In *Intelligent Data Analysis*, 8(5), 2004.

Email Answering Assistance by Semi-Supervised Text Classification

Tobias Scheffer
Humboldt-Universität zu Berlin
Department of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
scheffer@informatik.hu-berlin.de

February 19, 2004

Abstract

Many individuals, organizations, and companies have to answer large amounts of emails. Often, many of these emails contain variations of relatively few frequently asked questions. We address the problem of predicting which of several frequently used answers a user will choose to respond to an email. We map the problem to a semi-supervised text classification problem. In a case study with emails that have been sent to a corporate customer service department, we investigate the ability of the naive Bayesian and support vector classifier to identify the appropriate answers to emails. We study how effectively the transductive Support Vector Machine and the co-training algorithm utilize unlabeled data and investigate why co-training is only beneficial when very few labeled data are available. In addition, we describe a practical assistance system.

1 Introduction

Companies allocate considerable economic resources to communication with their customers. A continuously increasing share of this communication takes place via email; marketing, sales and customer service departments as well as dedicated call centers have to process high volumes of emails with many messages containing repetitive routine questions. It appears overly ambitious to completely automate this process; however, any software support that leads to a significant productivity increase is already greatly beneficial. Our approach to support this process is to predict which answer a user will most likely send in response to an incoming email, and to propose this answer to the user. The user, however, is free to modify – or to dismiss – the proposed answer.

Our approach is to learn a predictor that decides which of a small set of standard answers a user is most likely to choose in reply to a given inbound message. We learn such a predictor from the available data: inbound and outbound emails. We transform the email answering problem into a set of semi-supervised text classification problems. Contrasting studies that investigate identification of general subject areas of emails (*e.g.,* Green & Edwards, 1996), we explore whether text classification algorithms can identify instances of a specific frequently asked question.

Many approaches are known that learn text classifiers from data. The Support Vector Machine (SVM) (Vapnik, 1998) is generally considered to be one of the most accurate algorithms; this is supported, for instance, by the TREC filtering challenge (Lewis, 1997). The naive Bayes algorithm (*e.g.,* McCallum & Nigam, 1998) is also widely used for text classification problems.

Semi-supervised learning algorithms (Cooper & Freeman, 1970; for an overview, see Seeger, 2001) utilize unlabeled data to improve classification accuracy. In the naive Bayesian framework, unlabeled training data can be utilized by the EM parameter estimation algorithm. The transductive Support Vector Machine (Joachims, 1999a) provides a method for utilizing unlabeled data in the support vector framework. The co-training algorithm (Blum & Mitchell, 1998) exploits independencies between attribute subsets; it can be wrapped around support vector classifiers or other learning methods to utilize unlabeled data.

The contribution of this paper is threefold. Firstly, we analyze the problem of answering emails, discussing practically relevant aspects. Secondly, we present a case study showing how well the naive Bayes algorithm, the Support Vector Machine, the transductive Support Vector Machine and the co-training algorithm can identify instances of particular questions in emails. We will see that transduction and co-training are beneficial for this problem if and only if very few training examples are available. We present further experiments that provide insights on why this is the case for co-training. Thirdly, we describe how we integrated machine learning algorithms into a practical answering assistance system.

The rest of this paper is organized as follows. In Section 2, we analyze the problem setting and introduce our case study. We discuss our general approach and our mapping of the email answering problem to a set of semi-supervised text classification problems in Section 3 and discuss the performance of this approach in the context of our case study. In Section 4, we describe the transductive SVM and co-training algorithm; here, we also investigate the effectiveness – and the reasons for the lack of effectiveness – of co-training empirically. In Section 5, we describe how we have integrated our learning approach into the Responsio email management system. Section 6 discusses related approaches.

2 Problem Setting and Case Study

We study the problem of email answering, focusing on application scenarios for which a substantial fraction of all incoming emails can be answered with a modestly large set of *standard answers.* We expect to find this situation in corporate service centers. We assume that a set of standard answers $A_1, \ldots, A_n$

has been identified manually. Hence, we consider the problem of predicting which of n standard answers $A_1, \ldots, A_n$ a user will reply to an email.

In order to learn a predictor, we are initially given a repository $\{x_1, \ldots, x_m\}$ of inbound, and $\{y_1, \ldots, y_{m'}\}$ of outbound emails. Typically, these repositories contain at least hundreds, but often thousands of emails stored on a corporate email server. When the assistance system is being used, additional inbound and outbound messages are perceived, possibly governed by a drifting distribution.

Although both inbound and outbound emails are stored, it is not trivial to identify which outbound email has been sent in reply to a particular inbound email; neither the emails nor the internal data structures of the Outlook email client contain explicit links. When an outbound email does not *exactly* match one of the standard answers, this does *not* necessarily mean that none of the standard answers is the correct prediction. The user could have written an answer that is semantically equivalent to one of the answers A_i but uses a few different terms.

A characteristic property of the email answering domain is a non-stationarity of the distribution of inbound emails. While the likelihood $P(x|A_i)$ is quite stable over time, the prior probability $P(A_i)$ is not. Consider, for example, a server breakdown which will lead to a sharp increase in the probability of complaints; or an advertising campaign for a new product which will lead to a high volume of requests for information on that product.

What is the appropriate utility criterion for this problem? Our goal is to assist the user by proposing answers to emails. Whenever we propose the answer that the user accepts, he or she benefits; whereas, when we propose a different answer, the user has to manually select or write an answer. Hence, the optimal predictor proposes the answer A_i which is most likely given x (*i.e.,* maximizes $P(A_i|x)$). and thereby minimizes the probability of the need for the user to write an answer manually.

The case study that we discuss in this paper is based on data which was provided by the TELES European Internet Academy, an education provider that offers classes held via the internet. In order to evaluate the predictors, we manually labeled all inbound emails within a certain period with the matching answer. Table 1 provides an overview of the data statistics. Roughly 72% of all emails received can be answered by one of nine standard answers. The most frequent question "product inquiries" (requests for the information brochure) already covers 42% of all inbound emails.

We briefly summarize the basic principles of ROC analysis which we use to assess decision functions (Bradley, 1997; Provost et al., 1998). The *receiver operating characteristic* (ROC) curve of a decision function plots the number of true positives against the number of false positives. By comparing the decision function against a decreasingly large threshold value we observe a trajectory of classifiers described by the ROC curve.

The area under the ROC curve is equal to the probability that, when we draw one positive and one negative example at random, the decision function assigns a higher value to the positive example than to the negative. Hence, the area under the ROC curve (the *AUC performance*) is a very natural measure of

Table 1: Statistics of the TEIA email data set.

Frequently answered question	emails	percentage
Product inquiries	224	42%
Server down	56	10%
Send access data	22	4%
Degrees offered	21	4%
Free trial period	15	3%
Government stipends	13	2%
Homework late	13	2%
TELES product inquiries	7	1%
Scholarships	7	1%
Individual questions	150	28%
Total	528	100%

the ability of a decision function to separate positive from negative examples.

3 Email Answering by Classification

In this Section, we discuss our general approach that reduces the email answering problem to a semi-supervised text classification problem.

Firstly, we have to deal with the non-stationarity of the prior $P(A_i)$. In order to predict the answer that is most likely given x, we have to choose $\text{argmax}_i P(A_i|x) = \text{argmax}_\text{i} P(x|A_i)P(A_i)$ where $P(x|A_i)$ is the likelihood of question x given that it will be answered with A_i and $P(A_i)$ is the prior probability of answer A_i. Assuming that the answer will be exactly one of $A_1, \ldots, A_n$ we have $\sum_i P(A_i) = 1$.

We know that the likelihood $P(x|A_i)$ is stationary; only a small number of probabilities $P(A_i)$ has to be estimated dynamically. Equation 1 averages the time dependent priors (estimated by counting occurrences of the A_i in the outbound emails within time interval t) discounted over time. $t = 0$ is the present; a large value for the parameter λ has to be chosen when $P(A_i)$ is drifting fast; $\lambda = 0$ is chosen for a stationary $P(A_i)$.

$$P(A_i) = \frac{\sum_{t=0}^{T} e^{-\lambda t} P(A_i|t)}{\sum_{t=0}^{T} e^{-\lambda t}} \tag{1}$$

We can now focus on estimating the (stationary) likelihood $P(x|A_i)$ from the data. In order to map the email answering problem to a classification problem, we have to identify positive and negative examples for each answer A_i.

When the email assistance system is set up initially, we manually identify small sets S_i of inbound emails which can appropriately be answered using standard answer A_i. When the system is being used, it monitors all mails. When an outbound mail equals one of the standard answers A_i, then the corresponding inbound mail becomes a member of S_i; it is an example of an email that the user answers with A_i.

We use the following heuristic to identify cases where an outbound email is a response to a particular inbound mail. The recipient has to match the sender of the inbound mail, and the subject lines have to match up to a prefix ("Re:" for English or "AW:" for German email clients). Furthermore, the outbound mail has to be sent while the inbound mail is visible in one of the active windows. (We are able to check the latter condition because our email assistance system is integrated into the Outlook email client and monitors user activity.) Using this rule, we are able to identify *some* inbound emails S_i as positive examples for each standard answer A_i.

We also need to identify negative examples. We can safely assume that no two different standard answers A_i and A_j are semantically equivalent. Hence, when an email has been answered by A_i, we can conclude that it is a negative example for all A_j, $j \neq i$. When the answer to an inbound email is different from all standard answers A_i, we cannot conclude that the inbound mail is a negative example for all standard answers because the response might have been semantically equivalent, or very similar, to one of the standard answers. Such emails are unlabeled examples in the resulting text classification problem.

For the same reason, we cannot obtain examples of inbound emails for which no standard answer is appropriate; hence, we cannot estimate $P($no standard answer$)$ or $P(\bar{A}_i)$ for any A_i. Thus, we have a small set of positive and negative examples for each A_i. Additionally, we have a large quantity of emails for which we cannot determine the appropriate answer.

3.1 Predicting one of Multiple Answers with SVMs

We now address the problem of predicting the most likely of several mutually exclusive answers using Support Vector Machines. We first have to reduce the multi-class learning problem to a set of binary learning problems. In our application, the class priors are unevenly distributed and are unstationary. Therefore, we use a probabilistic one-against-all approach that explicitly considers the prior probabilities.

The SVM returns an uncalibrated decision function f_i for each binary classification problem; our decision on the answer to x has to be based on the $f_i(x)$ (Equation 2). We discriminate each class A_i against all other classes; that is, we have to assume that A_i is conditionally independent of all $f_{j \neq i}(x)$, given $f_i(x)$. Since we have dynamic estimates of the non-stationary $P(A_i)$, Bayes' equation (Equation 3) provides us with a mechanism that combines n binary decision functions and the prior estimates optimally.

$$\begin{aligned} \mathrm{argmax_i} P(A_i|x) &= \mathrm{argmax_i} P(A_i|f_1(x), \ldots, f_n(x)) & (2) \\ &\approx \mathrm{argmax_i} P(A_i|f_i(x)) = \mathrm{argmax_i} P(f_i(x)|A_i) P(A_i) & (3) \end{aligned}$$

Equation 3 is only applicable for discrete $f_i(x)$, while the decision function values are really continuous. In order to estimate $P(f_i(x)|A_i)$ we have to fit a parametric model to the data. Now we assume that the decision function values are governed by normal distributions: $P(f_i(x)|A_i) = N[\mu_i, \sigma_i^2](f_i(x))$. We estimate μ_i, σ_i, $\bar{\mu}_i$, and $\bar{\sigma}_i^2$ from the training data: we learn a decision function $f_i(x)$ and estimate mean value μ and variance σ_i^2 of the positive and $\bar{\mu}_i^2$ and $\bar{\sigma}_i^2$ of the negative examples. In the next step we calculate the posteriors $P_i(A_i|f_i(x))$ by applying Bayes' equation as shown in Equation 4.

$$P_i(A_i|f_i(x)) = \frac{N[\mu_i, \sigma_i^2](f_i(x))P(A_i)}{N[\mu_i, \sigma_i^2](f_i(x))P(A_i) + N[\bar{\mu}_i, \bar{\sigma}_i^2](f_i(x))P(\bar{A}_i)} \tag{4}$$

We have now reduced the email answering problem to a semi-supervised text classification problem. We have n binary classification problems for which few labeled positive and negative and many unlabeled examples are available. We need a text classifier that returns a (possibly uncalibrated) decision function $f_i : X \rightarrow real$ for each of the answers A_i.

3.2 Naive Bayes and SVMs for Answer Prediction

In our approach, a text classifier for each standard answer has to identify instances of questions that can appropriately be answered with a common standard answer. Following up on our case study, we will now evaluate empirically how well classifiers are able to identify instances of these frequently asked questions. We consider a Naive Bayes classifier and the support vector machine SVM^{light} (Joachims, 1999a). Both classifiers use the bag-of-words representation which considers only the words occurring in a document, but not the word order. As preprocessing operation, we tokenize the documents but do not apply a stemmer. For SVM^{light}, we calculate tf-idf vectors; that is, we multiply the frequency of each term with its inverse document frequency and normalize the resulting vector.

In order to estimate the AUC performance and its standard deviation for a decision function, we perform between 7 and 20-fold stratified cross validation and average the AUC values measured on the held out data. In order to plot the actual ROC curves, we also perform n-fold cross validation. In each fold, we file the decision function values of the held out examples into one global histogram for positives and one histogram for negatives. After between 7 and 20 folds, we calculate the ROC curves from the resulting two histograms.

First, we study the performance of a decision function provided by the Naive Bayes algorithm as well as the support vector machine SVM^{light} (Joachims, 1999a). We use the default parameter settings for SVM^{light}. Figure 1 shows that the SVM impressively outperforms Naive Bayes in all cases except for one (TELES product inquiries). Remarkably, the SVM is able to identify even very specialized questions with as little as seven positive examples with between 80 and 95% AUC performance. It has earlier been observed that the probability estimates of Naive Bayes approach zero and one, respectively, as the length of analyzed document increases (Bennett, 2000). This implies that Naive Bayes performs poorly when not all documents are equally long, as is the case here.

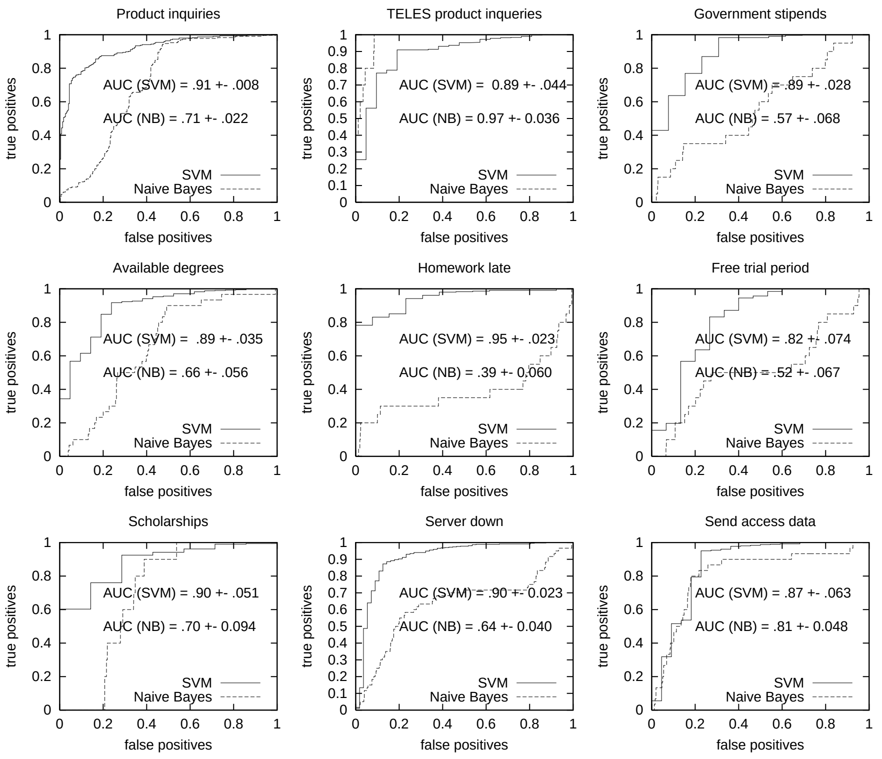


Figure 1: ROC curves for nine most frequently asked questions of naive Bayes and the support vector machine.

4 Semi-Supervised Learning

We briefly sketch two approaches that allow to utilize unlabeled data for support vector learning: the transductive SVM, and the co-training algorithm.

4.1 Transduction

In order to calculate the decision function for an instance x, the support vector machine calculates a linear function $f(x) = wx + b$. Model parameters w and b are chosen such that the margin is maximized. The soft margin SVM allows for some violation of this margin and minimizes a trade-off between margin width and slack variables ξ_i.

Optimization Problem 1 *Given labeled data D_l and parameter C, minimize Equation 5 over all w, b, and $\xi_1, \dots, \xi_{m_l}$, subject to the constraints 6 and 7.*

$$\min_{w,b,\xi} \frac{1}{2}|w|^2 + C \sum_{j=1}^{m_l} \xi_j \qquad (5)$$

$$\forall_{j=1}^{m_l} y_j(wx_j + b) \geq 1 - \xi_j \tag{6}$$
$$\forall_{j=1}^{m_l} \xi_j > 0 \tag{7}$$

The *transductive Support Vector Machine (TSVM)* (Joachims, 1999b) furthermore considers unlabeled data $D_u = \{x_1^*, \ldots, x_{m_u}^*\}$. This unlabeled data can (but need not) be new instances which the SVM is to classify. In transductive support vector learning, the optimization problem is reformulated such that the margin between all (labeled and unlabeled) examples and hyperplane is maximized. However, only for the labeled examples we know on which side of the hyperplane the instances have to lie.

Optimization Problem 2 *Given labeled data D_l, unlabeled data D_u, and parameters C and C^*, the* TSVM optimization problem *is to minimize Equation 8 over all possible values of w, b, $y_1^*, \ldots, y_{m_u}^*$, $\xi_1, \ldots, \xi_{m_l}$, and $\xi_1^*, \ldots, \xi_{m_u}^*$ subject to the constraints 9, 10, and 11.*

$$\min_{w,b,\xi,\xi^*,y^*} \frac{1}{2}|w|^2 + C\sum_{j=1}^{m_l} \xi_j + C^* \sum_{j=1}^{m_u} \xi_j^* \tag{8}$$
$$\forall_{j=1}^{m_l} y_j(wx_j + b) \geq 1 - \xi_j \tag{9}$$
$$\forall_{j=1}^{m_u} y_j^*(wx_j + b) \geq 1 - \xi_j^* \tag{10}$$
$$\forall_{j=1}^{m_l} \xi_j > 0,\ \forall_{j=1}^{m_u} \xi_j^* > 0 \tag{11}$$

The TSVM algorithm starts by learning parameters from the labeled data and labels the unlabeled data using these parameters. It iterates a training step and switches the labels of pairs of unlabeled data such that optimization criterion 2 is maximized; the influence of the unlabeled data is increased after every iteration. The TSVM algorithm is described in (Joachims, 1999a).

In the next set of experiments, we observe how the transductive support vector machine improves the performance by utilizing the available unlabeled data. We successively reduce the amount of labeled data and use the remaining data (with stripped class labels) as unlabeled and hold-out data (we use the same setting for the co-training experiments described in the following). We average five re-sampled iterations with distinct labeled training sets. We compare the SVM performance (only the labeled data are used by the SVM) to the performance of the transductive SVM (using both labeled and unlabeled data). Table 2 shows the results for category "general product inquiries"; Table 3 for "server breakdown".

When the labeled sample contains 24 positive and 33 negative instances for "general product inquiries", or $10 + 30$ instances for "server breakdown", then SVM and transductive SVM perform equally well. With 50 positive and 66 negative examples, transduction even deteriorates accuracy for the "product inquiries" class. When the labeled sample is smaller, then the transductive SVM outperforms the regular SVM significantly. We can conclude that, for this problem, transduction is beneficial and improves recognition significantly if (and only if) only few labeled data are available. Note that the SVM with only 8+11 examples ("general product inquiries") or 5+15 examples ("server breakdown"), respectively, still outperforms the naive Bayes algorithm *with all available data.*

Table 2: SVM and transductive SVM, "general product inquiries".

Labeled data	SVM (AUC)	TSVM (AUC)
50 pos + 66 neg	0.889 ± 0.005	0.861 ± 0.008
24 pos + 33 neg	0.87 ± 0.0072	0.876 ± 0.007
16 pos + 22 neg	0.855 ± 0.007	0.879 ± 0.007
8 pos + 11 neg	0.795 ± 0.0087	0.876 ± 0.0068

Table 3: SVM and transductive SVM, "server breakdown".

Labeled data	SVM (AUC)	TSVM (AUC)
10 pos + 30 neg	0.889 ± 0.0088	0.878 ± 0.0088
5 pos + 15 neg	0.792 ± 0.01	0.859 ± 0.009

4.2 Co-Training

Blum and Mitchell (1998) have proposed the co-training algorithm that utilizes unlabeled data. In their approach, the available attributes V are split into two subsets V_1 and V_2 such that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. A labeled example (x, a) is then viewed as (x_1, x_2, a) where x_1 contains the values of the attributes in V_1 and x_2 the values of attributes in V_2.

The idea of co-training is to learn two classifiers $f^1(x_1)$ and $f^2(x_2)$ which bootstrap each other by providing each other with labels for the unlabeled data. Co-training is applicable when either attribute set suffices to learn the target f – *i.e.,* there are classifiers f^1 and f^2 such that for all x: $f^1(x_1) = f^2(x_2) = f(x)$ (the *compatibility* assumption). When the views are furthermore *independent* given the class labels – $P(x_1|f(x), x_2) = P(x_1|f(x))$ – then co-training labels unlabeled examples in a way that is essentially identical to drawing new labeled data at random (Blum & Mitchell, 1998).

As V_1, we use randomly drawn 50% of the words occurring in the training corpus; V_2 contains the remaining words. $f^1(x_1)$ and $f^2(x_2)$ are trained from the labeled examples. Now f^1 selects two examples from the unlabeled data that it most confidently rates positive and negative, respectively, and adds them to the labeled sample. If the representations in the two views are truly independent, then the new examples are randomly drawn positive and negative examples for f^2. Now f^2 selects two unlabeled examples, the two hypotheses are retrained, and the process recurs. The algorithm is presented in Table 4.

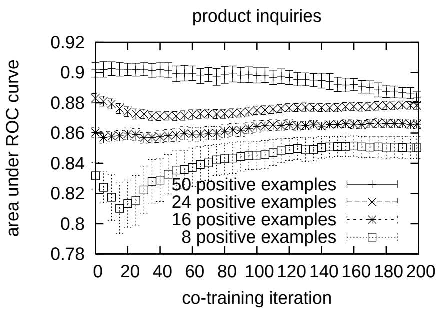
The compatibility and independence assumptions are usually violated to some degree in practice. However, empirical studies (Muslea et al., 2002; Kiritchenko & Matwin, 2002) show that co-training can nevertheless improve performance in some cases. In particular, text classification problems seem to be

Table 4: Co-training algorithm.

```
Input: Labeled examples $D_l$ in views $V_1$ and $V_2$; unlabeled data $D_u$; number
of iterations $T$; "step size" $n_p$ and $n_n$.

  1. Train $f_0^1$ and $f_0^2$ on $D_l$ using attribute sets $V_1$ and $V_2$, respectively.
  2. For $i = 1 \ldots T$ until $D_u = \emptyset$:
      (a) For $v = 1 \ldots 2$: Remove $n_p$ elements with greatest $f_{i-1}^v(x_j^*)$, from $D_u$
          and add $(x_j^*, +1)$ to $D_l$.
      (b) For $v = 1 \ldots 2$: Remove $n_n$ elements with smallest $f_{i-1}^v(x_j^*)$, from
          $D_u$, add $(x_j^*, -1)$ to $D_l$.
      (c) Train $f_i^1$ and $f_i^2$ on $D_l$ using attribute sets $V_1$ and $V_2$, respectively.
  3. Return the combined classifier $f(x) = f_T^1(x_1) + f_T^2(x_2)$.
```

particularly suited for co-training (Nigam & Ghani, 2000). In the next set of experiments, we use co-training in association with SVM^{light}. Our aim is to evaluate whether co-training effectively utilizes the unlabeled data that is available here.



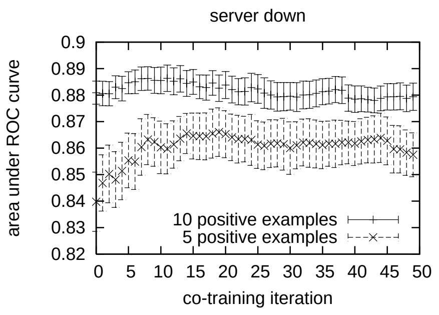


Figure 2: Change of AUC performance with increasing numbers of co-training iterations.

Figure 2 shows the AUC performance against the number of co-training iterations, averaged over 20 random splits of the attributes into V_1 and V_2. Co-training improves performance only when at most 16 positive examples are available for product inquiries and when at most 5 positive examples are available for server breakdown. When 50 or more labeled positive (and correspondingly many negative) example are available, then classification performance deteriorates over the co-training iterations.

What are the reasons for this deterioration? The co-training algorithm relies on the assumptions of compatibility – *i.e.,* there are classifiers f^1 and f^2 such

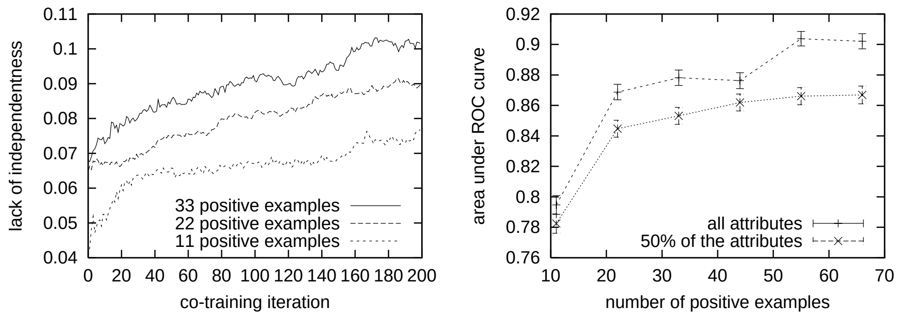


Figure 3: Lack of independence between classifiers and compatibility of attribute subsets.

that for all x: $f^1(x_1) = f^2(x_2) = f(x)$ – and conditional independence of attributes given the class labels – $P(x_1|f(x), x_2) = P(x_1|f(x))$. We will now investigate on a possible relation between the labeled sample size and violation of independence and compatibility assumption.

The compatibility assumption requires the existence of a classifier using either attribute subset. We measure the AUC performance of a classifier with access to all attributes (over varying amounts of labeled training data) versus a classifier that can perceive only 50% of the attributes. We average over 10 different randomly drawn attribute subsets. Figure 3 (right hand side, for "product inquiries") shows the corresponding curves.

We see that the classifier that uses half of the available attributes converges towards an AUC performance of 0.87 whereas a decision based on all attributes achieves an AUC of 0.9. Note that the linear combination of two classifiers that use one half of the available attributes each can of course lead to a higher AUC than each of the individual classifiers (Figure 2 shows that the co-trained hypothesis achieves up to 0.9 AUC). However, as the two individual classifiers label more and more data for their co-training peer, the noise level of the labeled increases to the level of accuracy achieved by the labeling classifier. The observation that 50% of the attributes allow for an accuracy of about 0.87 AUC explains the accuracy deterioration of co-training processes with a starting accuracy above that value.

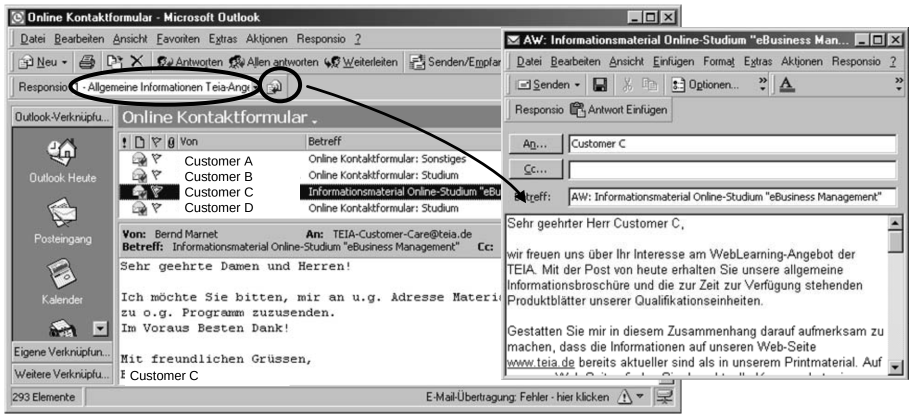
Let us now focus on the assumed independence of the two classifiers. Co-training can only improve classifier performance when, at some point, an error of one peer hypothesis h_1 for an example x is corrected after the other hypothesis h_2 adds the correct label of x to the labeled sample. Let E_1 be a random variable that assumes value 1 when, for an example x, hypothesis h_1 errs; correspondingly, E_2 indicates that h_2 errs for an example x. Co-training is most effective when E_1 and E_2 are statistically independent ($P(E_1, E_2) = P(E_1)P(E_2)$) whereas, when E_1 and E_2 are entirely coupled ($P(E_1 = x|E_2 = x) = 1$), co-training will not have any effect because the hypotheses already agree on every single unlabeled instance.

We measure $P(E_1, E_2) - P(E_1)P(E_2)$, a natural measure of the "lack of independence" between the two classifiers (for the "product inquiries" class), for three different labeled sample sizes and over the co-training iterations. The left hand side of Figure 3 clearly shows that the lack of independence increases with the size of the labeled sample and furthermore increases over the co-training iterations. The lack of independence between the attribute subsets induces a lack of independence between the classifiers that is greater, when more labeled examples are shared. This lack of independence between the classifiers explains why co-training is only beneficial when only very small labeled samples are available.

The benefit of both, co-training and transduction is greatest, when only few labeled data are available. Transduction outperforms co-training for product inquiries; transduction and co-training perform similar for server breakdown. Both, transduction and co-training increase the computation time substantially (from the order of seconds to as many as 20 minutes).

5 The Responsio Email Management System

We have integrated the learning algorithms into the Responsio email management system (Kockelkorn et al., 2003). The key design principle of Responsio is that, once it is installed and the standard answers are entered, it does not require any extra effort from the user. The system observes incoming emails and replies sent, but does not require explicit feedback. Responsio is an add-on to Microsoft Outlook. The control elements (shown in Figure 4) are loaded into Outlook as a COM object.


Online Kontaktformular - Microsoft Outlook
Datei Bearbeiten Ansicht Favoriten Extras Aktionen Responsio ?
Neu Antworten Allen antworten Weiterleiten Senden/Empfan
Responsio 1 - Allgemeine Informationen Teia-Ange
Outlook-Verknüpfu...
Outlook Heute
Posteingang
Kalender
Eigene Verknüpfun...
Weitere Verknüpfu...
Online Kontaktformular
293 Elemente
E-Mail-Übertragung: Fehler - hier klicken
AW: Informationsmaterial Online-Studium "eBusiness Man...
Datei Bearbeiten Ansicht Einfügen Format Extras Aktionen Responsio ?
Senden Optionen...
Responsio Antwort Einfügen
An...
Cc...
Betreff:

Von	Betreff
Customer A	Online Kontaktformular: Sonstiges
Customer B	Online Kontaktformular: Studium
Customer C	Informationsmaterial Online-Studium "eBu
Customer D	Online Kontaktformular: Studium

Von: Bernd Marnet **An:** TEIA-Customer-Care@teia.de
Betreff: Informationsmaterial Online-Studium "eBusiness Management" **Cc:**

```
Sehr geehrte Damen und Herren!

Ich möchte Sie bitten, mir an u.g. Adresse Materi
zu o.g. Programm zuzusenden.
Im Voraus Besten Dank!

Mit freundlichen Grüssen,
I Customer C
```

An: Customer C
Cc:
Betreff: AW: Informationsmaterial Online-Studium "eBusiness Management"

Sehr geehrter Herr Customer C,

wir freuen uns über Ihr Interesse am WebLearning-Angebot der TEIA. Mit der Post von heute erhalten Sie unsere allgemeine Informationsbroschüre und die zur Zeit zur Verfügung stehenden Produktblätter unserer Qualifikationseinheiten.

Gestatten Sie mir in diesem Zusammenhang darauf aufmerksam zu machen, dass die Informationen auf unseren Web-Seite www.teia.de bereits aktueller sind als in unserem Printmaterial. Auf

Figure 4: When an email is read, the most likely answer is displayed in a special field in the Outlook window. On clicking the "Auto-Answer" button, a reply window with the proposed answer text is created.

When an email is selected, the COM add-in sends the email body to a second process which identifies the language of the email, executes the language specific classifiers and determines the posterior probabilities of the configured answers. The classifier process notifies the COM add-in of the most likely answer which is displayed in the field circled in Figure 4. When the user clicks the "auto answer" button (also circled), Responsio extracts first and last name of the sender, identifies the gender by comparing the first names against a list, and formulates a salutation line followed by the proposed standard answer. The system opens a reply window with the proposed answer filled in.

Whenever an email is sent, Responsio identifies whether the outbound mail is a reply to an inbound mail by matching recipient and subject line to sender and subject line of all emails that are visible in one of the Outlook windows. When the sent email includes one of the standard answers, the inbound mail is filed into the list of example mails for that answer. These examples can be viewed in the Responsio manager window. It is also possible to manually drag and drop emails into the example folders. Whenever an example list changes, the training unit starts a process with the learning algorithm.

6 Discussion and Related Results

We have discussed the problem of identifying instances of frequently asked questions in emails, using only stored inbound and outbound emails as training data. Our empirical data shows that identifying a relatively small set of standard questions automatically is feasible; we obtained AUC performance of between 80 and 95% using as little as seven labeled positive examples. The transductive Support Vector Machine and the co-training algorithm utilize the available unlabeled data and improve recognition rate considerably if and only if only few labeled training examples are available.

Unlabeled data provides learning algorithms with additional information on the classification task. It is perhaps intuitive that additional information should improve the learning results but our experience – as well as earlier results by Kiritchenko and Matwin (2002) (using email data) and Denis et al. (2003); Muslea et al. (2002); Cozman et al. (2003) – show that this is *not necessarily* the case. Our investigations show that splitting tf-idf features at random causes a lack of compatibility; furthermore, a lack of independence can be observed that grows with the labeled sample size and number of co-training iterations. Another drawback of both the transductive SVM and co-training lies in a substantial increase in computation time. For use in a desktop application, the efficiency of the transductive SVM would need to be improved.

The Responsio email management system is not intrusive; *i.e.,* the user is not required to provide additional feedback. A limitation of these data sources is that we cannot determine example emails for which no standard answer is appropriate (we cannot decide whether two syntactically different answers are really semantically different). Hence, we can neither estimate P(no standard answer) nor $P(\bar{A}_i)$ for any A_i.

Information retrieval (Baeza-Yates & Ribeiro-Neto, 1999) offers a wide spectrum of techniques to measure the similarity between a question and questions in an FAQ list. While this approach is followed in many FAQ systems, it does not take all the information into account that is available in the particular domain of email answering: emails received in the past. An FAQ list contains only one single instance of each question whereas we typically have many instances of each questions available that we can utilize to recognize further instances of these questions more accurately.

The domain of question answering (Vorhees, 1999) is loosely related to our email answering problem. In our application domain, a large fraction of incoming questions can be answered by very few answers. These answers can be pre-configured; the difficulty lies in recognizing instances of these frequently asked questions very robustly. Question answering systems solve a problem that is in a way more difficult. For arbitrary questions posed in natural language, these systems select an answer sentence from a large corpus, such as an encyclopedia. Applied to email, any grammar-based approach will face the difficulty of ungrammatical mails.

Several email assistance systems have been presented. Green and Edwards (1996); Boone (1998); Segal and Kephart (1999); Crawford et al. (2002) use text classifiers in order to predict the correct folder for an email. In contrast to these studies, we study the feasibility of identifying instances of particular questions rather than general subject categories.

The related problem of filtering spam emails has been studied frequently. Methods based on the naive Bayes algorithm has been explored (Pantel & Lin, 1998; Sahami et al., 1998; Kolcz & Alspector, 2001) and compared to keyword based approaches (Androutsopoulos et al., 2000), rule induction (Cohen, 1996; Provost, 1999) and Support Vector Machines (Drucker et al., 1999). Machine learning of personalized spam classifiers requires the user to manually label emails as spam; for instance, the Netscape email client has been equipped with "this is spam" and "this is not spam" buttons. By contrast, our approach generates additional examples for the email answering task by observing inbound and outbound messages.

ACKNOWLEDGMENT

The author is supported by the German Science Foundation (DFG) under grant SCHE540/10-1. Michael Kockelkorn, Andreas Lüneburg, and Clint Heyer contributed to the implementation that was also supported by Michael May and Stefan Wrobel of the Fraunhofer Institute of Autonomous Intelligent Systems, and by Hewlett Packard Consulting, Germany.

References

Androutsopoulos, I., Koutsias, J., Chandrinos, K., & Spyropoulos, C. (2000). An experimental comparison of naive bayesian and keyword based anti-spam filtering with personal email messsages. *Proceedings of the International ACM SIGIR Conference*.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

Bennett, P. (2000). *Assessing the calibration of naive Bayes' posterior estimates*. Technical Report. CMU.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the Conference on Computational Learning Theory* (pp. 92–100).

Boone, T. (1998). Concept features in Re:Agent, an intelligent email agent. *Autonomous Agents*.

Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Cohen, W. (1996). Learnig rules that classify email. *Proceedings of the IEEE Spring Symposium on Machine learning for Information Access*.

Cooper, D., & Freeman, J. (1970). On the asymptotic improvement in the outcome of supervised learning provided by additional nonsupervised learning. *IEEE Transactions on Computers*, *C-19*, 1055–1063.

Cozman, F., Cohen, I., & Cirelo, M. (2003). Semi-supervised learning of mixture models. *Proceedings of the International Conference on Machine Learning* (pp. 99–106).

Crawford, E., Kay, J., & McCreath, E. (2002). IEMS - the intelligent email sorter. *Proceedings of the International Conference on Machine Learning*.

Denis, F., Laurent, A., Gilleron, R., & Tommasi, M. (2003). Text classification and co-training from positive and unlabeled examples. *ICML Workshop on the Continuum from Labeled to Unlabeled Data*.

Drucker, H., Wu, D., & Vapnik, V. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, *10*.

Green, C., & Edwards, P. (1996). Using machine learning to enhance software tools for internet information management. *Proceedings of the AAAI Workshop on Internet Information Management*.

Joachims, T. (1999a). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Joachims, T. (1999b). Transductive inference for text classification using support vector machines. *Proceedings of the International Conference on Machine Learning* (pp. 200–209).

Kiritchenko, S., & Matwin, S. (2002). *Email classification with co-training.* Technical Report. University of Ottawa.

Kockelkorn, M., Lüneburg, A., & Scheffer, T. (2003). Learning to answer emails. *Proceedings of the International Symposium on Intelligent Data Analysis.*

Kolcz, A., & Alspector, J. (2001). SVM-based filtering of e-mail spam with content-specific misclassification costs. *Proceedings of the ICDM Workshop on Text Mining.*

Lewis, D. (1997). The TREC-5 filtering track. *Proceedings of the Fifth Text Retrieval Conference.*

McCallum, A., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. *Proceedings of the International Conference on Machine Learning.*

Muslea, I., Kloblock, C., & Minton, S. (2002). Active + semi-supervised learning = robust multi-view learning. *Proceedings of the International Conference on Machine Learning* (pp. 435–442).

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *Proceedings of Information and Knowledge Management.*

Pantel, P., & Lin, D. (1998). Spamcop: a spam classification and organization program. *Proceedings of the AAAI Workshop on Learning for Text Categorization.*

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing inductive algorithms. *Proceedings of the International Conference on Machine Learning* (pp. 445–453).

Provost, J. (1999). *Naive Bayes vs. rule-learning in classification of email.* (Technical Report AI-TR-99-284). University of Texas at Austin.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk email. *Proceedings of the AAAI Workshop on Learning for Text Categorization.*

Seeger, M. (2001). *Learning with labeled and unlabeled data.* Technical Report. University of Edinburgh.

Segal, R., & Kephart, J. (1999). MailCat: An intelligent assistant for organizing mail. *Autonomous Agents.*

Vapnik, V. (1998). *Statistical Learning Theory.* Wiley.

Vorhees, E. (1999). The TREC-8 question answering track report. *Proceedings of TREC-8.*