Multi-modality in One-class Classification

Matthijs Hovelynck
Cognitive Artificial Intelligence
Utrecht University, The Netherlands
mhovelynck@gmail.com

Boris Chidlovskii
Xerox Research Centre Europe
6, chemin de Maupertuis, Meylan, France
chidlovskii@xrce.xerox.com

ABSTRACT

We propose a method for improving classification performance in a one-class setting by combining classifiers of different modalities. We apply the method to the problem of distinguishing *responsive documents* in a corpus of e-mails, like Enron Corpus. We extract the social network of actors which is implicit in a large body of electronic communication and turn it into valuable features for classifying the exchanged documents. Working in a one-class setting we follow a semi-supervised approach based on the Mapping Convergence framework. We propose an alternative interpretation, that allows for broader applicability when positive and negative items are not naturally separable. We propose an extension to the one-class evaluation framework in truly one-case cases when only some positive training examples are available. We extent the one-class setting to the *co-training* principle that enables us to take advantage of multiple views on the data. We report evaluation results of this extension on three different corpora including Enron Corpus.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Applications**]: Data Mining; H.4.3 [**Communications Applications**]: [Electronic mail]

General Terms

Algorithms, Measurement, Performance

Keywords

Enron, one-class classification, responsiveness, co-training

1. INTRODUCTION

In a world where information becomes available in ever increasing quantities, document classification plays an important role. Obvious applications range from search engines to spam filtering, but also more specific tasks can be approached with the same techniques. In this paper we address a particular problem when the documents are reviewed 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* documents (i.e. that could be of importance to the case). The number of documents under review could easily run in the millions.

The review of documents has to be done by legal experts and is very time-consuming. Even for human annotators the accuracy is not very high; moreover an important mismatch usually exists between annotations by different people. It turns out that both speed and accuracy of reviewers can be improved dramatically by grouping and ordering documents.

Different technologies have been developed to support human annotators that mine the corpus structure and present documents in a natural order [3, 13]. Some take into account the textual contents of documents only. Obviously, other levels of description, such as the group of people that worked on the document or the visual layout, could be of importance in distinguishing relevant items. This idea could be applied in a multitude of different tasks, but classification towards responsiveness provides an ideal testbed since document review intuitively takes different criteria into consideration, not just the textual contents.

The publicly available 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. Besides coming from a community that has been a subject to corporate litigation, e-mails allow one to successfully construct an alternative representation of documents. Considerable structure may be hidden within a large body of e-mails from a community: the social network that is implicit in the group of people communicating one to another. In the following, we will use the term of *multi-modality* to refer to different levels of document description to aid classification tasks.

Considering responsive documents in the Enron Corpus, we target *improving 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 the same principles might be successfully applied to similar classification problems.

The contributions of this paper can be summarized as follows:

- Working in a one-class setting we adopt a semi-supervised approach based on the Mapping Convergence framework [31]. For broader applicability we propose an alternative interpretation that allows to size down the requirement on the natural separability of positive and negative items.

- We propose an extension to the one-class evaluation framework which turns to be useful when a small number of positive training examples are available and the ratio of positives is unknown. We use UCI and INEX08 datasets to prove its usefullness.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

- We extent the one-class setting to the co-training [4] that enables us to take advantage of the availability of multiple redundant views on the data. We evaluate the *Mapping Co-Convergence* on the responsiveness task in Enron Corpus.
- 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 show that a combination of text-based and social network-based modalities does improve classification results.

The paper is organized as follows. In Section 2 we introduce the responsiveness task on the Enron Corpus, then we report on the cleaning steps required to preprocess the data set. In Section 3 we present the text-based representation of documents, and one based on the implicit social network. Then in Section 4 we discuss the one-class setting and the Mapping Convergence principle. We extend the one-class evaluation framework and describe how to combine multiple modalities by co-training. In Section 5 we present a number of evaluation results obtained on the INEX08 and Letter Recognition datasets and the Enron Corpus. Section 7 enumerates the most important conclusions of our study.

2. THE ENRON CORPUS

Large bodies of documents, resembling the ones when reviewing for corporate litigation, are very rare to appear in the public domain. Both privacy-related and legal issues make often such collections, in the rare case that they are assembled at all, unpublishable. Consequently, the Enron Corpus 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 Enron Corpus opened up completely new possibilities to the research, including subject classification [18], folder detection [3], social network analysis [27], hierarchy detection [25], identity modeling [10], reconstruction of threading [30] and large (e-mail) network visualization [13].

Our approach is complementary to previous work in that we explicitly use the multi-modality of e-mails for classification. E-mails consist of text, but also implicitly instantiates a social network of people. We model these distinct levels and combine them 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.

There exist several versions of the Enron Corpus available online. Besides the raw text corpus [8], a relational database version is also available [27]. In this database, however, some important information regarding the social network has been discarded. This information is very important in our approach, thus we construct a database from scratch.

2.1 Responsive documents

When large corporations get involved in litigation and receive a subpoena to produce evidence regarding some case, they have to go through enormous amounts of data to obey the court order. The number of documents 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 considerable.

Different studies have shown that tools that group and order documents yield good results. In order to support human document review one would require a large set of pre-annotated data. Several attempts have been made to manually annotate fragments of the Enron Corpus [3]. All of them are relatively small and typically annotated with subject, emotional tone or other primary properties. Annotation with responsiveness is tedious and expensive, requires legal expertise and is usually not publicly available.

Our solution is to use the lists of government exhibits published on the website of the United States *Department of Justice* (DOJ) [1]. More specifically we use the set of e-mails used in the trials against the two CEO's of Enron, Ken Lay and Jeff Skilling; this set actually reflects the part of the corpus that has been found of interest.

From the DOJ we obtain a list of 186 e-mails. Being a very small set compared to the size of the entire corpus, we expect that the set covers most types of e-mails found to be relevant, yet we 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.

Working with the DOJ set means that we have to work in a one-class setting. The algorithm we will develop in the next sections 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.

2.2 Data cleaning

The Enron Corpus contains a large number of *duplicate* messages, ambiguous references to actors and other inconsistencies. The first problem is due to the automatic generation of the e-mail folders (e.g. *all documents*, *sent items*) and e-mails copies appearing both in sender's and receiver's folders. We unify identical messages using their titles and body digests.

The second problem is the existence of *ambiguities* and *inconsistencies* among the references in sender and receiver fields. To extract the implicit social network, we identify the references which are pointing to the same person. For example, 'Richard Buy', 'Rick Buy', 'Richard B. Buy', 'richard.buy@enron.com', 'rbuy@ect.enron.com', 'Buy-R', etc. probably all refer to the same person.

We use regular expressions to extract firstname, lastname and, when available, e-mail address from all references. Ee first reassemble references occuring in the headers of the same message. Then we relate them to references in other messages. An *actor* is a collection of references pointing to the same person. We use the e-mail address as a primary cue. 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).

2.3 Resulting data set

After the message deduplication and reference resolution, the resulting database contains 248,155 messages and a network of 114,986 different actors. Despite all our efforts, some ambiguity and inconsistency may still exist.

Key characteristics of the resulting set are displayed in Figure 1. In the first plot, we see that the frequency of words follows the power law distribution. 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. Importantly, the average e-mail is a relatively short text, one more reason to try to use other properties in classification. The third plot shows that even though

[1] http://www.usdoj.gov/enron/jury_trial.htm.

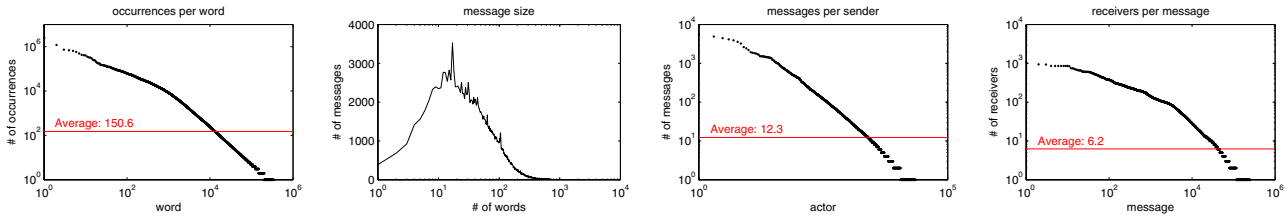

Figure 1: Characteristics of the resulting data set.

there are some very active actors in the network, most are sending very few e-mails. The number of receivers per message fits the power law 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.

3. DOCUMENT REPRESENTATION

The most obvious level of document representation is the textual content. It includes the body and subject fields in a *bag-of-words* representation, that has proven to be well suited for classification. The idea is to construct a vector where each feature represents a word in the lexicon and the feature values express some weight.

Then 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.

3.1 Text representation

We generate the lexicon by applying a number of criteria, in particular, we choose terms with a minimum occurrence of 4 in the entire corpus and a minimum length of 3 characters. Also, all keywords are stemmed with a Porter stemmer and lowercased.

We then construct vectors with each feature representing one word. The document frequency df of a word w is the number of documents it occurs in, the term frequency tf is the number of word occurrences in a document d, N is the number of documents. For the experiments, we consider three different text representations: *binary*, *frequency* and *tf-idf* values, defined as follows:

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

The feature set based on a bag-of-words representation are high-dimensional (around 105,000) and the feature vectors are very sparse. This makes it particularly suited for the SVM classification with a linear kernel [14]. There is a lot of redundancy in the feature vector: a document typically has a large number of cues that signal a particular classification.

Another problem with high-dimensional feature spaces for one-class classifier is the so-called *curse of dimensionality* [11]. It has been observed that noise hurts performance significantly, especially in the case of learning from unbalanced data. To investigate this effect, we test two alternative ways to reducing the feature space. The first is based on the *feature selection* where we select a subset of F words the highest document frequency. The other way of reducing the dimensionality is by *semantic clustering* of features. We use a soft clustering approach proposed in [2]. 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 length 6,522, where each feature represents a certain semantic field. Ambiguous words contribute to multiple features.

3.2 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, computer science and epidemiology [19]. With the advent of techniques to handle large graphs and the emergence of huge, real-life linked structures on the internet, structure in social networks has become a subject of intensive research.

The structure of a large e-mail corpus, like Enron, is not homogeneous. The lines of communication implicitly instantiate a network of actors. By setting thresholds on the minimal number of e-mails per connection, one can generate graphs with different levels of connectedness. We set this threshold to 2, making sure to keep the majority of the traffic while discarding any accidental links with no or little meaning. Due to the power law distribution of the connection strength, this reduces the number of actors considerably, without losing much of relevant information. We take the largest connected component (95% of the actors) of this graph, resulting in a network of 30,094 actors.

We expect our social network features to represent the position/role 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 [25]. We adopt and report below a set of commonly used features to represent key properties of actors [5, 25].

The communication network is a *directed weighted graph* $G = \langle V, E \rangle$, where V contains n nodes and the weight w of edge $(s,t) \in E$ reflects the activity from s to t (the number of e-mails sent). The first feature represents the activity of the actor in the network:

1. *the number of e-mails* sent, $m(v) = \sum_{(v,t)\in E} w(v,t)$.

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 linking to a large number of authorities are *hubs* [15].

2. *hub score* $h(v)$ is given by v-th element of the principal eigenvector of AA^T where A is the adjacency matrix corresponding to graph G;

3. *authority score* $a(h)$ is given by v-th element of the principal eigenvector of $A^T A$.

The next group is a set of different centrality measures to model the position of the node in network. These depend on a *undirected unweighted* version $G' = \langle V, E' \rangle$ of graph G. We calculate the shortest paths in G' using a Matlab library for working with large graphs [12]. We obtain the distance d_{st} from node s to t and the number σ_{st} of paths from s to t. 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 more feature characterizes the connectedness in the direct neighborhood of node v:

9. *clustering coefficient*: $CC(v) = \frac{2|(s,t)|}{(\deg(v)(\deg(v)-1))}$: $(v, s),(v, t), (s, t) \in E'$.

The final group calculates all cliques in the graph using a Matlab implementation of [6]. It includes three following features:

10. the *number* $clq(v)$ *of cliques* the actor v is in;
11. a *raw clique score* where each clique in $clq(v)$ of size n is given a weight 2^{n-1}, $CS_R(v) = \sum_{q \in clq(v)} 2^{size(q)-1}$;
12. a *weighted clique score* where each clique is weighted by the sum of activities of its members,

$$CS_w(v) = \sum_{q \in clq(v)} 2^{size(q)-1} \sum_{w \in q} m(w).$$

All scores are scaled to a value in $[0, 1]$ range, where 1 indicates a higher importance. Each node can be assigned an aggregated *social score* [25] which is a linear combination of all 12 features, with all features equally weighted.

Note we are not classifying the nodes (actors) in the social network, 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 each message.

4. ONE-CLASS CLASSIFICATION

The responsiveness problem in the Enron Corpus is typical for Information Retrieval settings. A small set of positive training examples is relatively straightforward to obtain (e.g. the medical records of patients that are diagnosed with a particular disease or the e-mails on the DOJ exhibit list). However it is often very difficult (if not impossible) to develop a set of objects that reflects the complement of the positive class.

Three key assumptions of the problem are the following:

- In a very large set of documents, we have a small set of truly positive examples $P = (\mathbf{x}_1, \ldots, \mathbf{x}_l)$ completed with a large unlabeled set $U = (\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u})$;
- 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.

In the next subsection we define an Support Vector Machines 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 [11, 17, 24].

We therefore adopt a semi-supervised framework, Mapping Convergence by [31], that copes with the positive examples and the huge amount of unlabeled data. Our core idea is to confidently assign examples from the unlabeled set as negatives in order to support a process of convergence towards an optimal hypothesis. We adapt the existing one-class interpretation to start with a very sparse set of positives and still obtain a good estimate of the performance of the generated hypotheses. Then we extend the algorithm implementing ideas from co-training [4] to combine complementary classifiers.

4.1 One-Class SVM

For Support Vector Machines, a few extensions have been proposed to enable learning in a one-class setting. Both the Support Vector Data Description algorithm (SVDD) [29] and One Class Support Vector Machines (OC-SVM) [26] are shown to be equivalent when data vectors are scaled to unit length [28]. We will use OC-SVMs, it allows us to formulate the optimization problem as in ν-SVM and keep the intuitivity of parameter μ.

The main idea behind OC-SVMs is to create a hyperplane in feature space where the projections of data are separated from the origin with a large margin. The data is separable from the origin if there exists such a vector $\mathbf{w}$ 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:

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

$$K(\mathbf{x}_i, \mathbf{x}_j) > 0 \qquad \forall i, j \tag{1}$$

$$K(\mathbf{x}_i, \mathbf{x}_i) = 1 \qquad \forall i. \tag{2}$$

It results in all mappings being in the positive orthant and on the unit sphere and being much tighter than for other kernels.

As it is pointed out in [26], there exists a strong connection between OC-SVMs and binary classification. Supposing we have a parametrization $(\mathbf{w}, \rho)$ for the supporting hyperplane of a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, we also have that $(\mathbf{w}, 0)$ is the parametrization of the maximally separating hyperplane for the labeled data set $\{(\mathbf{x}_1, +1), \ldots, (\mathbf{x}_\ell, +1), (-\mathbf{x}_1, -1), \ldots, (-\mathbf{x}_\ell, -1)\}$.

Also, supposing that we have a maximally separating hyperplane parametrized by $(\mathbf{w}, 0)$ for a data set $\{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_\ell, y_\ell)\}$ and with a margin $\rho/\|\mathbf{w}\|$, we know that the supporting hyperplane for $\{y_1\mathbf{x}_1, \ldots, y_\ell\mathbf{x}_\ell\}$ is parametrized by $(\mathbf{w}, \rho)$. 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 conventional interpretation, 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.

There exist two types of problems where one-class learning is particularly attractive [33]. In the first case, a majority set is well-sampled, but it is hard to sample from the minority class. This is the case in [20, 21] 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, thus reducing the problem to outlier detection.

In the second type of problems, the target class can be accurately sampled, but the data appears extremely unbalanced. In regular

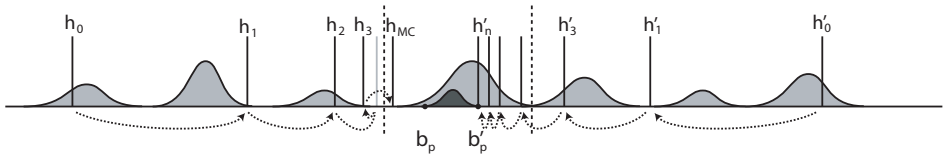

Figure 2: The Mapping Convergence in 1-dimensional space.

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 [24].

In a comparative study by Manevitz [17], 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. Yet, it remains computationally much less intensive than Neural Networks.

4.2 Mapping Convergence

To benefit from a large set of unlabeled data, the Mapping Convergence (MC) [31] assumes the existence of a gap between negative and positive data and tries to exploit it.

A basic intuition behind the MC is that, given a hypothesis h, items that are further from the decision boundary are classified with a higher probability. In Algorithm 1, this first approximation $\hat{N}_0$ of the negative distribution serves as input for the converging stage to move the boundary towards the positive training examples P. At iteration i, it trains an SVM on P and the constructed negatives N. The resulting hypothesis h_i 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, when no new negative items are discovered and the boundary comes to a hold.

Figure 2 from [31] is particularly instructive for understanding the MC. It shows seven clusters in 1D space, where 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).

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: $h_0 \Leftarrow$ train C_1 on P
2: $\hat{N}_0 \Leftarrow$ *strong negatives* ($\leq 10\%$) from U by h_0
$\hat{P}_0 \Leftarrow$ 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: $h_{i+1} \Leftarrow$ train C_2 on P and N
7: $\hat{N}_{i+1} \Leftarrow$ negatives from $\hat{P}_i$ by h_{i+1}
$\hat{P}_{i+1} \Leftarrow$ positives from $\hat{P}_i$ by h_{i+1}
8: $i \Leftarrow i + 1$
9: **end while**

Mapping Convergence is able to take advantage of the underlying structure of the unlabeled data and to 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. 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 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, it is unlikely that the algorithm selects a boundary that is very far from the training data. On the right-hand side, 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 put in $\hat{N}_0$ 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 labeled positives P very much.

4.3 A sequence of hypotheses

It has been shown that the MC performance dramatically decreases when the positive training set is severely undersampled [31]. This is explained by its incapacity to detect the gap between positive and negative data when too much unlabeled items act as noise. Iterations will not stop, causing the algorithm to overfit.

To better understand the MC dynamics, we created random data in $\mathbb{R}^2$ with no gap between positive and negative data. The dataset has 75 positives and 1,225 unlabeled data points, of which 1,000 are negatives and 225 are noise. We randomly generate Gaussian centroids of positive data, surrounded by negative data. The noise is added by switching the classification of 1% of the data. 80% of the data is selected randomly for training and the remaining 20% is used for testing [2].

The data as well as the MC iterations are shown in Figure 3. Figure 3.a displays the distribution of negatives (dots), unlabeled positives (diamonds) and labeled positives (circles). The MC takes as input P the labeled positives only, in Figure 3.b all unlabeled data U is represented as dots. The mapping stage (iteration 0) creates a conservative hypothesis excluding only a small proportion of U; its decision boundary is in Figure 3.b. From that the convergence proceeds (Figures 3.c to 3.e), with each iteration decreasing the number of unlabeled items from U. The performance on the

[2]Other ratios between positives and negatives give similar results.

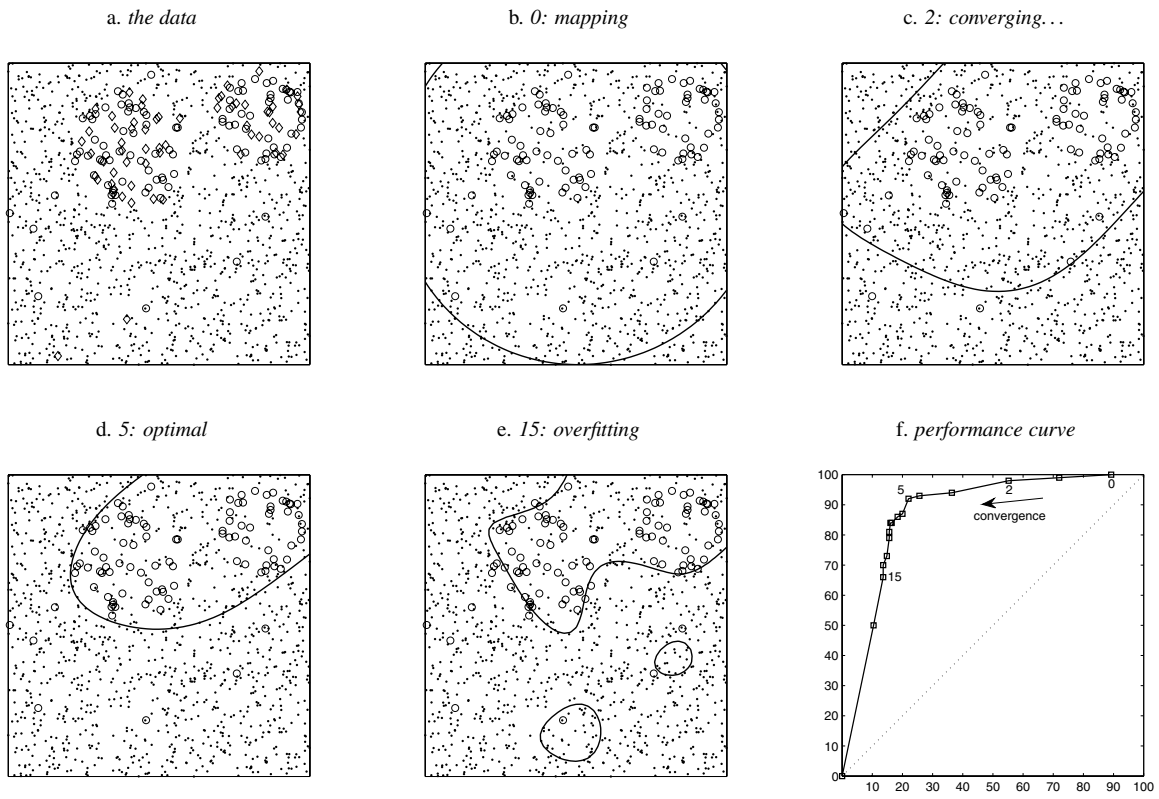

Figure 3: Random data in $\mathbb{R}^2$: representation of Mapping Convergence and the $\hat{P}P$ performance curve.

positives P starts decreasing, until 15-th iteration where overfitting takes place, as shown in Figure 3.e. Meanwhile, the fifth iteration produces the classifier with a fairly accurate description of the data.

Finally, the plot in Figure 3.f shows how the hypotheses h_i, $i = 0, 1, 2, \ldots$ are performing on separating out a small proportion of the entire data set (x-axis), while maintaining performance on the positive subset (y-axis). This novel performance measure is formally defined in the following section. Unlike the standard measures like precision/recall used on fully annotated data, the main advantage of the novel measure is its applicability in the truly one-class setting, when the true labels of U are unknown.

When working in the one-class setting, like the responsiveness in the Enron Corpus, we want to know when to stop the iterations before overfitting occurs. We will consider the convergence as a sequence of hypotheses, where each step is represented as a decision boundary in the feature space, and identify the hypothesis h_i that maximizes some performance measure. The key idea is that h_i should address a small part of U that retains a large part of the positive items from P. In that way we will find a hypothesis that strikes a good bias-variance balance.

4.4 $\hat{P}P$ measure

In the following, we use curves like the one in Figure 3.f to track the distinguishing power of a classifier. The MC produces a sequence of classifiers h_i, $i = 0, 1, 2, \ldots$. For each h_i, we plot on x-axis the percentage of the entire data set classified positively, $\hat{P}_i/|U|$. On the y-axis, we plot the percentage of the positives that is found within that space, $|P \cap h_i^+(P)|/|P|$. We call it $\hat{P}P$ *measure*. The one-class classifier h_0 used in the mapping phase produces the first (unconnected) point in the plane. Then we construct the curve, starting from the right-top with the mapping hypothesis and moving left and down with each step.

The curve of $\hat{P}P$ measures can be interpreted in a ROC-like fashion. The upper left corner represents a perfect classifier, points on the diagonal are random selection from U. On each iteration step of MC, a classifier is generated that gives a point on a curve like Figure 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 (it moves left), but potentially also a lower performance on the true positives (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.

Contrary to a genuine ROC curve, the $P\hat{P}$-curve is not continuous, but a sequence of points (x_i, y_i), on the plane. Points (0,0) an (100,100) refer to the two naive decisions of retaining none or all data in P and U. The first choice of the measuring is to build a piece-wise linear function induced by points (0,0), (x_i, y_i), (100,100). Unfortunately, the step-like behaviour of points (x_i, y_i) is not guaranteed. As we will see in Section 5, they may have an erratic behavior that makes impossible to calculate the area under the curve (AUC).

An alternative to the AUC is to identify the point in the $P\hat{P}$-curve that discard most of data in U, while keeping a large part of the positive data P to be classified correctly. The point on the curve that is closest to $(0, 100)$ is considered as the best classifier. The distance measure can be weighted to assign more importance to recall or precision, in this paper we will use the Euclidean distance.

4.5 Generating hypotheses from unlabeled data

When applying the framework described above to the Enron case, we should estimate the percentage of positives returned by each hypothesis h_i 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.

As a solution, we introduce a cross-validation step in the algorithm. Each step in MC process is carried out with a 5-fold split over the positive data P. We train a hypothesis on 4 parts, and predict on the remaining fifth part and the entire unlabeled set U. This results in exactly one prediction per item in P, and then we aggregate the five predictions for the items in U to obtain a solid estimate of the performance of the hypothesis.

Note that LIBSVM by default predicts class labels it can produce probability estimates for each of the classes, using a method by Platt [22] that fits a logistic function to the output of an SVM to generate posterior probabilities. This algorithm presumes that the equal distribution of positives and negatives in training and test sets. 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 and converges to the actual one.

Therefore we altered the LIBSVM code to directly output the distance to the decision plane and use it as a measure of the confidence of prediction. After scaling to (0,1) we can aggregate the predictions of multiple classifiers, for example by taking averages. A more advanced way of capturing the confidence of predictions might be subject of future research.

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 a higher computational cost.

4.6 Combining classifiers

In Section 2 we defined two complementary views of documents in the Enron Corpus. Combining different classifiers to improve overall performance is known as *ensemble learning* [9]. In this section we propose to combine the MC algorithm with the idea of co-training.

First we consider a *naïve* way of combining predictions. When the MC algorithm produces 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 one classifier based on view A and a second based on view B. 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 each prediction is a number in the [0,1] range and represents the 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) in $\hat{P}P$ measure. Moving left in the performance plane means a smaller part of U, while moving up corresponds to retaining more positives from P. As we will see in Section 5, 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* [4]. 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 *Mapping Co-Convergence* we propose is depicted as Algorithm 2.

Algorithm 2 Mapping Co-Convergence

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

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

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

There are three important differences between Algorithm 2 and Algorithm 1 discussed in Section 4.2. First, it starts with not one but n different views of the data. Second, the views interact by means of the *aggregation function* that combines the predictions to select the next items to label. Finally, in the convergence phase, not all items recognized as negative are added to N, but only the ones that are agreed upon with the highest certainty are labeled, limited to 5% of the entire data set.

Note that the two-step assemble learning is not a unique way to combine different modalities. In Web page classification [23], there exist other techniques to take both content and links structure into account. Simple approaches convert one type of information to the other. For example, in spam blog classification, Kolari et al. [16] concatenate outlink features with the content features of the blog. Other techniques [32] try to jointly process all modalities. They use the same set of latent concepts to simultaneously capture the pairwise relationships between modalities. One-class classification with the joint modeling of different modalities may be a subject of further research.

5. RESULTS AND ANALYSIS

We develop our framework in Python, and use the LIBSVM library [7]. 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,

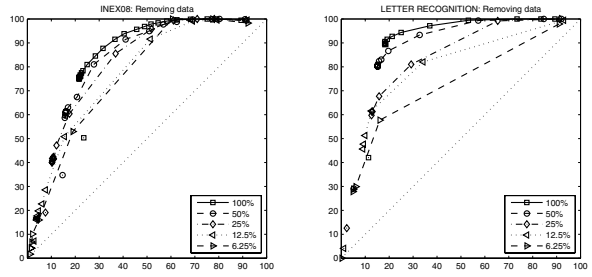

Figure 4: $\hat{P}P$ curves for different sizes of positive data. a) INEX08, b) Letter recognition.

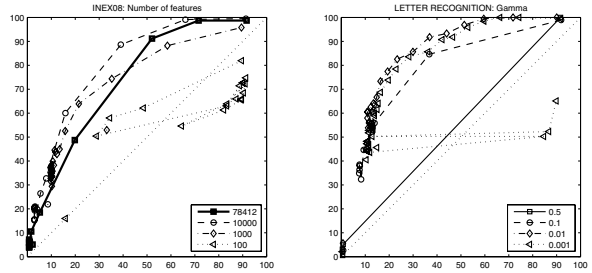

Figure 5: $\hat{P}P$ curves. a) Feature selection in INEX08, b) γ values in Letter Recognition.

controlling the smoothness of the decision boundary. 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 use 5-fold cross-validation, for testing on unlabeled items we hold out 500 items.

5.1 Two reference corpora

We first report some $\hat{P}P$ curves on two reference corpora: the INEX08 Wikipedia challenge corpus [1], containing text files with subject class labels and the Letter Recognition dataset from the UCI ML repository, containing samples of handwriting classified with the correct symbol. The INEX08 corpus contains 11,437 objects described by 78,412 features representing a BOW 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 positives (21%), 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 is the best option. We use classes 0, 1, 2, 3 and 4 as positives (20%) and the rest as negatives.

In both cases we use 80% of the data to train a hypothesis and the remaining 20% to test it. Figure 4 shows the influence of removing positive training data on both corpora. 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 5.a shows the influence of reducing the number of features for INEX08 set. 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). The final experiment in Figure 5.b shows the influence of the γ parameter of a Gaussian kernel on the Letter Recognition data. We perform the experiment with 12.5% of the positives and 50% as noise (ratio 1/4). Large values of γ give larger steps, eventually jumping to (0,0) immediately. Small values give overfitting, evidenced by erratic behaviour.

5.2 Classifying Enron

Now we present our experiments on classifying the Enron corpus using different representations discussed in Section 3. First we try to get the optimal setting of parameters to obtain good classifiers for each modality, before we will try to combine their predictions.

Text-based representations. For the text-based representations, we first determine what type of feature values performs better. As shown in Figure 6.a, The performance curve in Figure 6.a shows that both binary and tf-idf values give similar results, with tf-idf being slightly better and more stable. 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 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.

Because we use a one-class classifier, the dimension of the feature space might be important. Even though with our reference corpus we had a slight increase in performance by reducing the number of features, no such thing happens with this one. In Figure 6.b we even see that the OC-SVM performs considerably worse with less features. The convergence itself however is not hurt. Also the reduction of features using semantic clustering does not seem to improve anything.

Social network-based representations. When classifying with the social network features, we have a lot less tuning to do. The feature values are not sparse and reducing the number of features is not considered. However, with the RBF kernel we have an additional γ parameter. Bigger values lead to underfitting because of 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 $\gamma = 0.1$, see Figure 6.c. Optimally we select 71.5% of the positives in 9.4% of the data.

5.3 Combining classifiers

Finally we present the results of combining the classifiers from the previous section. 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. For an aggregation function we simply take the average of the predictions. In Figure 7.a, we combine the classifiers we get at the 12th and 13th steps on the social network curve and 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.

Mapping Co-Convergence. This approach combines the different views on the data on each convergence step. 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 as shown in Figure 7.b is very satisfactory, retaining 93% of the positives while discarding 90% of the unlabeled data.

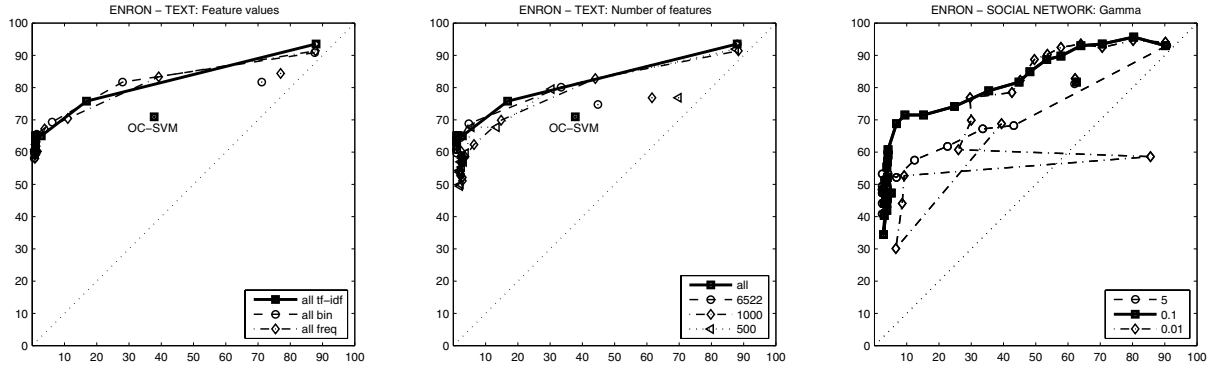

Figure 6: Classifying Enron Corpus. a) text-based, b) social network-based, c) γ values.

classifier	part of U	part of P	distance
text (bow tf-idf)	24.19	16.80	29.45
social network	29.49	9.40	30.00
text + social (naïve)	18.38	9.22	20.56
text + social (MCC)	**9.68**	**6.40**	**11.60**

Table 1: Comparison of the different classifiers.

As explained before, we represent any curve by its "best" classifier. We take the Euclidian distance to the perfect classifier (0,100), to be compare different classifiers. Table 1 reports the best classifiers of all curves discussed in this section. We can see that the combination of social network-based and text-based feature sets allows to achieve the best performance.

6. ACKNOWLEDGMENT

We would like to thank Virginia Dignum from Utrecht University for early discussions and feedback on the preliminary report. This work is partially supported by the FRAGRANCE Project co-funded by the French Association on Research and Technology (ANRT).

7. CONCLUSION

From the results presented above 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 hte first approximation of the negative set to support convergence. From our experiments, 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 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 [22] is actually implemented in the LIBSVM toolkit [7] 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 definately 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.

8. REFERENCES

[1] INEX08 Wikipedia Challenge Corpus. `http://xmlmining.lip6.fr/`, 2008.

[2] Julien Ah-Pine and Guillaume Jacquet. Clique-based clustering for improving named entity recognition systems. In *EACL*, pages 51–59, 2009.

[3] 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.

[4] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. Workshop on Comp. Learning Theory*. Morgan Kaufmann Publishers, 1998.

[5] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

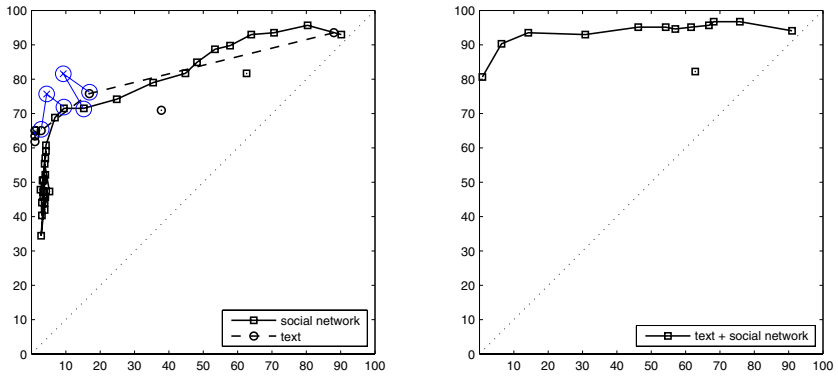

Figure 7: Enron classifiers combination: a) naïve combination; b) Mapping Co-Convergence framework.

[6] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *CACM*, 16(9):575–577, 1973.

[7] 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`.

[8] William Cohen. `http://www.cs.cmu.edu/~enron/`.

[9] Thomas G. Dietterich. Ensemble methods in Machine Learning. In J. Kittler and F. Roli, editors, *Proc. Multiple Classifier Systems, 2000*, volume 1857, pages 1–15, 2000.

[10] Tamer Elsayed and Douglas Oard. Modeling identity in archival collections of email: a preliminary study. In *Conference on Email and Anti-Spam*, 2006.

[11] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In *Advances in Soft Computing : The Challenge of Complexity*, pages 425–438, 2006.

[12] David Gleich. MatlabBGL: a Matlab graph library. `http://www.stanford.edu/\~dgleich/programs/matlab\_bgl/`, 2008.

[13] Jeffrey Heer. Exploring Enron: Visual data mining of e-mail. `http://www.jheer.org/enron/`, 2005.

[14] Thorsten Joachims. A statistical learning model of text classification for Support Vector Machines. In *Proc. ACM SIGIR Conf.*, pages 128–136, 2001.

[15] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[16] Pranam Kolari, Tim Finin, and Anupam Joshi. Svms for the blogosphere: Blog identification and splog detection. In *AAAI Spring Symp. CAAW*, 2006.

[17] Larry M. Manevitz and Malik Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[18] Andrew McCallum, Xuerui Wang, and Andres Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *J. Artificial Intelligence Research*, 30:249–272, 2007.

[19] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[20] 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.

[21] 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: Proc. Sixth Intern. Conf. Data Mining*, pages 488–498, 2006.

[22] 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.

[23] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2):1–31, 2009.

[24] Bhavani Raskutti and Adam Kowalczyk. Extreme re-balancing for SVMs: a case study. *SIGKDD Explorations*, 6(1):60–69, 2004.

[25] Ryan Rowe, German Creamer, Shlomo Hershkop, and Salvatore J. Stolfo. Automated social hierarchy detection through email network analysis. In *Proc. Workshop on Web Mining and Social Network Analysis*, pages 109–117, 2007.

[26] 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.

[27] Jitesh Shetty and Jafar Adibi. Discovering important nodes through graph entropy the case of Enron email database. In *LinkKDD '05: Proc. 3rd International Workshop on Link Discovery*, pages 74–81, 2005.

[28] David M. J. Tax. *One-class Classification*. PhD thesis, Delft University of Technology, The Netherlands, June 2001.

[29] David M. J. Tax and Robert P. W. Duin. Support Vector Domain Description. *Pattern Recogn. Lett.*, 20(11-13):1191–1199, 1999.

[30] Jen-Yuan Yeh and Aaron Harnly. Email thread reassembly using similarity matching. In *Conference on Email and Anti-Spam*, 2006.

[31] Hwanjo Yu. Single-class classification with Mapping Convergence. *Machine Learning*, 61(1-3):49–69, 2005.

[32] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *ACM SIGIR Conf.*, pages 487–494, 2007.

[33] Lin Zhuang and Honghua Dai. Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers*, 1(7):32–40, 2006.