Mach Learn (2007) 69: 55–74
DOI 10.1007/s10994-007-5023-9

Extending boosting for large scale spoken language understanding

Gokhan Tur

Received: 19 August 2005 / Revised: 10 June 2007 / Accepted: 28 August 2007 /
Published online: 25 September 2007
Springer Science+Business Media, LLC 2007

Abstract We propose three methods for extending the Boosting family of classifiers motivated by the real-life problems we have encountered. First, we propose a semisupervised learning method for exploiting the unlabeled data in Boosting. We then present a novel classification model adaptation method. The goal of adaptation is optimizing an existing model for a new target application, which is similar to the previous one but may have different classes or class distributions. Finally, we present an efficient and effective cost-sensitive classification method that extends Boosting to allow for weighted classes. We evaluated these methods for call classification in the AT&T VoiceTone® spoken language understanding system. Our results indicate that it is possible to obtain the same classification performance by using 30% less labeled data when the unlabeled data is utilized through semisupervised learning. Using model adaptation we can achieve the same classification accuracy using less than half of the labeled data from the new application. Finally, we present significant improvements in the "important" (i.e., higher weighted) classes without a significant loss in overall performance using the proposed cost-sensitive classification method.

Keywords Boosting · Semisupervised learning · Model adaptation · Cost-sensitive classification · Call classification · Spoken dialog systems

1 Introduction

Statistical classification algorithms have long been studied in the machine learning community. Typically, classification models are trained using large amounts of task data that are usually labeled by humans. By "labeling", we mean assigning one or more of the predefined classes to each example. Building better classification systems in a shorter time frame is a

Editor: Dan Roth.

This work was done when the author was with AT&T Labs-Research, Florham Park, NJ 07932.

G. Tur (✉)
SRI International Speech Technology and Research Lab, Menlo Park, CA 94025, USA
e-mail: gokhan@speech.sri.com

need for most real-world applications. For instance, consider a natural language call routing system where the aim is to route the incoming calls in a customer care call center. In such a system the aim is to identify the customer's intent (call-type), which is basically framed as a call classification problem. Consider the utterance *I would like to know my account balance*. Assuming that the utterance is recognized correctly by an automatic speech recognizer, the corresponding call-type would be *Tellme(Balance)* and the action would be prompting the balance to the user or routing this call to the billing department. In cases where numerous such systems for different call centers from different domains or companies need to be built, preparing labeled data for each system is a very expensive, time consuming, and laborious process. Assuming that the bottleneck is not the collection of data but instead labeling it, we propose using semisupervised learning.

Another problem with most classification algorithms is that they make the assumption that the input channel is stationary, that is, the distribution of the incoming (training and test) examples is constant. Although keeping the training and test sets fixed may be a good idea in comparing different classification approaches, for applications like those described above, the stationary distribution assumption may not always be true. For example, if the corresponding company introduces a new service, one may expect callers to start inquiring about that service, which may be unseen or very infrequent in the training data. Continuous human labeling of new data and model retraining alleviates this problem. However, adaptation to time-varying statistics by incrementing the effect of a small set of newly labeled examples might be a more cost-effective and faster solution. Likewise, in some cases there may be a new application very similar to an existing one, such as product solutions for two companies from the same sector. Therefore, the old models can be adapted to the new application. In this paper, we propose using model adaptation techniques for this problem.

Furthermore, typically in the proposed classification algorithms all classes have the same importance or cost for multiclass classification. For most real-world applications, this is not the case; not all classes have the same weight. The accuracy of a particular class can be more critical than others or high precision may be needed for some classes. Especially while dealing with more than a few classes, this is unavoidable. For the call routing example, misclassifying an utterance asking for an account balance as a request for cancellation is more costly than the other way around, as it may result in a lost customer. In this paper, we propose a cost-sensitive classification approach.

Our proposed methods depend on a particular classification algorithm: namely Boosting. Boosting is an iterative algorithm; on each iteration, a weak classifier is trained on a weighted training set, and at the end, the weak classifiers are combined into a single, combined classifier (Freund and Schapire 1997). The Boosting algorithm has been used successfully for many classification tasks, such as text categorization (Schapire and Singer 2000). We propose the following methods for extending the Boosting family of classifiers:

– *Semisupervised Learning*: The goal is to exploit the unlabeled data in Boosting. We propose a method for augmenting the classification model trained using the human-labeled examples with the machine-labeled examples in a weighted manner. The machine-labeled examples are automatically constructed using the labels output by the model trained with human-labeled examples. The resulting model is actually a superset of weak-learners of the initial model. The new weak learners are learned once the initial ones are applied to all the human- and machine-labeled data. We also compare this approach with a task-independent baseline approach where human-labeled data is simply concatenated to the machine-labeled data.
– *Model Adaptation*: The goal is to adapt an existing model to a new target application, which is similar but may have different classes or class distributions. This may also in-

clude continuous adaptation of an existing model to time-varying statistics or exploiting out-of-domain data for training the target model. The idea is similar to the semisupervised learning technique. Boosting applies the existing model to the new data, and then new weak learners are added to this existing model using the new data in a weighted manner.
- *Cost-Sensitive Classification*: The goal is to extend Boosting to allow weighted classes. Our main idea is to change the error function considering the weights of the classes, thus ensuring optimal accuracy at each iteration for the "important" classes. In other words, we have changed the criterion to choose the weak learner according to the associated costs of the classes.

In the following section we summarize the previous related work on all three of these areas. We then briefly describe Boosting in Sect. 3 since all extensions require modifications to the original algorithm. Sections 4, 5, and 6 present our proposed methods for semisupervised learning, model adaptation, and cost-sensitive classification, respectively. In Sect. 7 we present our results using a call classification task from the AT&T VoiceTone® spoken dialog system.[1]

2 Related work

The following describes the related work for each topic of our research.

2.1 Boosting for call classification and text categorization

Boosting, the machine learning algorithm we focus in this paper, has been used for a number of language processing tasks, such as call classification and text categorization. Schapire and Singer have presented empirical results comparing Boosting with other state-of-the-art classification methods (Schapire and Singer 2000). For example, for text categorization Boosting outperformed other methods such as Rocchio, Naive Bayes, k-nearest-neighbor (KNN), RIPPER, and Sleeping Experts on the newswire data, especially when numerous examples are available for training. They have also provided the first experiments on using Boosting for a simple call routing task with only six classes.

A later Boosting algorithm was used for the BBN call routing system (Zitouni et al. 2001). In this 23-way classification task, an additional 10% improvement has been observed compared to the previously best classification method, named Beta classification.

AT&T has successfully used Boosting in large-scale call classification systems, first with a Helpdesk application (Di Fabbrizio et al. 2002) and later for its enterprise customers as part of the AT&T VoiceTone® spoken dialog system (Gupta et al. 2006). The algorithm was then extended to handle manually written call classification rules to augment the labeled data in special cases (Schapire et al. 2005). This paper has emerged from the apparent needs of extensions to the basic Boosting algorithm during the real-life application developments for VoiceTone.

[1] The VoiceTone® system is provided by AT&T for customer care centers.

2.2 Semisupervised learning

Semisupervised learning algorithms that use both labeled and unlabeled data have been used for classification in order to reduce the need for labeled training data. Blum and Mitchell (1998) proposed a semisupervised learning approach called co-training. For co-training, the features in the problem domain should naturally divide into two sets. Then the examples classified with high confidence scores with one view can be used as the training data of other views. For example, for web page classification, one view can be the text in the web pages and another view can be the text in the hyperlinks pointing to those web pages. For the same task, Nigam et al. (2000) used an algorithm for learning from labeled and unlabeled documents based on the combination of the Expectation Maximization (EM) algorithm and a Naive Bayes classifier. Nigam and Ghani (2000) then combined co-training and EM algorithms, coming up with the Co-EM algorithm, which is the probabilistic version of co-training. Ghani (2002) later combined the Co-EM algorithm with error-correcting output coding (ECOC) to exploit the unlabeled data, in addition to the labeled data. For spoken language understanding, we have presented semisupervised learning approaches for exploiting unlabeled data (Tur and Hakkani-Tür 2003). Note that our focus is different from training call routing systems with automatic speech recognizer (ASR) output instead of manual transcriptions such as (Iyer et al. 2002; Alshawi 2003). In this paper we present how Boosting can be extended to augment an existing model using unlabeled data.

2.3 Model adaptation

Although statistical model adaptation has been well studied in some specific areas such as speech recognition for acoustic and language modeling (Riccardi and Gorin 2000; Bacchiani et al. 2004; Digalakis et al. 1995, among others), there is comparably less work done on machine learning and natural language processing. We believe this is the first study presenting model adaptation for language understanding. One recent study is on the adaptation of natural language understanding using a common adaptation method of *maximum a posteriori* (MAP) adaptation (He and Young 2004), which adapts the hidden vector state model built for an airline travel information application (ATIS) to another (DARPA Communicator). Another study is about supervised and unsupervised adaptation of probabilistic context-free grammars to a new domain again using MAP adaptation (Roark and Bacchiani 2003). For spoken language understanding, we have proposed a model adaptation approach using Boosting (Tur 2005).

In the machine learning literature, model adaptation has been studied under the titles of meta-learning (Prodromidis and Stolfo 1998) and multitask learning (Caruana 1997). Meta-learning aims at learning useful information from large and inherently distributed sources (in our case, applications). Given multiple classification models trained using local data, the goal is to train a meta-level classifier combining all these baseline models using a meta-level training set. Multitask learning aims at training tasks (in our case, applications) in parallel while using a shared representation. What is learned for each task can help other tasks be learned better.

2.4 Cost-sensitive classification

In the machine learning literature, vast majority of the classifiers, including Boosting, does not handle weighted classes automatically, a lack that we address in this work. There are a number of studies explicitly attacking this and they fall into three categories:

- Making existing classifiers cost-sensitive (Tur 2004; Drummond and Holte 2000; Fan et al. 1999). For example, Fan et al. (1999) have proposed the AdaCost algorithm which extends the AdaBoost algorithm giving weights to individual training examples instead of classes.
- Using Bayes risk theory to assign each example to its lowest risk class (Domingos 1999; Margineantu 2002; Zadrozny and Elkan 2001).
- Changing the class distributions in the training set such that the cost-insensitive classifier learned will perform equally to a cost sensitive classifier learned from the original set (Zadrozny et al. 2003).

While the first approach is classifier dependent, it may be more effective for some classification algorithms. The latter two approaches are classifier independent. The third approach, while practical, requires either loss of existing training data (for down-sampling) or replication of it (up-sampling) and hence may be suboptimal for some of the classification algorithms. The second approach requires the classifier to output a probability for each of the classes.

In the cases using Bayes risk theory, a cost matrix C is usually used, where the entry (i, j) is the cost of predicting class i when the true class is j. Then the Bayes optimal prediction for a sample x is the class that minimizes the expectation of the cost, or the conditional risk (Duda and Hart 1973):

$$R(i|x) = \sum_j P(j|x)C(i, j) \tag{1}$$

where $P(j|x)$ is the probability of sample x to be class j. Using this formula of the conditional risk demands good estimation of the class probabilities. For classifiers that output scores for classes, calibration can be used to estimate the probabilities, so cost-insensitive classifiers that have been well developed, such as SVM, Boosting, and so on can be used.

3 Boosting

We propose methods for using Boosting for semisupervised learning, model adaptation, and cost-sensitive classification. Before presenting these methods, we need to present the Boosting algorithm in detail. Boosting is an iterative algorithm; on each iteration, t, a weak classifier, h_t, is trained on a weighted training set, and at the end, the weak classifiers are combined into a single, combined classifier (Freund and Schapire 1997). For example, for text categorization, one can use word n-grams as features, and each weak classifier (e.g., decision stump, which is a single node decision tree) can check the absence or presence of an n-gram. The algorithm generalized for multiclass and multilabel classification is as follows:

Let $\mathcal{X} = x_1, \ldots, x_m$ denote the domain of possible training examples and $\mathcal{Y}$ be a finite set of classes of size $|\mathcal{Y}| = k$. For $Y \subseteq \mathcal{Y}$, let $Y[l]$ for class $l \in \mathcal{Y}$ be

$$Y[l] = \begin{cases} +1 & \text{if } l \in Y, \\ -1 & \text{otherwise} \end{cases}$$

for each example. The algorithm begins by initializing a uniform weight distribution $D_1(i, l)$ over training examples i and labels l. $D_t(i, l)$ indicates the weight of a training sample, i for the class, l. After each round this weight distribution is updated so that the example-class combinations that are easier to classify get lower weights and vice versa. The intended effect is to force the weak learning algorithm to concentrate on the examples and labels that will be the most beneficial to the overall goal of finding a highly accurate classification rule. More formally, the algorithm is as follows:

- Given training data from the instance space $S = \{(x_1, Y_1), \dots, (x_m, Y_m)\}$ where $x_i \in \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$.
- Initialize the distribution $D_1(i, l) = \frac{1}{mk}$.
- For each iteration $t = 1, \dots, T$ do
  - Train a base learner $h_t : \mathcal{X} \to \mathbb{R}$ using distribution D_t.
  - Update

$$D_{t+1}(i, l) = \frac{D_t(i, l) e^{-\alpha_t Y_i[l] h_t(x_i, l)}}{Z_t}$$

  where Z_t is a normalization factor

$$Z_t = \sum_{l \in \mathcal{Y}} \sum_{i=1}^{m} D_t(i, l) e^{-\alpha_t Y_i[l] h_t(x_i, l)}$$

  and α_t is the weight of the base learner.
- Output of the final classifier is then defined as

$$f(x, l) = \sum_{t=1}^{T} \alpha_t h_t(x, l).$$

The Boosting algorithm is independent of the weak classifiers employed. It is assumed that at each iteration a weak classifier h_t is trained using samples associated with classes. Each sample may have a different weight, hence the distribution D_t. The final score for each class, $f(x, l)$ is then simply defined as the weighted summation of the individual weak learners' scores, h_t. α_t is the weight of each weak learner. Although there may be many ways to compute this weight, one of them can be using error rate, ϵ_t of each weak learner:

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}.$$

Schapire and Singer (1999) have proved a bound on the empirical *Hamming loss* (HL) of H in the Boosting algorithm. Hamming loss is defined as the fraction of examples, i, and labels, l, for which the sign of $f(x_i, l)$, $H(x_i, l) \in \{-1, 1\}$, is different from $Y_i[l]$.

Theorem 1

$$HL(H) \leq \prod_{t=1}^{T} Z_t \tag{2}$$

where

$$HL(H) = \frac{1}{mk} |i, l : H(x_i, l) \neq Y_i[l]| = \frac{1}{mk} \sum_{l} \sum_{i} \delta(x_i, l) \tag{3}$$

where

$$\delta(x_i, l) = \begin{cases} 1 & \textit{if } H(x_i, l) \neq Y_i[l], \\ 0 & \textit{otherwise.} \end{cases}$$

Proof By unraveling the update rule, we have that

$$D_{T+1}(i+1) = \frac{e^{-Y_i[l]f(x_i,l)}}{mk\prod_t Z_t}. \tag{4}$$

Moreover, if $H(x_i,l) \neq Y_i[l]$ then $Y_i[l]f(x_i,l) \leq 0$ implying that $e^{-Y_i[l]f(x_i,l)} \geq 1$. Thus,

$$\delta(x_i,l) \leq e^{-Y_i[l]f(x_i,l)}. \tag{5}$$

Combining (3), (4), and (5), we get

$$\begin{aligned}\frac{1}{mk}\sum_l\sum_i \delta(x_i,l) &\leq \frac{1}{mk}\sum_l\sum_i e^{-Y_i[l]f(x_i,l)} \\ &= \sum_l\sum_i\left(\prod_{t=1}^T Z_t\right)D_{T+1}(i,l) = \prod_{t=1}^T Z_t. \quad \square\end{aligned} \tag{6}$$

Theorem 1 is important since it contains the loss function Boosting tries to minimize and the criterion weak learners need to minimize. For semisupervised learning and model adaptation the goal is to change the loss function to minimize and for the cost-sensitive classification the goal is to change the weak learner selection criterion.

As seen from (6), this algorithm can be seen as a procedure for finding a linear combination of base classifiers that attempts to minimize an exponential loss function (Schapire and Singer 1999), which in this case is:

$$\sum_i\sum_l e^{-Y_i[l]f(x_i,l)}. \tag{7}$$

An alternative would be to minimize a logistic loss function as suggested by (Friedman et al. 2000), namely

$$\sum_i\sum_l \ln(1+e^{-Y_i[l]f(x_i,l)}). \tag{8}$$

The confidence of a class, l, for example, x_i is then computed with a logistic function, $\rho()$, as

$$P(Y_i[l]=+1|x_i) = \rho(f(x_i,l)) = \frac{1}{1+e^{-Kf(x_i,l)}} \tag{9}$$

where K is 1 for logistic loss, and 2 for exponential loss (Collins et al. 2002). A more detailed explanation and analysis of this algorithm can be found in (Schapire 2001).

4 Semisupervised learning

The aim of semisupervised learning is to exploit the unlabeled examples for a statistical classification task. We assume that there is some amount of training data available for training an initial classifier. The basic idea is to use this classifier to label the unlabeled data automatically, and improve the classifier performance using the machine-labeled examples, thus reducing the amount of human-labeling effort necessary to come up with better statistical systems.

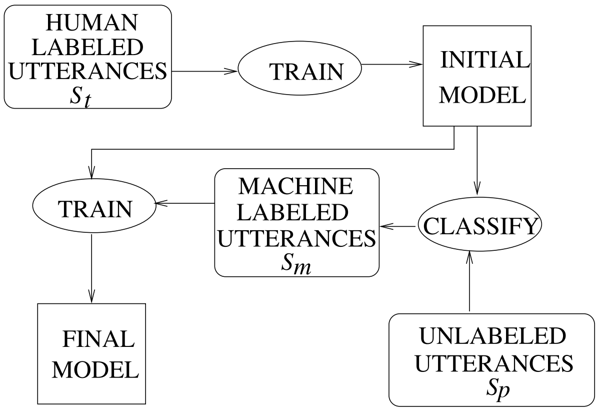
HUMAN LABELED UTTERANCES S_t
TRAIN
INITIAL MODEL
TRAIN
MACHINE LABELED UTTERANCES S_m
CLASSIFY
FINAL MODEL
UNLABELED UTTERANCES S_p

Fig. 1 Semisupervised learning framework

The simplest method for semisupervised learning is augmenting the human-labeled data with machine-labeled data. In this method, first an initial model is trained using the human-labeled data, which is then used to classify the unlabeled data. Then we add the unlabeled examples directly to the training data, by using the machine-labeled classes. In order to reduce the noise added because of classifier errors, we add only those examples that are classified with a confidence higher than some threshold. This threshold can be set using a separate held-out set. Then whole data including both human- and machine-labeled examples are used for training the classifier again. We will evaluate and compare this method with our proposed method in Sect. 7.

Figure 1 depicts the process proposed for semisupervised learning. This method is similar to incorporating prior knowledge into Boosting (Schapire et al. 2005). In that work, a model fitting both the human-labeled training data and the task knowledge is trained. In our case, the aim is to train a model that fits both the human-labeled and machine-labeled data. In that sense, this is actually an application of that work for semisupervised learning. Fitting a model and a data set is implemented as follows: We first train an initial model, using the human-labeled data. Then, the Boosting algorithm measures the fit to the machine-labeled data and the fit to the initial model. To measure the fit to the machine-labeled data, the algorithm uses the logistic loss as given by (8). The fit to the initial model is measured using the Kullback–Leibler (KL) divergence (or binary relative entropy). More formally, the algorithm now aims to minimize the following loss function, which is actually an extension of the logistic loss:

$$\sum_i \sum_l (\ln(1+e^{-Y_i[l]f(x_i,l)}) + \eta \mathrm{KL}(P(Y_i[.]=1|x_i) \parallel \rho(f(x_i,.)))) \tag{10}$$

where

$$KL(p \parallel q) = p\ln\left(\frac{p}{q}\right) + (1-p)\ln\left(\frac{1-p}{1-q}\right)$$

is the KL divergence between two distributions p and q. In our case, these two distributions correspond to the class confidences from the initial model, $P(Y_i[.]=1|x_i)$ and to the distribution from the constructed model, $\rho(f(x_i,.))$, as defined by (9). This term is basically the distance from the initial model built by human-labeled data and the new model built with machine-labeled data.[2] η is used to control the relative importance of these two terms. This weight may be determined empirically on a held-out set. In addition to that, in order to reduce the noise added because of classifier errors, we can exploit only those examples that are classified with a confidence higher than some threshold. This threshold is also optimized using the held-out set.

The modification of the AdaBoost algorithm to incorporate this new loss function is explained in detail in (Schapire et al. 2005). Equation (10) can be rewritten as

$$\begin{aligned} C + \sum_i \sum_l & \ln(1 + e^{-Y_i[l] f(x_i,l)}) \\ & + \eta P(Y_i[l] = 1|x_i) \ln(1 + e^{-f(x_i,l)}) \\ & + \eta(1 - P(Y_i[l] = 1|x_i)) \ln(1 + e^{f(x_i,l)}). \end{aligned}$$

This function can be minimized by adding two additional examples for each given example with weights $\eta P(Y_i[l] = 1|x_i)$ and $\eta(1 - P(Y_i[l] = 1|x_i))$ for $Y_i[l] = 1$ and $Y_i[l] = -1$, respectively. The AdaBoost algorithm then considers these weights of individual examples instead of starting with uniform distribution.

5 Model adaptation

The aim of model adaptation is to exploit the existing labeled data and models for improving the performance of the new similar applications using a supervised adaptation method. The basic assumption is that there is an existing model trained with data similar to the target application. Then the idea is to adapt this classification model using the small amount of already-labeled data from the target application, thus reducing the amount of human labeling necessary to come up with more accurate statistical classification systems. The very same adaptation technique can be employed for improving the existing model for nonstationary data, where the data characteristics of an application change over time.

There are at least two other ways of exploiting the existing labeled data from a similar application. We will evaluate and compare these methods to adaptation in Sect. 7.

– **Simple Data Concatenation** (*simple*): where the new classification model is trained using the data from the previous application concatenated to the data labeled for the target application.
– **Tagged Data Concatenation** (*tagged*): where the new classification model is trained using both data sets, but each set is tagged with the source application. That is, in addition to the examples, we use the source of each example as an additional feature during classification.

In our approach for adaptation, we begin with an existing classification model. Then using the labeled data from the target application we build an augmented model based on this existing model. Note that this model is not trained from scratch (hence the adaptation). The learning starts from an existing model and then augments that model. In other words, we add more iterations (hence more weak learners) to the existing model (which is nothing but a set of weak learners). This method is similar to exploiting unlabeled examples as

[2]Note that, although P and ρ do not give probabilities, but rather some confidence scores between 0 and 1, this function provides an estimate of distance.

presented in the previous section, where a model that fits both the manually-labeled training data and machine-labeled data is trained. In this case, the aim is to train a model that fits both a small amount of application-specific labeled data and the existing model from the similar application. More formally, the Boosting algorithm tries to minimize the following loss function:

$$\sum_i \sum_l (\ln(1 + e^{-Y_i[l] f(x_i,l)}) + \eta \mathrm{KL}(P(Y_i[.] = 1|x_i) \parallel \rho(f(x_i, .)))) \qquad (11)$$

where $P(Y_i[l] = 1|x_i)$ is the distribution from the initial existing model and $\rho(f(x_i, .))$ is the distribution from the newly constructed model as given by (9). This term is basically the distance from the existing model to the new model built with newly labeled in-domain data. Here, again, η is used to control the relative importance of these two terms and may be determined empirically on a held-out set.

6 Cost-sensitive classification

The aim of cost-sensitive classification is to extend Boosting to allow weighted classes. Our main idea is to change the criterion to choose the best weak learner at each round of Boosting.

The bound given by (1) implies that in order to minimize the training error, a reasonable approach would be minimizing Z_t on each round of Boosting. This leads to a criterion for finding weak hypotheses, $h_t(x,l)$ for a given iteration, t (Schapire and Singer 1999). Assume that the weak learners, h, make their predictions based on a partitioning of the domain $\mathcal{X}$ into disjoint blocks X_j. For example, if the weak learner is a decision stump, checking the absence or presence of an n-gram, there are two blocks for each class. Let $c_{jl} = h(x,l)$ for $x \in X_j$. Define

$$W_b^{jl} = \sum_{i:x_i \in X_j} D(i,l)(1 - \delta(x_i,l))$$

where $b = \{-,+\}$ depending on the value of $Y[l] \in -1, 1$. In other words W_+^{jl} (W_-^{jl}) is the total weight of samples in partition j (not) labeled as l. Using this terminology:

$$Z_t = \sum_l \sum_j \sum_{i:x_i \in X_j} D_t(i,l) e^{-Y_i[l] c_{jl}} = \sum_l \sum_j W_+^{jl} e^{-c_{jl}} + W_-^{jl} e^{c_{jl}}.$$

It is then straightforward to see that the optimal c_{jl}, minimizing Z_t is

$$c_{jl} = \frac{1}{2} \ln\left(\frac{W_+^{jl}}{W_-^{jl}}\right).$$

Putting this in place results in:

$$Z_t = \sum_l \sum_j 2\sqrt{W_+^{jl} W_-^{jl}}.$$

Then it is enough to choose the weak learner that minimizes this value.

Now assume that not all labels are equally important, that is, there is an associated cost or weight, w_l, to a given label, l. In such a case we need to define a weighted Hamming loss (WHL):

$$\mathrm{WHL}(H) = \frac{1}{mk} \sum_l w_l \sum_i \delta(x_i, l). \tag{12}$$

It is easy to see that when $w_l > 1$ minimizing Z_t may not be the best criterion for a weak learner to optimize the weighted Hamming loss, because the inequality

$$\mathrm{WHL}(H) \le \prod_{t=1}^{T} Z_t$$

may not hold.

Theorem 2 *With the weights of the classes being w_l, the following bound holds on the weighted Hamming loss of H:*

$$\mathrm{WHL}(H) \le \prod_{t=1}^{T} Y_t \quad \textit{if } w_l \ge 1\ \forall l$$

where

$$Y_t = \sum_{l \in \mathcal{Y}} \hat{w}_l \sum_{i=1}^{m} D_t(i, l) e^{-\alpha_t Y_i[l] h_t(x_i, l)}$$

where $\hat{w}_l$ is a function of w_l.

Proof By unraveling the update rule, we have that

$$D_{T+1}(i+1) = \frac{e^{-Y_i[l] f(x_i, l)}}{mk \prod_t Z_t}. \tag{13}$$

Moreover, if $H(x_i, l) \ne Y_i[l]$ then $Y_i[l] f(x_i, l) \le 0$ implying that $e^{-Y_i[l] f(x_i, l)} \ge 1$. Thus,

$$\delta(x_i, l) \le e^{-Y_i[l] f(x_i, l)}. \tag{14}$$

Combining (12), (13), and (14), we get

$$\begin{aligned}
\mathrm{WHL}(H) &= \frac{1}{mk} \sum_l w_l \sum_i \delta(x_i, l) \\
&\le \frac{1}{mk} \sum_l w_l \sum_i e^{-Y_i[l] f(x_i, l)} = \sum_l w_l \sum_i \left(\prod_{t=1}^{T} Z_t \right) D_{T+1}(i, l) \\
&= \left(\prod_{t=1}^{T} Z_t \right) \sum_l w_l \sum_i D_{T+1}(i, l) \le \left(\prod_{t=1}^{T} Z_t \right) \sum_l w_l
\end{aligned}$$

since $D_{T+1}(i, l) \le 1$.

Define the constant $W = \sum_l w_l$. Then

$$\mathrm{WHL}(H) \leq \left(\prod_{t=1}^{T} W^{\frac{1}{t}} Z_t\right) = \left(\prod_{t=1}^{T} Y_t\right)$$

when $\hat{w}_l = w_l \times W^{\frac{1}{t}}$. □

Following similar steps, this leads us to a new criterion to select the weak hypotheses: Choose the weak learner that minimizes the following:

$$Y_t = \sum_l \hat{w}_l \sum_j 2\sqrt{W_+^{jl} W_-^{jl}}.$$

Note that this is in contrast to the unweighted case, where the weak learner optimizes

$$Z_t = \sum_l \sum_j 2\sqrt{W_+^{jl} W_-^{jl}}.$$

In other words, this corresponds to changing the selection criterion for the weak learner, h_t, in the AdaBoost algorithm. No change is required in the algorithm presented in Sect. 3.

7 Experiments and results

We evaluated the proposed methods using the utterances from the database of the AT&T VoiceTone® spoken dialog system (Gupta et al. 2006). This is a call routing system where the aim is to route the input calls in a customer care call center. In this natural language spoken dialog system, callers are greeted by the open-ended prompt "*How May I Help You?*" Users then ask questions about their phone bills, calling plans, and so on. The system tries to identify the customer's intent (call-type), which is basically framed as a call classification problem similar to the literature (Chu-Carroll and Carpenter 1999; Gorin et al. 1997; Natarajan et al. 2002). If the system is unable to understand the caller with high enough confidence, then the conversation will proceed with either a clarification or a confirmation prompt.

In our experiments all the utterances are transcribed. We performed our tests using the Boostexter tool (Schapire and Singer 2000). For all experiments, we used word n-grams as features and decision stumps as weak classifiers.

7.1 Evaluation metrics

While evaluating classification performance, we used mainly two metrics, both using micro-averaging allowing multiple call-types. The first one is the *top class error rate* (TCER), which is the fraction of utterances in which the call-type with maximum probability was not one of the true call-types. Inspired by the information retrieval community, the second metric we used is the *F-Measure*, which is the harmonic mean of *recall* and *precision*. Recall is defined as the proportion of all the true call-types that are correctly deduced by the classifier. It is obtained by dividing the number of true positives by the sum of true positives and false negatives. Precision is defined as the proportion of all the accepted call-types that are also true. It is obtained by dividing true positives by the sum of true positives and false positives. True (False) positives are the number of call-types for an utterance for which the deduced call-type has got a confidence above a given threshold, hence accepted, and is (not) among the correct call-types. False negatives are the number of call-types for an utterance for which the deduced call-type has got a confidence less than a threshold, hence rejected, but is among the true call-types. More formally, let

$$a = \#\{truelabel = +, predicted = +\},$$
$$b = \#\{truelabel = +, predicted = -\},$$
$$c = \#\{truelabel = -, predicted = +\},$$
$$d = \#\{truelabel = -, predicted = -\}.$$

Then

$$recall = \frac{a}{(a+b)}, \qquad precision = \frac{a}{(a+c)},$$
$$F - Measure = \frac{2 \times recall \times precision}{recall + precision}.$$

One difference between these two evaluation metrics is that the top class error rate evaluates only the top-scoring call-type for an utterance, whereas the F-Measure evaluates all the call-types exceeding the given threshold. For lower thresholds, the precision is lower but recall is higher, and vice versa for higher thresholds. To optimize the F-Measure, we check its value for all thresholds between 0 and 1, and use the best one as the F-Measure of that system, since it is always possible to change the operational threshold of the system.

7.2 Semisupervised learning experiments

To evaluate the proposed semisupervised learning method, we selected a call classification application. The data characteristics are given in Table 1. In this first set of experiments, we kept the number of iterations fixed to 500. First, we selected the optimal threshold of top-scoring call-type confidences using TCER on the held-out set. Obviously, there is a trade-off in selecting the threshold. If it is set to a lower value, that means a larger amount of noisy data, and if it is set to a higher value, that means a lesser amount of useful or informative data. Figure 2 proves this behavior for the held-out set. We trained initial models using 2,000, 4,000, and 8,000 human-labeled utterances and then augmented these as described in the simple method, with the remaining data in the training set (using only machine-labeled call-types). On the x axis, we have different thresholds to select from the unlabeled data that the classifier uses, and on the y axis we have the classification error rate (TCER) if that data is also exploited. A threshold of 0 means using all the machine-labeled data and 1 means using none. As seen, there is consistently a 1–1.5% difference in classification error

Table 1 Data characteristics used in the semisupervised learning experiments

Training Data Size	57,829 utterances
Test Data Size	3,513 utterances
Held-out Data Size	3,500 utterances
Number of Call-types	49

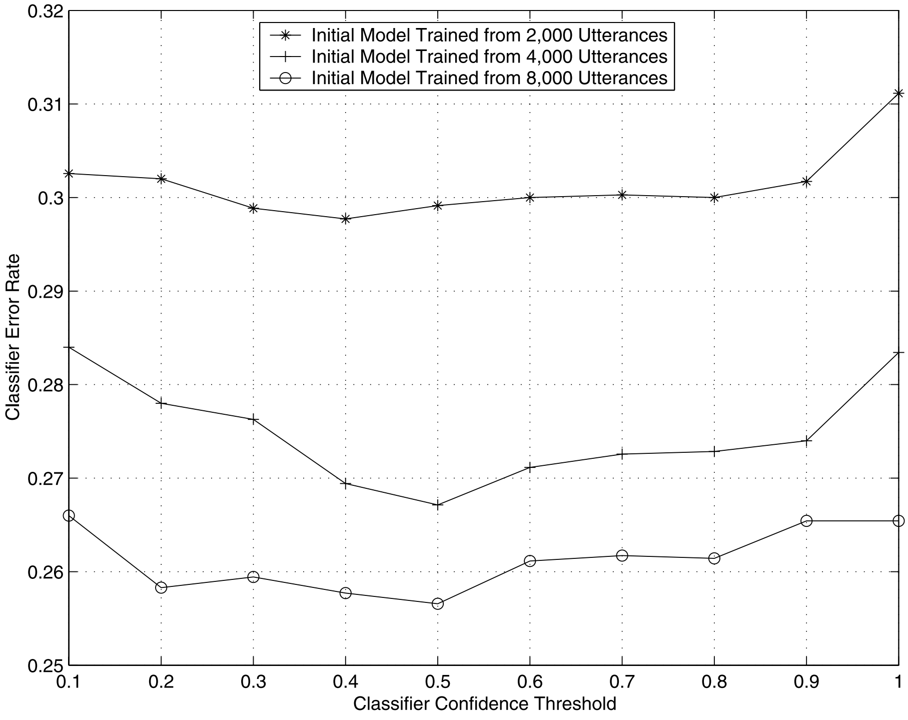

Fig. 2 Trade-off for choosing the threshold to select among the machine-labeled data on the held-out set

rates using various thresholds for each data size, and the lowest error rates are achieved with thresholds around 0.4–0.5.

Figure 3 depicts the performance using the proposed semisupervised learning method, by plotting the learning curves for various initial labeled data set sizes. In the figure, x axis is the amount of human-labeled training utterances, and y axis is the classification error rate of the corresponding model on the test set. The baseline is the top curve, with the highest error rate, where no machine-labeled data is used. To compare our proposed approach, we also employed the simple data augmentation method, where we concatenate the human-labeled and machine-labeled data. In these experiments, we selected 0.5 as the threshold for selecting machine-labeled data, and for each data size, we optimized the weight η in the proposed method using the held-out set. As in the case of the held-out set, we consistently obtained 1–1.5% classifier error rate reductions on the test set using both approaches when the labeled training data size is less than 15,000 utterances.[3] The reduction in the need for human-labeled data to achieve the same classification performance is around 30%. For example we get the same performance when we exploit machine-labeled data with 8,000 human-labeled utterances instead of 12,000 utterances. The proposed method performed consistently better (though not significantly) when there are fewer human-labeled examples. The simple semisupervised learning method outperformed the proposed method after 5,000 human-labeled utterances.

[3]For this test set, 1.3% is statistically significant according to a Z-test for 95% confidence interval.

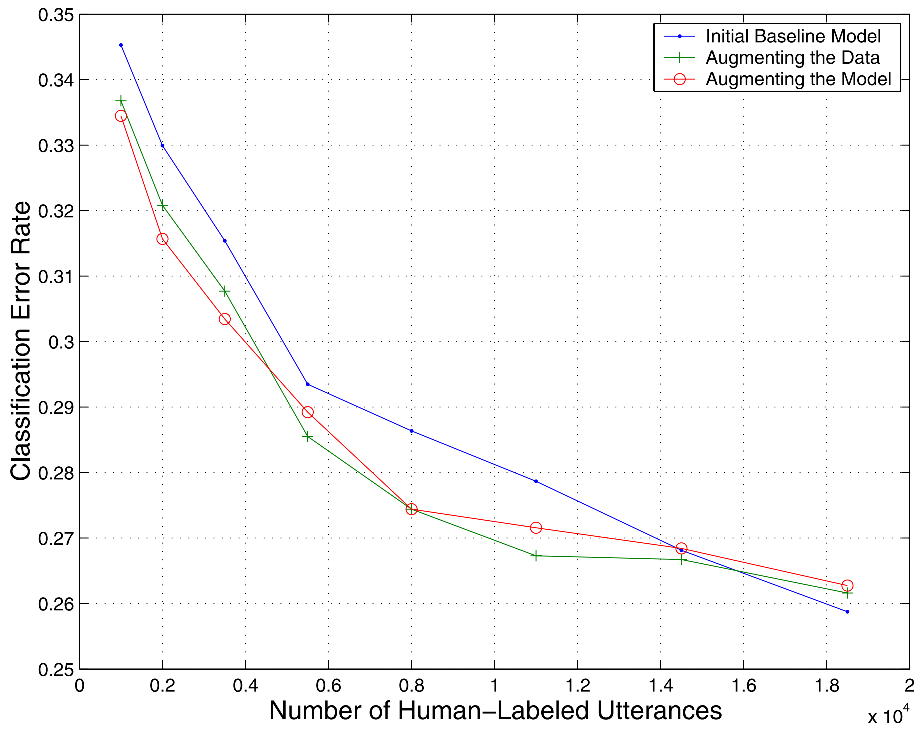

Fig. 3 Results using semisupervised learning. The top-most learning curve is obtained using just human-labeled data as a baseline. Below that lie the learning curves using the first (concatenation of human- and machine-labeled data) and second (Boosting adaptation) methods

7.3 Model adaptation experiments

To evaluate the proposed model adaptation method, we selected two applications, T_1 and T_2, both from the telecommunications domain, where users have requests about their phone bills, calling plans, and so on. The first application is concierge-like and has all the call-types the second application covers. The second application is used only for a specific subset of call-types. The data properties are shown in Table 2. We computed the perplexity of the call-type probability distribution as

$$Perplexity = 2^{-\sum_{c \in C}(p(c) \times \log\ p(c))}$$

where $p(c)$ is the prior probability of a call-type $c \in C$. As seen, the perplexity of the second application is significantly lower while the utterances are longer. As shown in Fig. 4 the class distributions for these two applications are significantly different. We have about 9 times more data for the first application. To avoid dealing with finding the optimal iteration numbers in Boosting, we iterated many times, got the error rate after each iteration and used the best error rate in all the results below.

In this experiment, the goal is adapting the classification model for T_1 using T_2 so that the resulting model for T_2 would perform better. Table 3 presents the baseline results using training and test data combinations. The rows indicate the training sets, and columns indicate the test sets. The values are the classification error rates, which are the ratios of the utterances

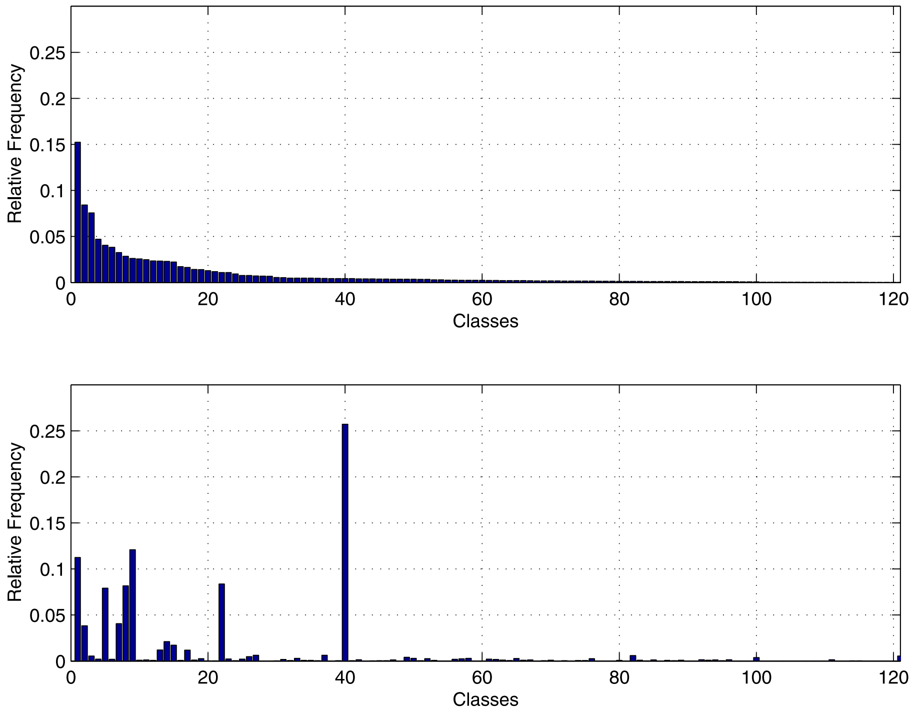

Fig. 4 Class distributions for applications T_1 (*above*) and T_2 (*below*). The classes for two applications are aligned

Table 2 Data characteristics used in the model adaptation experiments

	T_1	T_2
Training Data Size	53,022 utterances	5,866 utterances
Test Data Size	5,529 utterances	614 utterances
Number of Call-Types	121	98
Call-Type Perplexity	39.42	14.68
Average Utterance Length	8.06 words	10.57 words

for which the classifier's top scoring class is not one of the correct call-types. The third row is simply the concatenation of both training sets (indicated by *simple*). The fourth row (indicated by *tagged*) is obtained by training the classifier with an extra feature indicating the source of that utterance, either T_1 or T_2. The performance of the adaptation is shown in the last three rows (indicated by *adapt*). As seen, although the two applications are very similar, when the training set does not match the test set, the performance drops drastically. Adding T_1 training data to T_2 does not help, and actually hurts significantly.[4] This negative effect disappears when we denote the source of the training data, but no improvement has been observed on the performance of the classification model for T_2. Adaptation experiments

[4]For this test set, 3% is statistically significant according to a Z-test for 95% confidence interval.

Table 3 Adaptation results for the experiments. "*simple*" indicates simple concatenation, "*tagged*" indicates using an extra feature denoting the source of training data, "*adapt*" indicates adaptation with different η values

Training Set	Test Set	
	T_1	T_2
T_1	14.35%	26.87%
T_2	36.43%	13.36%
$T_1 + T_2$ *simple*	14.15%	16.78%
$T_1 + T_2$ *tagged*	14.05%	13.36%
$T_1 + T_2$ *adapt*$(\eta = 0.1)$	19.01%	**12.54%**
$T_1 + T_2$ *adapt*$(\eta = 0.5)$	16.13%	14.01%
$T_1 + T_2$ *adapt*$(\eta = 0.9)$	15.27%	15.96%

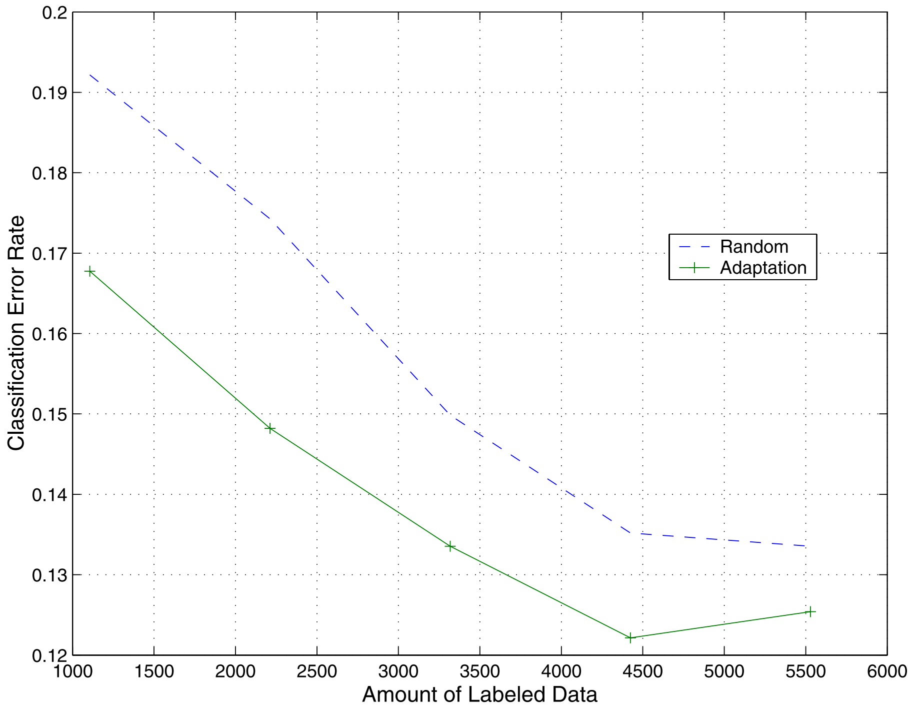

Fig. 5 Results using call-type classification model adaptation. x-axis is the amount of labeled data from application T_2. The top learning curve is obtained using just T_2 data as a baseline. Below that lies the learning curve using the adaptation

using different η values indicate interesting results. We see that using a value of 0.1, it is actually possible to outperform the model performance trained using only T_2 training data.

Since we expect the proposed adaptation method to work better with less application specific training data, we draw the learning curves as presented in Fig. 5 using 0.1 as the η value. The top curve is the baseline, obtained using random selection of only T_2 training data. When we adapt the T_1 model with only 1,106 utterances from T_2 we see 2.5% absolute

Table 4 Data characteristics used in the cost-sensitive classification experiments

Training Data Size	9,094 utterances
Test Data Size	5,171 utterances
Number of Call-Types	84
Call-Type Perplexity	32.64
Average Utterance Length	10.66 words

Table 5 Performance change when one call-type is weighted more than others

Weight	Overall			*Tellme(Balance)*		
	Recall	Precision	F-Measure	Recall	Precision	F-Measure
1	0.57	0.77	0.656	0.66	0.87	0.750
100	0.57	0.77	0.659	0.76	0.88	0.813
10000	0.56	0.73	0.631	0.75	0.82	0.784

improvement, which means 56% reduction (from about 2,500 utterances to 1,106 utterances for an error rate of 16.77%) in the amount of data needed to achieve that performance. As the number of human-labeled utterances from application T_2 increases, the difference between random and adaptation curves gets smaller, but the adaptation curve reaches the performance obtained with using all random data with 40% less data. Throughout the curves we see that adaptation performs better consistently than the baseline, though not significantly.

7.4 Cost-sensitive classification experiments

To evaluate the proposed cost-sensitive classification method, we selected a call classification application, again. Table 4 summarizes the characteristics of our application including amount of training and test data, total number of call-types, average utterance length, and call-type perplexity. Similar to semisupervised learning experiments, we kept the number of Boosting iterations fixed to 500.

As a first experiment, we chose one moderately frequent call-type, namely *Tellme(Balance)*, occurring 189 times and increased its weight, while keeping all other weights as 1. Table 5 shows the change in the performance of that call-type and that of the overall performance. We tried various weights for *Tellme(Balance)* as shown in the first column of the table. As seen, the F-Measure of that call-type has increased significantly by 6.3% absolute without a significant change in the overall (or individual class) performance(s) when the weight is set to 100.[5] Note that when we continue increasing the weight, performance begins to deteriorate because of the decrease in precision. The vast majority of the weak learners selected for the model trained with the weight of 10,000 are related to that higher-weighted call-type, making the other call-types harder to win, and even resulting in worse performance for that call-type because of overtraining.

To see whether the same behavior is true for a set of important call-types, not just one, we randomly selected 12 call-types, occurring 766 times, and gave them higher weights. Table 6 presents our results. The F-Measure for these call-types increased by 3.5% absolute without a significant loss in overall performance with a weight of 100, but note the steady decrease in overall performance with increasing weights although the performance of the important classes continues to improve.

[5]For this test set, 1.1% is statistically significant according to a Z-test for 95% confidence interval.

Table 6 Performance change when one set of call-types is weighted more than others

Weight	Overall			Important Call-types		
	Recall	Precision	F-Measure	Recall	Precision	F-Measure
1	0.57	0.77	0.656	0.49	0.73	0.575
100	0.57	0.76	0.649	0.53	0.77	0.610
10000	0.56	0.73	0.633	0.54	0.76	0.618

8 Conclusions

We have proposed three methods for extending the Boosting family of classifiers, namely, semisupervised learning, model adaptation, and cost-sensitive classification. We have shown the effectiveness of these methods in a real-life application, the AT&T VoiceTone call routing system.

Note that although these methods are classifier (namely, Boosting) dependent, the ideas are more general and can be applied to other classifiers. For example, in a Naive Bayes classifier, adaptation can be implemented as linear model interpolation or a Bayesian adaptation (like MAP) can be employed. It is also possible to apply the idea for other classification tasks that may need adaptation such as topic classification or named entity extraction.

Our future work includes unsupervised adaptation of classification models. This will enable us to bootstrap new models without labeling any application-specific data. For cost-sensitive classification we would like to extend this formulation so as to enable costs for pairs of classes instead of single classes.

Acknowledgements We thank Robert E. Schapire for providing us with the Boostexter classifier and his suggestions. We thank Dilek Hakkani-Tür, Murat Saraclar, Patrick Haffner, and Mazin Gilbert for many helpful discussions.

References

Alshawi, H. (2003). Effective utterance classification with unsupervised phonotactic models. In *Proceedings of the human language technology conference (HLT)-conference of the North American chapter of the association for computational linguistics (NAACL)*, Edmonton, Canada.

Bacchiani, M., Roark, B., & Saraclar, M. (2004). Language model adaptation with MAP estimation and the perceptron algorithm. In *Proceedings of the human language technology conference (HLT)-conference of the North American chapter of the association for computational linguistics (NAACL)*, Boston, MA.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the workshop on computational learning theory (COLT)*, Madison, WI.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Chu-Carroll, J., & Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics*, *25*(3), 361–388.

Collins, M., Schapire, R. E., & Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning, 48*(1/2/3).

Di Fabbrizio, G., Dutton, D., Gupta, N., Hollister, B., Rahim, M., Riccardi, G., Schapire, R., & Schroeter, J. (2002). AT&T help desk. In *Proceedings of the international conference on spoken language processing (ICSLP)*, Denver, CO.

Digalakis, V., Rtischev, D., & Neumeyer, L. G. (1995). Speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Transactions on Speech and Audio Processing, 3*(5).

Domingos, P. (1999). MetaCost: a general method for making classifiers cost sensitive. In: *Proceedings of the international conference on knowledge discovery and data mining (KDD)*, San Diego, CA.

Drummond, C., & Holte, R. C. (2000). Exploiting the cost in sensitivity of decision tree splitting criteria. In *Proceedings of the international conference on machine learning (ICML)*, Palo Alto, CA.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Fan, W., Stolfo, S. J., Zhang, J., & Chan, P. K. (1999). AdaCost: misclassification cost-sensitive boosting. In *Proceedings of the international conference on machine learning (ICML)*, Bled, Slovenia.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *38*(2), 337–374.

Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text categorization. In *Proceedings of the international conference on machine learning (ICML)*, Sydney, Australia.

Gorin, A. L., Riccardi, G., & Wright, J. H. (1997). How may I help You? *Speech Communication*, *23*, 113–127.

Gupta, N., Tur, G., Hakkani-Tür, D., Bangalore, S., Riccardi, G., & Rahim, M. (2006). The AT&T spoken language understanding system. *IEEE Transactions on Speech and Audio Processing*, *14*(1), 213–222.

He, Y., & Young, S. (2004). Robustness issues in a data-driven spoken language understanding system. In *Proceedings of the HLT/NAACL workshop on spoken language understanding*, Boston, MA.

Iyer, R., Gish, H., & McCarthy, D. (2002). Unsupervised training techniques for natural language call routing. In: *Proceedings of the international conference on acoustics, speech and signal processing (ICASSP)*, Orlando, FL.

Margineantu, D. (2002). Class probability estimation and cost-sensitive classification decisions. In *Proceedings of the European conference on machine learning (ICML)*, Helsinki, Finland.

Natarajan, P., Prasad, R., Suhm, B., & McCarthy, D. (2002). Speech enabled natural language call routing: BBN call director. In *Proceedings of the international conference on spoken language processing (ICSLP)*, Denver, CO.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the international conference on information and knowledge management (CIKM)*, McLean, VA.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*(2/3), 103–134.

Prodromidis, A. L., & Stolfo, S. (1998). Mining databases with different schemas: integrating incompatible classifiers. In *Proceedings of the international conference on knowledge discovery and data mining (KDD)*, New York, NY.

Riccardi, G., & Gorin, A. L. (2000). Stochastic language adaptation over time and state in a natural spoken dialog system. *IEEE Transactions on Speech and Audio Processing*, *8*(1), 3–9.

Roark, B., & Bacchiani, M. (2003). Supervised and unsupervised PCFG adaptation to novel domains. In *Proceedings of the human language technology conference (HLT)-conference of the North American chapter of the association for computational linguistics (NAACL)*, Edmonton, Canada.

Schapire, R. E. (2001). The boosting approach to machine learning: an overview. In *Proceedings of the MSRI workshop on nonlinear estimation and classification*, Berkeley, CA.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schapire, R. E., & Singer, Y. (2000). Boostexter: a boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Schapire, R. E., Rochery, M., Rahim, M., & Gupta, N. (2005). Boosting with prior knowledge for call classification. *IEEE Transactions on Speech and Audio Processing, 13*(2).

Tur, G. (2004). Cost-sensitive call classification. In *Proceedings of the international conference on spoken language processing (ICSLP)*, Jeju-Island, Korea.

Tur, G. (2005). Model adaptation for spoken language understanding. In *Proceedings of the international conference on acoustics, speech and signal processing (ICASSP)*, Philadelphia, PA.

Tur, G., & Hakkani-Tür, D. (2003). Exploiting unlabeled utterances for spoken language understanding. In *Proceedings of the European conference on speech communication and technology (EUROSPEECH)*, Geneva, Switzerland.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the international conference on knowledge discovery and data mining (KDD)*, San Francisco, CA.

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the IEEE international conference on data mining*, Melbourne, FL.

Zitouni, I., Kuo, H.-K. J., & Lee, C.-H. (2001). Natural language call routing: towards combination and boosting of classifiers. In *Proceedings of the IEEE automatic speech recognition and understanding (ASRU) workshop*, Trento, Italy.