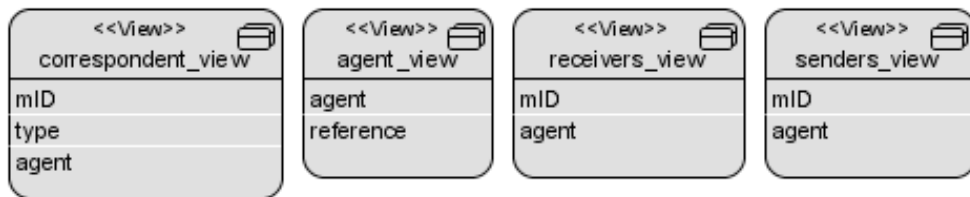


Figure A.2: Database schema of Enron DB: views.

correspondent_view shows all correspondents (agents) per message.

agent_view shows the references that are linked together as an agent.

senders_view shows only senders.

receivers_view shows only receivers.

Appendix B

Reference corpora: classification results

B.1 INEX 2008 XML mining corpus

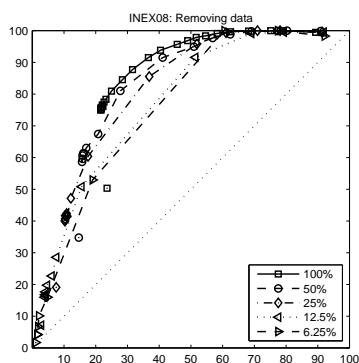


Figure B.1: We consider an artificial division of classes 8 and 6 as positive (21%), and the rest as negative. In all cases we use 80% of the data to train a hypothesis and the remaining 20% to test it. This first experiment aims to show the influence of removing positive training data. We start with all positive data as known positives and all of the negatives as unlabeled items. We then remove part of the positives (50%, 25%, 12.5% and 6.25%) and observe that performance degrades: larger steps and conversion to (0,0) in the extreme cases.

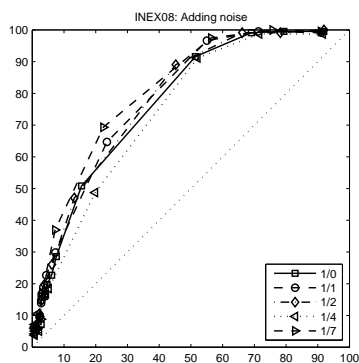


Figure B.2: Here we continue the experiment by looking at the influence of adding noise to the unlabeled set. We start out with 12,5% of the positives and all of the negatives as unlabeled (1/0). We then add more and more noise by putting unused positives in the unlabeled set as noise, up until 7 times the size of the positive training set as noise. Performance stays quite stable.

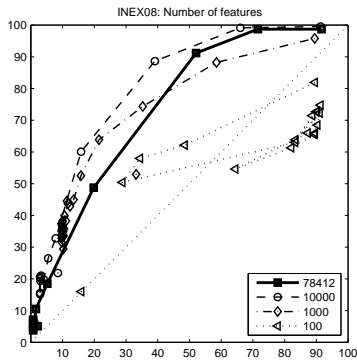


Figure B.3: This image aims to show the influence of reducing the number of features. We remove words with low document frequencies. We perform the experiment with 12.5% of the positives and 50% as noise (ratio 1/4). Some reduction seems to be useful (10.000 features), but too much reduction leads to overfitting (100 features).

B.2 Letter recognition data set

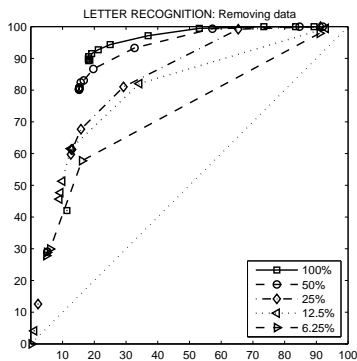


Figure B.4: We consider an artificial division of classes 0, 1, 2, 3 and 4 as positive (20%), and the rest as negative. In all cases we use 80% of the data to train a hypothesis and the remaining 20% to test it. Again, this first experiment aims to show the influence of removing positive training data. We start with all positive data as known positives and all of the negatives as unlabeled items. We then remove part of the positives (50%, 25%, 12.5% and 6.25%) and observe that performance degrades: larger steps and conversion to (0,0) in the extreme cases.

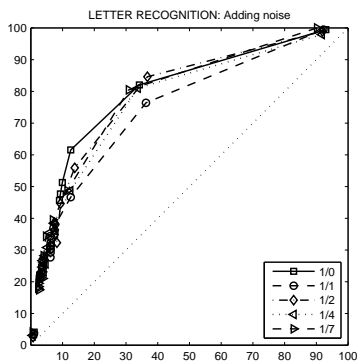


Figure B.5: Here we continue the experiment by looking at the influence of adding noise to the unlabeled set. We start out with 12,5% of the positives and all of the negatives as unlabeled (1/0). We then add more and more noise by putting unused positives in the unlabeled set as noise, up until 7 times the size of the positive training set as noise. Performance stays quite stable.

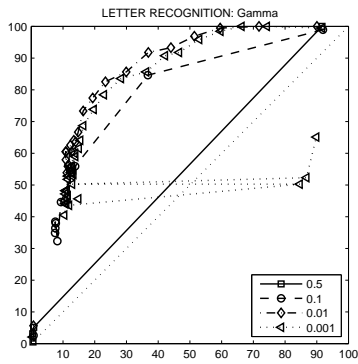


Figure B.6: This final experiment aims to show the influence of the γ parameter on the framework in case of a Gaussian kernel. We perform the experiment with 12.5% of the positives and 50% as noise (ratio 1/4). Large values for γ give larger steps, eventually jumping to (0,0) immediately. Small values give overfitting, evidenced by erratic behaviour. The optimal value has to be determined experimentally.

Appendix C

Enron Corpus: classification results

C.1 Text-based features

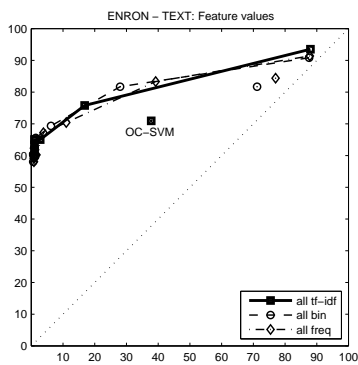


Figure C.1: The performance on the Enron corpus with different feature values. For all three performance is similar. A huge step is taken in the first iteration. From our experiments it seems that tf-idf is a little more stable.

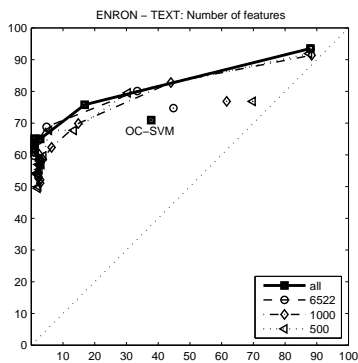


Figure C.2: Classification of the Enron corpus with varying numbers of features. Contrary to our experiment on the INEX08 corpus, here we see no improvement with reducing the number of features.

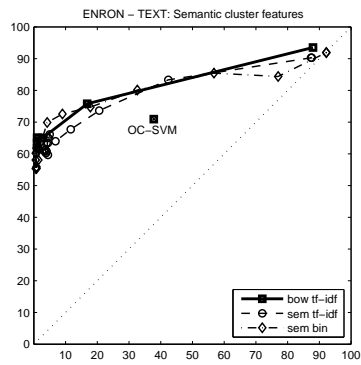


Figure C.3: Comparing performance with feature vectors constructed using the semantic clustering with the bag-of-words representation. The only obvious improvement is the fact that smaller conversion steps are taken.

C.2 Social network-based features

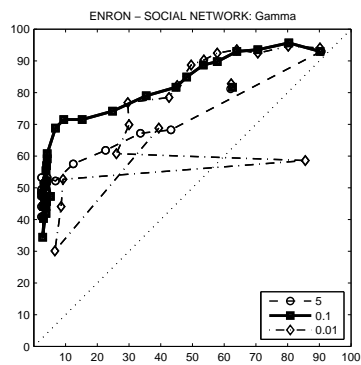


Figure C.4: Using the RBF kernel, the γ parameter plays an important role. We find that 0.1 gives the best performance in this case. Larger values lead to underfitting (i.e. large steps), smaller values lead to overfitting (i.e. erratic behaviour).

List of Figures

2.1	Bias-variance trade-off	13
2.2	Structural Risk Minimization	14
3.1	Maximizing the margin.	16
4.1	Characteristic plots of the resulting data set	28
5.1	Artificial data in \mathbb{R}^2	36
5.2	An illustration of MC in 1-dimensional space.	37
5.3	Random data in \mathbb{R}^2	40
5.4	INEX08, Letter recognition: approximation.	42
6.1	Enron: text-based.	48
6.2	Enron: social network-based.	48
6.3	Enron: naïve combination.	49
6.4	Enron: MCC	49
A.1	Database schema of Enron DB.	61
A.2	Database schema of Enron DB: views.	62
B.1	INEX08: removing data	63
B.2	INEX08: adding noise	63
B.3	INEX08: reducing number of features	64
B.4	Letter recognition: removing data	64
B.5	Letter recognition: adding noise	64
B.6	Letter recognition: gamma parameter	65
C.1	Enron: feature values	67
C.2	Enron: number of features	67
C.3	Enron: semantic clustering	68
C.4	Enron: gamma for RBF	68