IEEE Transactions on Speech and Audio Processing, 13(2), March, 2005.

Boosting with Prior Knowledge for Call Classification

Robert E. Schapire, Marie Rochery, Mazin Rahim and Narendra Gupta

***Abstract*—**
The use of boosting for call classification in spoken language understanding is described in this paper. An extension to the AdaBoost algorithm is presented that permits the incorporation of prior knowledge of the application as a means of compensating for the large dependence on training data. We give a convergence result for the algorithm, and we describe experiments on four datasets showing that prior knowledge can substantially improve classification performance.

I. Introduction

Building *robust* natural-language understanding for spoken dialogue applications such as those for automated customer care [8] and help desks [5] presents several technical challenges: (1) the need for *accurate* large-vocabulary recognition to accommodate for the variety of input requests, (2) parsing and understanding users' requests, and (3) supporting mixed-initiative and conversational dialogue. The spontaneous input and language variation for these sets of applications present major challenges to both speech recognition and language understanding.

Creating robust natural language systems is highly dependent on the availability of data for training the recognition and understanding models. In this paper, we consider the task of extracting the meaning of a user's request as a multi-label classification problem. We investigate the use of Freund and Schapire's AdaBoost algorithm [6] which combines many simple and moderately accurate categorization rules that are trained sequentially into a single, highly accurate model that can accurately predict a user's request. It has been shown to outperform traditional methods for text categorization [15].

Like many machine-learning methods, the AdaBoost algorithm is entirely data-driven in the sense that the classifier it generates is derived exclusively from the evidence present in the training data itself. This can be a problem since it means that spoken dialogue systems or new functionalities cannot be deployed until sufficient real data has been collected.

In this paper, we explore the use of human-crafted knowledge to compensate for the lack of data in building robust classifiers. In its standard form, boosting does not allow for the direct incorporation of such prior knowledge. We describe a new modification of boosting that combines and balances human expertise with available training data. We aim for an approach that allows the human's rough judgments to be refined, reinforced and adjusted by the statistics of the training data, but in a manner that does not permit the data to entirely overwhelm human judgments.

Prior knowledge may be acquired from several sources, e.g., human judgment, application guidelines and manuals, world knowledge, and in-domain website. In fact while developing a spoken dialogue system designers do have access to one or more such sources of knowledge. Designers use these sources of knowledge to deduce information crucial for the development of the dialogue system, i.e., the functionalities to support, and a basic understanding of how users may interact with the application. It would be only prudent, therefore, to also use these sources of knowledge for bootstrapping the text categorization module needed for the natural language understanding, especially when data is limited.

As an example, prior knowledge allows us to encode rules that can classify user responses to confirmation questions like: "So you want to fly from Boston to New York on Sunday evening?" A user response containing "yes", "okay", "correct", "all right", "fine", etc. is highly indicative of a positive confirmation.

The basic idea of our approach is to modify the loss function used by boosting so that the algorithm balances two terms, one measuring fit to the training data, and the other measuring fit to a human-built model. The actual algorithmic modification that this entails turns out to be very simple, only requiring the addition of weighted pseudo-examples to the training set. In this respect, our method turns out to be similar to one suggested by Pazzani and Billsus [12] for modifying the naive Bayes algorithm to incorporate prior knowledge.

We allow prior knowledge that may be of any form that provides guesses, however rough, of the conditional probability of class labels for each training example. We include one example of how such a model can be easily built for text categorization tasks from human-chosen keywords.

Our approach is based on the boosting-style algorithm for logistic regression described by Collins, Schapire and Singer [2], and we use their results to prove a simple convergence theorem for our algorithm. Although presented only in the boosting setting, our general technique for incorporating prior knowledge can be combined with any method based on logistic regression.

We present several experiments using boosting on both speech and text corpora. We describe experiments on datasets derived from two spoken-dialogue applications. We also present results on two text-based benchmark datasets. In each

R. Schapire conducted this research while with AT&T Labs, and is now with Princeton University, Department of Computer Science, 35 Olden Street, Princeton, NJ 08544 USA (email: schapire@cs.princeton.edu).

M. Rochery conducted this research while visiting AT&T Labs, and is now with Ariana, INRIA, 2004 Route des Lucioles BP 93, 06902 Sophia Antipolis cedex, France (email: marie.rochery@sophia.inria.fr).

M. Rahim and N. Gupta are with AT&T Labs – Research, Shannon Laboratory, 180 Park Avenue, Florham Park, NJ 07932 USA (email: {mazin, ngupta}@research.att.com).

case, we compare boosting with and without prior knowledge. The results show that prior knowledge can substantially improve performance, particularly when data is greatly limited.

II. Boosting and Logistic Regression

We assume that we are given a set of training examples $(x_1, y_1), \ldots, (x_m, y_m)$. Each x_i is called an *instance*. In this paper, each x_i will generally be the text of a transcribed or recognized utterance; however, in general, x_i may incorporate other information about what was spoken, or more generally, about whatever the object is that is to be classified. Each y_i is the *label* or *class* assigned to the instance x_i; for instance, y_i may indicate call type. For simplicity, we assume for now that there are only two classes, -1 and $+1$. Let $\mathcal{X}$ and $\mathcal{Y}$ be the spaces of all possible instances and all possible labels, respectively. Thus, for now, $\mathcal{Y} = \{-1, +1\}$. When discussing probabilities, we assume that all training and test examples (x, y) are selected independently from some distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$.

The goal of a learning algorithm is to use the training data to derive a rule that accurately predicts the class of any new instance x; such a prediction rule is called a *classifier*. The approach that we take is based on a machine-learning method called *boosting* [6], [13]. The basic idea of boosting is to build a highly accurate classifier by combining many "weak" or "simple" *base classifiers*, each one of which may only be moderately accurate. To obtain these base classifiers, we assume we have access to a *base learning algorithm* that we use as a black-box subroutine.

The collection of base classifiers is constructed in rounds. On each round t, the base learner is used to generate a base classifier h_t. Besides supplying the base learner with training data, the boosting algorithm also provides a set of nonnegative weights W_t over the training examples. Intuitively, the weights encode how important it is that h_t correctly classify each training example. Generally, the examples that were most often misclassified by the preceding base classifiers will be given the most weight so as to force the base learner to focus on the "hardest" examples.

Following Schapire and Singer [14], we use *confidence-rated* classifiers h that, rather than outputting simply -1 or $+1$, output a real number $h(x)$ whose sign (-1 or $+1$) is interpreted as a prediction, and whose magnitude $|h(x)|$ is a measure of "confidence." We refer to these as *base functions*.

Although our eventual goal is classification, we focus on estimating probabilities which can be converted into classifications in the obvious way by thresholding. Specifically, given training data, we wish to build a rule that estimates the conditional probability that $y = +1$ given x when test example (x, y) is chosen according to $\mathcal{D}$. In logistic regression, we do this by building a real-valued function $f : \mathcal{X} \to \mathbb{R}$ and estimating this probability by $\sigma(f(x))$ where

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Later, f will be of a particular form, namely, a linear combination of base functions. Once such a model has been postulated, we can attempt to find f by maximizing the conditional likelihood of the data, or equivalently, minimizing the negative log conditional likelihood which works out to be

$$\sum_i \ln\left(1 + \exp(-y_i f(x_i))\right). \tag{3}$$

```
Input: $(x_1, y_1), \ldots, (x_m, y_m)$
    where $x_i \in X$, $y_i \in \{-1, +1\}$
for $t = 1, \ldots T$:
  • let
        $W_t(i) = \frac{1}{1 + \exp\left(y_i \sum_{t'=1}^{t-1} h_{t'}(x_i)\right)}$    (1)
  • use the $W_t(i)$'s to obtain base function $h_t : X \to \mathbb{R}$ from
    base learner; the base learner should minimize the objective function:
        $\sum_i W_t(i) e^{-y_i h_t(x_i)}$    (2)
Output final classifier: $f(x) = \sum_{t=1}^{T} h_t(x)$
```

Fig. 1. A binary boosting algorithm.

A connection between boosting and logistic regression was first suggested by Friedman, Hastie and Tibshirani [7], and was further explored by Duffy and Helmbold [4]. Along these same lines, Collins, Schapire and Singer [2] describe a variant of Freund and Schapire's [6] AdaBoost algorithm for minimizing Eq. (3) over functions f that are linear combinations of base functions. Pseudo-code for the algorithm that we use, which we call AdaBoost.L, is shown in Fig. 1. This algorithm is the same as that of Collins, Schapire and Singer except in the manner in which base functions h_t are chosen on each round; our algorithm is essentially a confidence-rated version of theirs.

Like AdaBoost, AdaBoost.L works in rounds. On each round, a set of weights $W_t(i)$ over the training set is computed as in Eq. (1) and used to find a base function $h_t : \mathcal{X} \to \mathbb{R}$. This base function should minimize Eq. (2) over some space of base functions. After T rounds, the sum of all the h_t's is output as the final function f. This procedure is in fact identical to confidence-rated AdaBoost if we instead compute $W_t(i)$ using the rule

$$W_t(i) = \exp\left(-y_i \sum_{t'=1}^{t-1} h_{t'}(x_i)\right).$$

A. Convergence

We can use the results and techniques of Collins, Schapire and Singer [2] to prove the convergence of this algorithm to the minimum of Eq. (3), provided the base functions have a particular form so that the space $\mathcal{H}$ of base functions is *semi-finite*, meaning that $\mathcal{H}$ contains a finite set of functions $\mathcal{G}$ for which:

1) every function in $\mathcal{H}$ can be written as a linear combination of the functions in $\mathcal{G}$, and
2) αg is in $\mathcal{H}$ for every $\alpha \in \mathbb{R}$ and $g \in \mathcal{G}$.

Theorem 1: Assume the base functions h_t in Fig. 1 minimize Eq. (2) over a semi-finite space $\mathcal{H}$. Then as $T \to \infty$, the loss in Eq. (3) for the final function f converges to the infemum of this loss over all linear combinations of functions in $\mathcal{H}$.

Proof sketch: Collins, Schapire and Singer [2] proved the convergence of their sequential-update algorithm when run with a finite set of base functions. Since our algorithm AdaBoost.L is a confidence-rated variant of theirs, to prove the result, we only need to show that, on each round, AdaBoost.L makes at least as much progress as their sequential-update algorithm applied to the finite set $\mathcal{G}$. In particular, we note that

$$\begin{aligned}\sum_i W_t(i) e^{-y_i h_t(x_i)} &= \min_{h \in \mathcal{H}} \sum_i W_t(i) e^{-y_i h(x_i)} \\ &\le \min_{\alpha \in \mathbb{R}, g \in \mathcal{G}} \sum_i W_t(i) e^{-y_i \alpha g(x_i)} \\ &\le \sum_i W_t(i) e^{-y_i \alpha_t g_t(x_i)}\end{aligned}$$

where α_t and g_t are the choices that would have been made by their algorithm. With these additional steps, their proof of convergence is easily modified. $\blacksquare$

The base learning algorithm that we use in our experiments for finding base functions is the same as in Schapire and Singer's [15] BoosTexter system. These experiments all deal with text, and each base function tests for the presence or absence of a particular word, short phrase or other simple pattern, henceforth referred to simply as a *term*. If the term is present, then one value is output; otherwise, some other value is output. For instance, the base function might be: "If the word 'yes' occurs in the text, then output $+1.731$, else output -2.171." Schapire and Singer [15] describe a base learning algorithm that efficiently finds the best base function of this form, i.e., the one minimizing Eq. (2). It can be seen that this space of base functions is semi-finite since there are only finitely many terms and since a rule of this form can be decomposed as $a_0 g_0 + a_1 g_1$ where $a_0, a_1 \in \mathbb{R}$ and g_1 (respectively, g_0) outputs 1 if the term is present (respectively, absent), and 0 otherwise.

III. Incorporating Prior Knowledge

We now describe our modification to boosting to incorporate prior knowledge. In our approach, a human expert must begin by constructing a rule π mapping each instance x to an estimated conditional probability distribution $\pi(y|x)$ over the possible label values $y \in \{-1, +1\}$. We discuss below some methods for constructing such a rule.

Given this background or prior model and training data, we now have two possibly conflicting goals in constructing a predictor: (1) fit the data, and (2) fit the prior model. As before, we measure fit to the data using log conditional likelihood as in Eq. (3). To measure fit to the prior model, for each example x_i, we use relative entropy (also called Kullback-Leibler divergence) between the prior model distribution $\pi(\cdot|x_i)$ and the distribution over labels associated with our constructed logistic model $\sigma(f(x_i))$. More precisely, letting $\pi_+(x) = \pi(y = +1|x)$, we measure fit to the prior model by

$$\sum_i \mathrm{RE}\left(\pi_+(x_i) \ \| \ \sigma(f(x_i))\right) \tag{4}$$

where

$$\mathrm{RE}\left(p \ \| \ q\right) = p \ln(p/q) + (1-p)\ln((1-p)/(1-q))$$

is binary relative entropy. The relative importance of the two terms is controlled by the parameter η.

Putting these together, we get the objective function

$$\begin{aligned}\sum_i & [\ln\left(1 + \exp(-y_i f(x_i))\right) \\ & \quad + \eta \mathrm{RE}\left(\pi_+(x_i) \ \| \ \sigma(f(x_i))\right)].\end{aligned} \tag{5}$$

This can be rewritten as

$$\begin{aligned}C \ + \ \sum_i & [\ln(1 + e^{-y_i f(x_i)}) \\ & + \eta \pi_+(x_i) \ln(1 + e^{-f(x_i)}) \\ & + \eta(1 - \pi_+(x_i)) \ln(1 + e^{f(x_i)})]\end{aligned} \tag{6}$$

where C is a term that is independent of f, and so can be disregarded. Note that this objective function has the same form as Eq. (3) over a larger set and with the addition of nonnegative weights on each term.

Thus, to minimize Eq. (6), we apply the AdaBoost.L procedure described in Section II to a larger weighted training set. This new set includes all of the original training examples (x_i, y_i), each with unit weight. In addition, for each training example (x_i, y_i), we create two new training examples $(x_i, +1)$ and $(x_i, -1)$ with weights $\eta\pi_+(x_i)$ and $\eta(1 - \pi_+(x_i))$, respectively. Thus, we triple the number of examples.[1] During training, these weights w_0 are now used in computing W_t so that

$$W_t(i) = \frac{w_0(i)}{1 + \exp\left(y_i \sum_{t'=0}^{t-1} h_{t'}(x_i)\right)}$$

(here, i ranges over all of the examples in the *new* training set). The modification of Theorem 1 for weighted training sets is straightforward.

One final modification that we make is to add a 0-th base function h_0 that is based on π_+ so as to incorporate π_+ right from the start. In particular, we take

$$h_0(x) = \sigma^{-1}(\pi_+(x)) = \ln\left(\frac{\pi_+(x)}{1 - \pi_+(x)}\right)$$

and include h_0 in computing the final classifier f.

A. *Multiclass problems*

Up until now, we have assumed a binary prediction problem with $\mathcal{Y} = \{-1, +1\}$. More generally, we follow Schapire and Singer's [14], [15] approach to multiclass problems in which more than two classes are allowed and furthermore in which each example may belong to multiple classes. The intuitive idea is to reduce to binary questions which ask if each example is or is not in each of the classes.

In particular, suppose that there are k classes $\mathcal{Y} = \{1, 2, \ldots, k\}$. Each label $\mathbf{y}_i$ is now a vector in $\{-1, +1\}^k$

[1] Although, by noticing that (x_i, y_i) occurs twice, we can actually get away with only doubling the training set.

where the ℓ-th component indicates if the example is or is not in class ℓ. Our purpose now is to find a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and $\sigma(f(x,\ell))$ is then the estimated probability that example x belongs to class ℓ. Treating each class separately, the objective function in Eq. (3) becomes

$$\sum_i \sum_\ell \ln\left(1 + e^{-y_{i\ell} f(x_i,\ell)}\right).$$

The boosting algorithm AdaBoost.L is modified straightforwardly: Maintaining weights on example-label pairs, Eq. (1) becomes

$$W_t(i,\ell) = \frac{1}{1 + \exp\left(y_{i\ell} \sum_{t'=1}^{t-1} h_{t'}(x_i,\ell)\right)},$$

and Eq. (2) becomes

$$\sum_i \sum_\ell W_t(i,\ell) e^{-y_{i\ell} h_t(x_i,\ell)}.$$

As was done by Schapire and Singer [15], our base learner finds rules that still test for the presence or absence of a term, but now outputs a whole vector of numbers (one for each class) depending on the result of this test.

Our prior knowledge now gives guessed estimates $\pi(\ell|x)$ of the conditional probability that example x belongs to class ℓ. We do not require that $\pi(\cdot|x)$ be a probability distribution. The objective function in Eqs. (5) and (6) becomes

$$\begin{aligned} & \sum_i \sum_\ell [\ln(1 + e^{-y_{i\ell} f(x_i,\ell)}) \\ & \quad + \eta \mathrm{RE}\left(\pi(\ell|x_i) \;\|\; \sigma(f(x_i,\ell))\right)] \\ = & \sum_i \sum_\ell [\ln(1 + e^{-y_{i\ell} f(x_i,\ell)}) \\ & \quad + \eta\pi(\ell|x_i) \ln(1 + e^{-f(x_i,\ell)}) \\ & \quad + \eta(1 - \pi(\ell|x_i)) \ln(1 + e^{f(x_i,\ell)})] + C. \end{aligned}$$

So to handle this objective function, similar to the binary case, we create a new training set with weights over example-label pairs: The original examples $(x_i, \mathbf{y}_i)$ occur with unit weight $w_0(i,\ell) = 1$. Each such example is replicated twice as $(x_i, +\mathbf{1})$ and $(x_i, -\mathbf{1})$ where $\mathbf{1}$ is the all ones vector. Letting $i+m$ and $i+2m$ be the indices of the new replicated examples, their weights are, respectively,

$$\begin{aligned} w_0(i+m,\ell) &= \eta\pi(\ell|x_i) \\ \text{and } w_0(i+2m,\ell) &= \eta(1 - \pi(\ell|x_i)). \end{aligned}$$

IV. Experiments

In this section, we describe experiments comparing boosting with prior knowledge against boosting with no such knowledge, particularly when data is substantially limited. We did not compare to other text categorization methods, since this was not the purpose of the study; moreover, Schapire and Singer [15] carried out extensive experiments comparing boosting to several other methods on text categorization problems.

Class	*Keywords*
japan	japan, tokyo, yen
bush	bush, george, president, election
israel	israel, jerusalem, peres, sharon, palestinian, israeli, arafat
britx	britain, british, england, english, london, thatcher
gulf	gulf, iraq, saudi, arab, iraqi, saddam, hussein, kuwait
german	german, germany, bonn, berlin, mark
weather	weather, rain, snow, cold, ice, sun, sunny, cloudy
dollargold	dollar, gold, price
hostages	hostages, ransom, holding, hostage
budget	budget, deficit, taxes
arts	art, painting, artist, music, entertainment, museum, theater
dukakis	dukakis, boston, taxes, governor
yugoslavia	yugoslavia
quayle	quayle, dan
ireland	ireland, ira, dublin
burma	burma
bonds	bond, bonds, yield, interest
nielsens	nielsens, rating, t v, tv
boxoffice	box office, movie
tickertalk	stock, bond, bonds, stocks, price, earnings

TABLE I

THE KEYWORDS USED FOR EACH CLASS ON THE *AP-Titles* DATASET.

We used two publicly available text categorization datasets and two proprietary speech categorization datasets. The latter datasets come from the application that was the original motivation for this work as described in Section I. We chose the former datasets because they are large, and also because they naturally lent themselves to the easy construction of a human-crafted model. We could not use a substantially larger number of datasets because of the inherently intensive, subjective and non-automatic nature of building such models.

A. Benchmark datasets

In the first set of experiments, we used these two benchmark text-categorization datasets:

- *AP-Titles*: This is a corpus of Associated Press newswire headlines [11], [10]. The object is to classify the headlines by topic. We used the preparation of this dataset described by Schapire and Singer [15] consisting of 29,841 examples and 20 classes.
- *Newsgroups*: This dataset consists of Usenet articles collected by Lang [9] from 20 different newsgroups. The object is to predict which newsgroup a particular article was posted to. One thousand articles were collected for each newsgroup. However, after removing duplicates, the total number of articles dropped to 19,466.

Base learner: We used the same base learner as described by Schapire and Singer [15]. As described in Section II-A, this base learner searches for base functions that test for the presence or absence of a term. In our experiments, we used all possible terms which were sequences of up to three words, possibly with the middle word matching a "wildcard." In the multiclass case, if the term is present, a particular set of values is output

Class	*Keywords*
alt.atheism	god, atheism, christ, jesus, religion, atheist
comp.graphics	graphics, color, computer, computers, plot, screen
comp.os.ms-windows.misc	computer, computers, operating system, microsoft, windows, ms, dos
comp.sys.ibm.pc.hardware	computer, computers, ibm, pc, clone, hardware, cpu, disk
comp.sys.mac.hardware	computer, computers, mac, macintosh, hardware, cpu, disk
comp.windows.x	computer, computers, windows, x, unix
misc.forsale	for sale, asking, selling, price
rec.autos	car, drive, fast, jaguar, toyota, ford, honda, volkswagen, gm, chevrolet, tire, engine
rec.motorcycles	motorcycle, honda, harley, wheel, engine, throttle
rec.sport.baseball	baseball, hit, strike, ball, base, bases, homerun, runs, out, outs
rec.sport.hockey	hockey, stick, puck, goal, check
sci.crypt	cryptography, encrypt, cipher, decrypt, security, secret, key
sci.electronics	electronics, computer, computers, chip, electric
sci.med	medicine, doctor, science, heal, sick, cancer
sci.space	space, astronaut, nasa, rocket, space shuttle
soc.religion.christian	religion, christian, jesus, christ, god, catholic, protestant
talk.politics.guns	guns, gun, nra, brady, kill, shoot, shot
talk.politics.mideast	mideast, israel, jordan, arafat, palestinian, syria, lebanon, saudi, iraq, iran
talk.politics.misc	politics, clinton, president, congress, senate, congressman, senator
talk.religion.misc	religion, jewish, christian, catholic, protestant, god, believe

TABLE II
THE KEYWORDS USED FOR EACH CLASS ON THE *Newsgroups* DATASET.

by the base function, one for each class; if the term is absent, some other set of values is output. It can be shown, using hash tables and manipulation of the loss function, that an efficient search can be employed on each round to quickly find the best base function of this form minimizing Eq. (2).

Prior models: Our framework permits prior knowledge of any kind, so long as it provides estimates, however rough, of the probability of any example belonging to any class. Here we describe one possible technique for creating such a rough model.

For each dataset, one of the authors, with access to the list of categories but not to the data itself, thought up a handful of keywords for each class. These lists of keywords are shown in Tables I and II. These keywords were produced through an entirely subjective process of free association with general knowledge of what the categories were about (and also the time period during which the data was connected), but no other information or access to the data. Although this step required direct human involvement, the rest of the process of generating a prior model was fully automatic.

We next used these keywords to build a very simple and naive model. We purposely used a model that is very far from perfect to see how the algorithm performs with prior knowledge that is as rough as can be expected in practice. We began be defining the conditional probability of a class ℓ given the presence or absence of a keyword w, denoted $\pi(\ell|w)$ or $\pi(\ell|\overline{w})$. We let

$$\pi(\ell|w) = \begin{cases} 0.9/n_w & \text{if } w \text{ is a keyword for } \ell \\ 0.1/(k - n_w) & \text{otherwise} \end{cases}$$

where n_w is the number of classes listing w as a keyword. In other words, if the keyword w is listed for a single class ℓ then seeing the word gives a 90% probability that the correct class is ℓ; if w is listed for several classes, the 90% probability is divided equally among them. The remaining 10% probability is divided equally among all classes not listing w as a keyword.

If w is not present, we assign equal probability to all classes: $\pi(\ell|\overline{w}) = 1/k$. We also define the prior distribution of classes to be uniform: $\pi(\ell) = 1/k$.

Given these rules, we make the naive assumption that the keywords are conditionally independent of one another given the class. We can then use Bayes' rule to compute the probability (under π) of each class given the presence or absence of all the keywords. This becomes our estimate $\pi(\ell|x)$.

Experimental set-up: For each dataset and on each run, we first randomly permuted the data. We then trained boosting, with or without prior knowledge, on the first m examples, for $m = 25, 50, 100, 200, \ldots, M$, where M is 12800 for *AP-Titles* and 6400 for *Newsgroups*. The remaining examples (i.e., starting with example $M + 1$) are used for testing. We ran each experiment ten times and averaged the results. We fixed the number of rounds of boosting to 1000. We set the parameter η using the heuristic formula $2000m^{-1.66}$ (which was chosen to interpolate smoothly between guesses at appropriate values of η for a couple values of m). This setting conforms to the basic intuition that, when more data is available, less weight should be given to the prior knowledge. No experiments were conducted to determine if better performance could be achieved with a wiser choice of η.

Results: Figs. 2 and 3 show the results of these experiments. The figures show test error rate for boosting with and

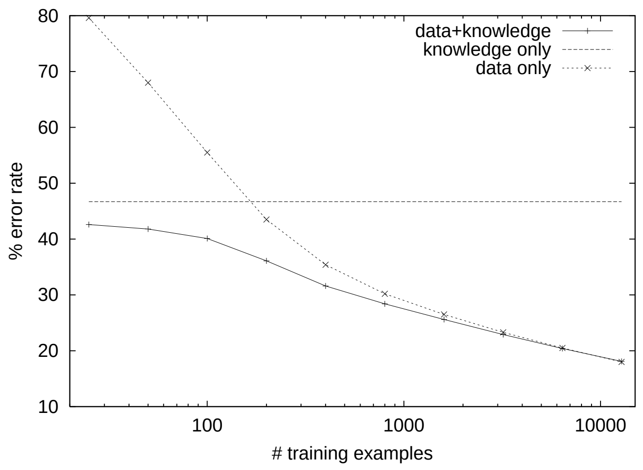

Fig. 2. Comparison of test error rate using prior knowledge and data separately or together on the *AP-Titles* dataset, measured as a function of the number of training examples.

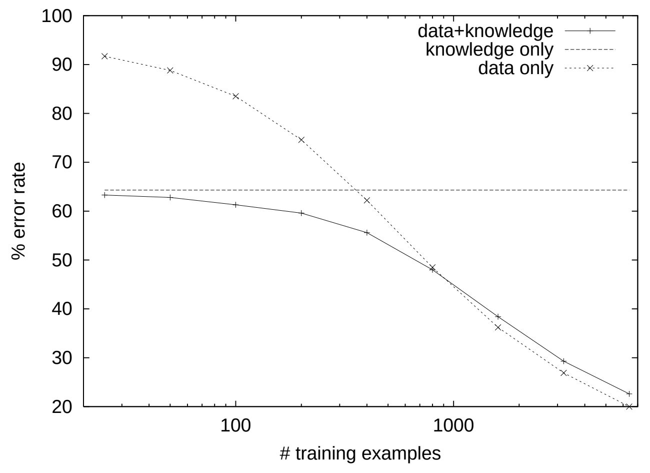

Fig. 3. Comparison of test error rate using prior knowledge and data separately or together on the *Newsgroups* dataset, measured as a function of the number of training examples.

without prior knowledge measured as a function of the number of training examples. The figures also show the error rate achieved using the prior model alone with no training examples at all. Here, error rate means fraction of test examples on which the top-scoring label was not one of the correct labels (recall that each example may belong to more than one class). The results are averaged over the ten runs.

For fairly small datasets, using prior knowledge gives dramatic improvements over straight boosting. On large training sets on the *Newsgroups* dataset, the imperfect nature of the prior knowledge eventually hurts performance, although this effect is not seen on *AP-Titles*.

B. Spoken-dialogue datasets

We next describe experiments on datasets from two AT&T spoken-dialogue applications. In both applications, the goal is to extract the meaning of utterances spoken by telephone callers. These utterances are then passed through an automatic speech recognizer. Our goal is to train a classifier that can categorize the resulting (very noisy) text. The classifier's output would then be passed to the dialogue manager which carries on with the dialogue by formulating an appropriate response to the caller's utterance.

The two applications are:

- *How May I Help You?* (*HMIHY*): Here, the goal is to identify a particular call type, such as collect call, a request for billing information, etc. There are 15 different classes. We did experiments with 50 to 1600 sentences in the training set and 2991 sentences in the test set. More information about this dataset is provided by Gorin, Riccardi and Wright [8].
- *HelpDesk*: This application provides information about AT&T's Natural Voices text-to-speech engine. For instance, the caller can ask for a demo, price information, or a sales representative. There are 22 different classes. We trained models on 100 to 2675 sentences and tested on 2000 sentences.

Prior models: The prior models were built in a similar fashion to those used in the preceding experiments, although we allowed the human more freedom in choosing probabilities, and more rules were used. For example, to encode a rule indicating that a user response containing "yes", "okay", "correct", "all right", or "fine" is highly indicative of a positive confirmation, we can define

$$\pi(\text{Yes}|\text{yes} \vee \text{okay} \vee \text{correct} \vee \text{all right} \vee \text{fine}) = 0.9,$$

that is, the probability of the "Yes" class, given that any of these keywords were uttered, is 0.9. Another example is for classifying user requests to be connected to an operator/service agent. This can be expressed by the rule

$$\pi(\text{Agent}|\text{speak}\wedge(\text{human}\vee\text{operator}\vee(\text{service}\wedge\text{agent}))) = 0.95.$$

Any probability "left over" by such a rule is spread over the remaining classes in proportion to their prior probabilities which we estimated using the available training data. Likewise, these prior or default probabilities were used when the given rule is not satisfied. As before, the probabilities assigned by the various rules are combined using Bayes rule with a naive independence assumption.

Experimental set-up: We performed similar experiments to those described in Section IV-A measuring classification accuracy as a function of the number of examples used during training, and comparing models built only with some training examples and models built with both hand-crafted rules (prior knowledge) and training examples.

On the *HMIHY* dataset, we trained the models on 50, 100, 200, 300 rounds when the number of available training examples was respectively 50, 100, 200, 400 and up. The parameter η was selected empirically based on the number of available training examples. We set η to 1 when the number of training examples was less than or equal to 200, 0.1 when it was between 400 and 800, and 0.01 when it was greater. The dashed line in Fig. 4 shows the classification accuracy for models built on hand-crafted rules and training examples whereas the solid lines show the classification accuracy for models built either on training examples only or on hand-crafted rules only. An improvement in accuracy is observed when using hand-crafted rules and training examples together. This comes from the fact that some patterns of the hand-crafted rules are not in the data at all or are not in a sufficient number of sentences to have a statistical impact when training only on the data.

In this experiment, when fewer training examples were available (< 100 examples) exploiting human expertise provided classification accuracy levels that are equivalent to models trained on four times the amount of training data. When the number of training examples is larger (> 100), accuracy levels become equivalent to two times the amount of training data. When larger than 6000 sentences were available, both models were found to converge to similar classification accuracy.

Results: Fig. 5 shows a similar comparison for the *HelpDesk* task. We trained the models on 100, 200, 400, 600, 800, 1000 rounds when the number of training examples was respectively 100, 200, 400, 800, 1600, and up. We set η to 0.1 when the number of training examples was less than or equal

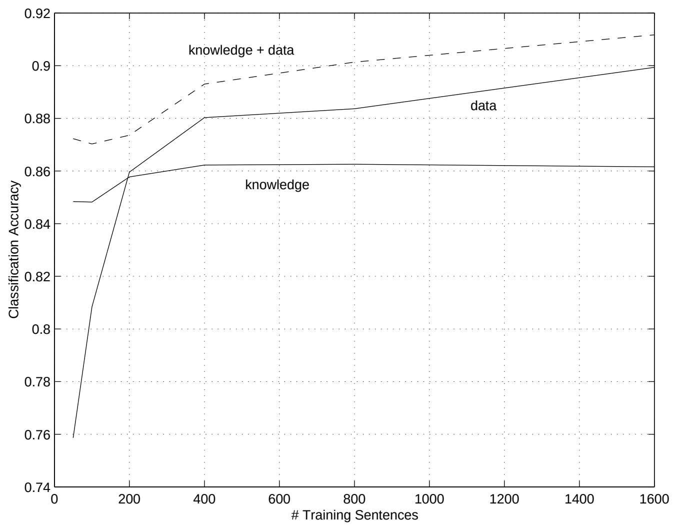

Fig. 4. Comparison of performance using data and knowledge separately or together on the *HMIHY* task.

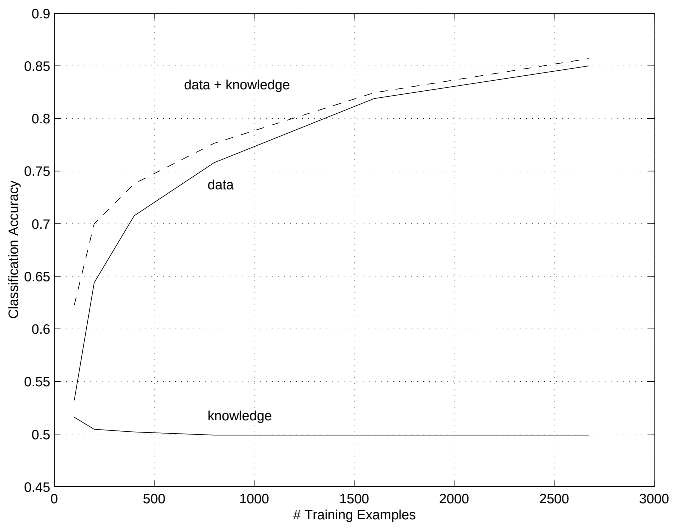

Fig. 5. Comparison of performance using data and knowledge separately or together on the *HelpDesk* task.

to 1600 and to 0.01 otherwise. Fig. 5 shows an improvement in classification accuracy when hand-crafted rules are being used. This improvement is up to 9% absolute with 100 training examples and drops to 0.5% when more data becomes available.

In the figures, the knowledge-only curves are not perfectly flat. This comes from the fact that the models from the knowledge take into account the empirical distribution of classes of the available training examples rather than using a uniform distribution as was done in Section IV-A.

A further experiment was performed to evaluate the accuracy of our classifier when new semantic classes are added following system training. This is the situation when new functionalities are needed following system deployment but with no data available. Fig. 6 shows the classification accuracy when four additional semantic classes are added to the *HMIHY* model after being trained on 11 classes. Although the system performance drops in general, the results demonstrate that incorporating human judgment helps to provide an initial boost in performance even when no data is present.

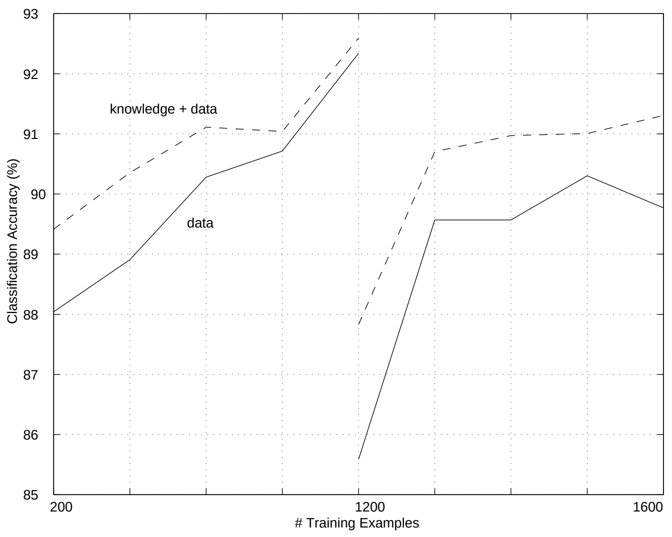

Fig. 6. Adding new semantic classes following model training.

V. Variations and extensions

We have described the extension of one particular boosting algorithm to incorporate prior knowledge. However, the same basic technique can be applied to a great variety of boosting algorithms. For instance, we have used Schapire and Singer's [14] confidence-rated boosting framework in which the base functions map to real numbers whose magnitude indicate a level of confidence. This choice is orthogonal to our basic method for incorporating prior knowledge. Although this approach can substantially speed up convergence when using a rather weak base learner, in some settings, one may wish to use a more standard base learner like C4.5 outputting "hard" predictions in $\{-1, +1\}$ and for which the goal is simply (weighted) error minimization; for this, a more basic version of AdaBoost can be used.

We also have chosen a particular method of extending binary AdaBoost to the multiclass case, an extension that Schapire and Singer [14] call AdaBoost.MH. We could instead use one of the other multiclass extensions such as AdaBoost.MR [6], [14] as modified for logistic regression by Collins, Schapire and Singer [2].

In fact, our approach is not even limited to boosting algorithms. The basic idea of modifying the loss function used in logistic regression by adding pseudo-examples can be applied to any algorithm for logistic regression.

Note that our measure of fit to the prior model given in Eq. (4) is independent of the actual training labels y_i. This means that we need not limit this term to labeled data: if, as is often the case, we have access to abundant unlabeled data, we can use it instead for this term.

Another idea for future research is to follow the co-training approach studied by Blum and Mitchell [1] and Collins and Singer [3] in which we train two models, say f and g, which we force to give similar predictions on a large set of unlabeled data. In this case, the term in Eq. (4) might be replaced by something like

$$\sum_i \mathrm{RE}\left(\sigma(g(x_i)) \;\|\; \sigma(f(x_i))\right)$$

where the sum is over the unlabeled dataset.

VI. Conclusion

We have described a new and simple method for incorporating prior knowledge into boosting as a means of compensating for insufficient data. Our approach exploits a statistical view of boosting first put forth by Friedman, Hastie and Tibshirani [7] which led to a number of boosting-like algorithms for logistic regression, including that of Collins, Schapire and Singer [2]. We used this probabilistic interpretation as a basis for modifying the underlying loss function to incorporate the prior knowledge. As shown in this paper, the resulting algorithm requires only the addition of weighted pseudo-examples, and the convergence of the algorithm follows from the work of Collins, Schapire and Singer [2].

Our experiments on text-categorization datasets indicate that performance using both data and prior knowledge can be much better than with either alone, especially when very little data is available. This improvement in performance is vital in applications like the spoken-dialogue system described in Section I which need to be deployed before enough data can be collected, but which can then take advantage of the data that later becomes available following initial deployment.

Future research is needed to determine the effectiveness of this general technique for other kinds of learning problems, when using different base learners, and when using other methods for logistic regression.

References

[1] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[2] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

[3] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[4] Nigel Duffy and David Helmbold. Potential boosters? In *Advances in Neural Information Processing Systems 11*, 1999.

[5] Giuseppe Di Fabbrizio, Dawn Dutton, Narendra Gupta, Barbara Hollister, Mazin Rahim, Giuseppe Riccardi, Robert Schapire, and Juergen Schroeter. AT&T help desk. In *7th International Conference on Spoken Language Processing*, 2002.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

[8] A. L. Gorin, G. Riccardi, and J. H. Wright. How may I help you? *Speech Communication*, 23(1-2):113–127, October 1997.

[9] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[10] David Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning: Proceedings of the Eleventh International Conference*, 1994.

[11] David Lewis and William Gale. Training text classifiers by uncertainty sampling. In *Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[12] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[13] Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[14] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[15] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, May/June 2000.