Using Transduction and Multi-view Learning to Answer Emails

Michael Kockelkorn[1], Andreas Lüneburg[1], and Tobias Scheffer[2]

[1] Tonxx, Berlin, Germany
{mk,alueneburg}@tonxx.com
[2] Humboldt Univerersity, School of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
scheffer@iws.cs.uni-magdeburg.de

Abstract. Many organizations and companies have to answer large amounts of emails. Often, most 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. Our approach effectively utilizes the data that is typically available in this setting: inbound and outbound emails stored on a server. We take into account that there are no explicit links between inbound and corresponding outbound mails on the server. We map the problem to a semi-supervised classification problem that can be addressed by algorithms such as the transductive support vector machine and multi-view learning. We evaluate our approach using emails sent to a corporate customer service department.

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, many of them 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 reply 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 stored on an email server. 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.,* [8]), we explore whether text classification algorithms can identify instances of a specific frequently asked question.

N. Lavrač et al. (Eds.): PKDD 2003, LNAI 2838, pp. 266–277, 2003.

Many approaches are known that learn text classifiers from data. The support vector machine (SVM) (*e.g.,* [10]) is generally considered to be one of the most accurate algorithms; this is supported, for instance, by the TREC filtering challenge [13]. The naive Bayes algorithm is also widely used for text classification.

Among the known algorithms that utilize unlabeled data, the transductive SVM and the multi-view framework apply for support vector learning. The transductive SVM [9] maximizes the distance between hyperplane and both, labeled and unlabeled data. In multi-view learning [3], two classifiers which use different attribute sets provide each other with labels for the unlabeled data.

The contribution of this paper is threefold. Firstly, we analyze the problem of answering emails, taking all practical aspects into account. Secondly, we present a case study on a practically relevant problem showing how well the naive Bayes algorithm, the support vector machine, the transductive support vector machine, and the co-training multi-view algorithm can identify instances of particular questions in emails. Thirdly, we describe how we integrated machine learning algorithms into a practical answering assistance system that is easy to use and provides immediate user benefit.

The rest of this paper is organized as follows. In Section 2, we analyze the problem setting. 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. In Section 4, we briefly describe the transductive SVM and the multi-view algorithm that we used for the case study that is presented in Section 5. In Section 6 we describe how we have integrated our learning approach into the Responsio email management system. Section 7 discusses related approaches.

2 Problem Setting

We consider the problem of predicting which of n (manually identified) standard answers $A_1, \ldots, A_n$ a user will reply to an email. In order to learn a predictor, we are 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 (at least) thousands of emails stored on a corporate email server.

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 equivalent to one of the answers A_i but uses a few different words.

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 an answer like "we apologize for experiencing technical problems..."; or consider

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? Out 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. Keeping these characteristics in mind, we can pose the problem which we want to solve as follows.

Problem 1. Given is a repository X of inbound emails and a repository Y of outbound emails in which instances of standard answers $A_1, \ldots, A_n$ occur. There is no explicit mapping between inbound and outbound mails and the prior probabilities $P(A_i)$ are non-stationary. The task is to generate a predictor for the most likely answer A_i to a new inbound email x.

3 Underlying Learning Problem

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 $\text{argmax}_i P(A_i|x) = \text{argmax}_i P(x|A_i)P(A_i)$ 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 $\sum_i P(A_i) = 1$; when the answer can be any subset of $\{A_1, \ldots, A_n\}$, then $P(A_i) + P(\bar{A}_i) = 1$ for each answer A_i.

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.

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

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.

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, either the inbound mail has to be quoted in the outbound mail, or the outbound mail has to be sent while the inbound mail was 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 as positive examples for $A_1, \ldots, A_n$.

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.

Text classifiers typically return 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 independent of all $f_j(x)$ for $i \neq j$. 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.

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

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. Following [2], we assume Gaussian likelihoods $P(f_i(x)|A_i)$ and estimate the μ_i, $\mu_{\bar{i}}$ and σ_i in a cross validation loop as follows. In each cross validation fold, we record the $f_i(x)$ for all held-out positive and negative instances. After that, we estimate μ_i, $\mu_{\bar{i}}$ and σ_i from the recorded decision function values of all examples. It is well known that Bayes' rule applied to a Gaussian likelihood yields a sigmoidal posterior; Equation 4 corresponds to Equation 3 for continuous $f_i(x)$ and Gaussian $P(f_i(x)|A_i)$.

$$P(A_i|f_i(x)) = \left(1 + e^{\frac{\mu_{\bar{i}} - \mu_i}{\sigma^2} f_i(x) + \frac{\mu_{\bar{i}}^2 - \mu_i^2}{2\sigma^2} + log\frac{1-P(A_i)}{P(A_i)}}\right)^{-1} \quad (4)$$

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 $f_i : X \to real$ for each of the answers A_i.

We considered a Naive Bayes classifier and the support vector machine SVM^{light} [9]. Both classifiers use the bag-of-words representation which con-

siders 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.

4 Using Unlabeled Data

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 learned from data $((x_1, y_1), \ldots, (x_m, y_m))$. Note that $\frac{w}{|w|}x_i + b$ is the distance between plain (w, b) and instance x_i; this margin is positive for positive examples ($y_i = +1$) and negative for negative examples ($y_i = -1$). Equivalently, $y_i(\frac{w}{|w|}x_i + b)$ is the positive margin for both positive and negative examples.

The optimization problem which the SVM learning procedure solves is to find w and b such that $y_i(wx_i + b)$ is positive for all examples (all instances lie on the "correct" side of the plain) and the smallest margin (over all examples) is maximized. Equivalently to maximizing $y_i(\frac{w}{|w|}x_i + b)$, it is usually demanded that $y_i(wx_i + b) \geq 1$ for all (x_i, y_i) and $|w|$ be minimized.

Optimization Problem 1 *Given data* $((x_1, y_1), \ldots, (x_m, y_m))$*; over all* w*,* b*, minimize* $|w|^2$*, subject to the constraint* $\forall_{i=1}^{m} y_i(wx_i + b) \geq 1$*.*

The SVM^{light} software package [9] implements an efficient optimization algorithm which solves optimization problem 1. The transductive support vector machine (TSVM) [10] furthermore considers unlabeled data. 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 hyperplain is maximized. However, only for the labeled examples we know on which side of the hyperplain the instances have to lie.

Optimization Problem 2 *Given labeled data* $((x_1, y_1), \ldots, (x_m, y_m))$ *and unlabeled data* $(x_1^*, \ldots, x_k^*)$*; over all* w*,* b*,* $(y_1^*, \ldots, y_k^*)$*, minimize* $|w|^2$*, subject to the constraints* $\forall_{i=1}^{m} y_i(wx_i + b) \geq 1$ *and* $\forall_{i=1}^{m} y_i^*(wx_i^* + b) \geq 1$*.*

The TSVM algorithm which solves optimization problem 2 is related to the EM algorithm. TSVM starts by learning parameters from the labeled data and labels the unlabeled data using these parameters. It iterates a training step (corresponding to the "M" step of EM) and switches the labels of the unlabeled data such that optimization criterion 2 is maximized (resembling the "E" step). The TSVM algorithm is described in [9].

Table 1. Co-training algorithm.

Given positive examples $(x_1, x_2, +)$, negative examples $(x_1, x_2, -)$ and unlabeled examples in two different views V_1 and V_2; number of iterations k.

1. Loop for k iterations
   (a) Train f_1 and f_2 using the labeled positive and negative examples.
   (b) Let f_1 and f_2 select the positive and negative example for which they make the most confident prediction. Remove the examples from the unlabeled data and add them to the labeled data.
2. Return the combined classifier $f(x) = f_1(x_1) + f_2(x_2)$.

4.2 Multi-view Learning

Blum and Mitchell [3] have proposed the multi-view approach to utilizing unlabeled data. In multi-view learning, 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 co-training algorithm is the most prominent multi-view algorithm. 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 converts unlabeled examples into randomly drawn labeled examples [3].

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 examples. 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 1.

The compatibility and independence assumptions are usually violated in practice. However, empirical studies [14,11] show that co-training can nevertheless improve performance. In particular, text classification problems seem to be particularly suited for co-training [15]. In our experiments, we use co-training in association with SVM^{light}.

5 Case Study

The data used in this study 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 2 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.

Table 2. 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%

We briefly summarize the basic principles of ROC analysis which we used to assess the decision functions [5,17]. 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 the ability of a decision function to separate positive from negative examples.

In order to estimate the AUC performance and its standard deviation for a decision function, we performed between 7 and 20-fold stratified cross validation and averaged the AUC values measured on the held out data. In order to plot the actual ROC curves, we also performed 10-fold cross validation. In each fold, we filed the decision function values of the held out examples into one global histogram for positives and one histogram for negatives. After 10 folds, we calculated the ROC curves from the resulting two histograms.

First, we studied the performance of a decision function provided by the Naive Bayes algorithm (which is used, for instance, in the commercial Autonomy Answer system) as well as the support vector machine SVM^{light} [9]. 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 [2]. This implies that Naive Bayes performs poorly when not all documents are equally long, as is the case here.

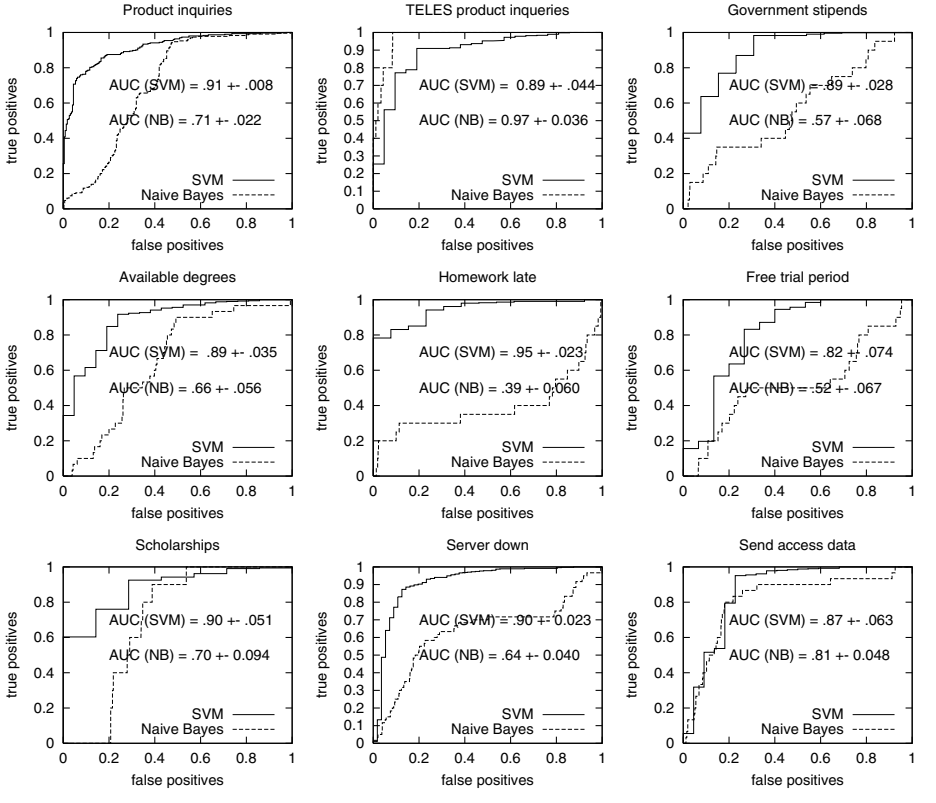

Fig. 1. ROC curves for nine most frequently asked questions of naive Bayes and the support vector machine.

In the next set of experiments, we observed how the transductive support vector machine improves 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 SVM performance (only the labeled data is used by the SVM) to the performance of the transductive SVM (using both labeled and unlabeled data). Table 3 shows the results for category "general product inqueries"; Table 4 for "server breakdown".

When the labeled sample is of size at least $24 + 33$ for "general product inqueries", or $10 + 30$ for "server breakdown", then SVM and transductive SVM

Table 3. SVM and transductive SVM, "general product inqueries".

Labeled data	SVM (AUC)	TSVM (AUC)
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 4. 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

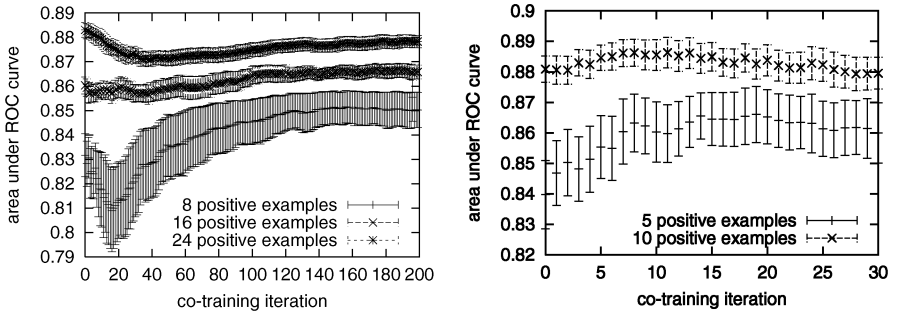

Fig. 2. Change of AUC performance with increasing numbers of co-training iterations.

perform equally well. When the labeled sample is smaller, then the transductive SVM outperforms the regular SVM significantly. We can conclude that 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 inqueries") or 5+15 examples ("server breakdown"), respectively, still outperforms the naive Bayes algorithm *with all available data.*

Finally, we want to study how the performance changes when we use co-training in association with the Support Vector Machine. Figure 2 shows the AUC performance against the number of co-training iteration. The results resemble those obtained with the TSVM: Co-training improves performance only when at most 16 positive examples are available for product inqueries and when at most 5 positive examples are available for server breakdown. The benefit of both, co-training and transduction is greatest, when only few labeled data are available. Transduction outperforms co-training for product inqueries; transduction and co-training perform similar for server breakdown.

6 The Responsio Email Management System

We integrated the learning algorithms into an email assistance system. The key design principle is that, once the standard answers are entered, it does not require

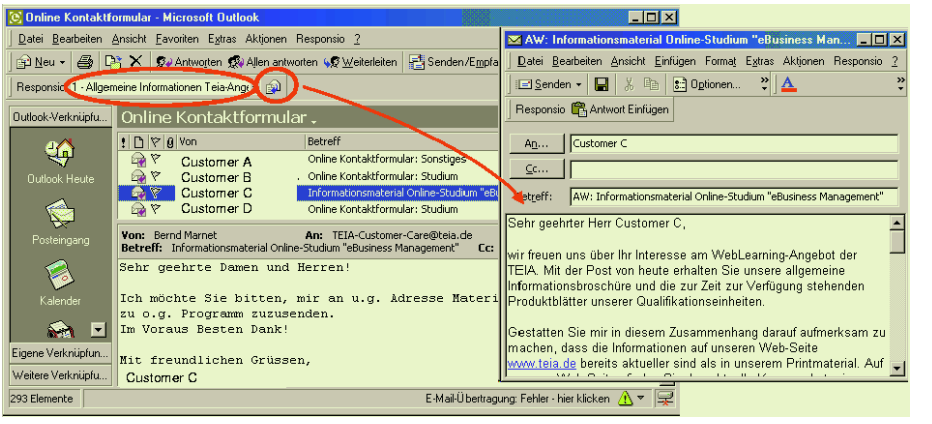

Fig. 3. When an email is read, the most likely answer is displayed in a special field in the Outlook window. On clicking the "Auto-Answer" button, a reply window with the proposed answer text is created.

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 (Figure 3) are loaded as a COM object.

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 marked. When the user clicks the "auto answer" button (circled in Figure 3), 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.

7 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 ques-

tions 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. The drawback of both semi-supervised algorithms is the increase in computation time from few seconds to several minutes. For use in a desktop application, efficiency is a crucial factor.

A limitation of the available data sources is that we cannot determine examples of emails for which no A_i is appropriate (we cannot decide whether two syntactically different answers are really semantically different). Therefore, we can neither estimate $P(\text{no standard answer})$ nor $P(\bar{A}_i)$ for any A_i.

Information retrieval 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 [20] is rather 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 robustly, even in very ungrammatical emails. Question answering systems solve a problem that is in a way more difficult: selecting an answer sentence from a large corpus (such as an encyclopedia) for arbitrary questions.

Several email assistance systems have been presented. [8,4,19,7] 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.

Related is the problem of filtering spam email. Keyword based approaches, Naive Bayes [1,16,18,12] and rule-based approaches [6] have been compared. Generating positive and negative examples for spam requires additional user interaction: the user might delete interesting emails just like spam after reading it. By contrast, our approach generates examples for the email answering task without imposing additional effort on the user.

Acknowledgment

We have received support from the German Science Foundation (DFG) under grant SCHE540/10-1, and from Hewlett Packard Consulting, Germany.

References

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

2. P. Bennett. Assessing the calibration of naive bayes' posterior estimates. Technical report, CMU, 2000.
3. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*, 1998.
4. T. Boone. Concept features in Re:Agent, an intelligent email agent. *Autonomous Agents*, 1998.
5. A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
6. W. Cohen. Learnig rules that classify email. In *Proceedings of the IEEE Spring Symposium on Machine learning for Information Access*, 1996.
7. E. Crawford, J. Kay, and E. McCreath. IEMS - the intelligent email sorter. In *Proceedings of the International Conference on Machine Learning*, 2002.
8. C. Green and P. Edwards. Using machine learning to enhance software tools for internet information management. In *Proceedings of the AAAI Workshop on Internet Information Management*, 1996.
9. T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, 1999.
10. T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, 1999.
11. S. Kiritchenko and S. Matwin. Email classification with co-training. Technical report, University of Ottawa, 2002.
12. A. Kolcz and J. Alspector. Svm-based filtering of e-mail spam with content-specific misclassification costs. In *Proceedings of the ICDM Workshop on Text Mining*, 2001.
13. D. Lewis. The trec-5 filtering track. In *Proceedings of the Fifth Text Retrieval Conference*, 1997.
14. I. Muslea, C. Kloblock, and S. Minton. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the International Conference on Machine Learning*, 2002.
15. K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of Information and Knowledge Management*, 2000.
16. P. Pantel and D. Lin. Spamcop: a spam classification and organization program. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*, 1998.
17. F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation in comparing classifiers. In *Proceedings of the International Conference on Machine Learning*, 1998.
18. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk email. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*, 1998.
19. R. Segal and J. Kephart. Mailcat: An intelligent assistant for organizing mail. In *Autonomous Agents*, 1999.
20. E. Vorhees. The trec-8 question answering track report. In *Proceedings of TREC-8*, 1999.