International Conference on Accoustics, Speech and Signal Processing, 2002.

COMBINING PRIOR KNOWLEDGE AND BOOSTING FOR CALL CLASSIFICATION IN SPOKEN LANGUAGE DIALOGUE

M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawi, S. Douglas

AT&T Labs - Research, 180 Park Avenue, Florham Park, NJ 07932

ABSTRACT

Data collection and annotation are major bottlenecks in rapid development of accurate syntactic and semantic models for natural-language dialogue systems. In this paper we show how human knowledge can be used when designing a language understanding system in a manner that would alleviate the dependence on large sets of data. In particular, we extend BoosTexter, a member of the boosting family of algorithms, to combine and balance hand-crafted rules with the statistics of available data. Experiments on two voice-enabled applications for customer care and help desk are presented.

1. INTRODUCTION

Building spoken natural-language dialogue systems for automated customer care and help desk applications presents several technical challenges: (1) the need for 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 for these sets of applications is significantly more complex than travel reservation systems [4] for example, and presents a major challenge to both speech recognition and language understanding.

Rapid prototyping of natural-language dialogue applications is highly influenced by the availability of data for training recognition and understanding models. In this paper we consider the scenario when data is either limited or unavailable for building one component of our understanding system, namely the module for identifying a user's request. We consider the task of detecting a user's request as a multi-label classification problem and apply BoosTexter, a member of the boosting family of algorithms that was first proposed by Freund and Schapire [2, 6]. BoosTexter is strictly data driven. It combines many simple and moderately accurate categorization rules that are trained sequentially into a single, highly accurate model that can accurately predict a class. It has been shown to outperform traditional methods for text categorization [7].

There are two principles we have examined: (1) Learning from existing data to generate new corpus that is relevant to a different task; (2) Balancing human expertise ranging from pragmatic knowledge to application specific rules with the statistics of the training data. Both strategies aim at rapidly prototyping spoken dialogue services when the amount of data available is severely limited, and can potentially support new service requests for which no data is available.

There are several possible schemes for learning from existing data to expand the training corpus. One approach is to use a Synonym dictionary look-up. Phrases such as "I'd like to" and "I want to", for example, would be considered as synonyms that can be interchanged in order to expand the size of the training corpus [1]. Another approach which is more syntactic-based is to define a similarity measure across phrases, interchanging them if they belong to the same cluster. Clusters are computed by encoding phrases using context-dependent feature vectors and applying a Mahalanobis distortion measure to compute their distance apart [5]. A third approach for expanding the training set is to turn an existing phrase grammar into a finite-state generator [5]. Unsupervised learning can then be applied for training the classier as it is commonly done in automatic speech recognition.

A more direct approach for bootstrapping and building robust and accurate text classifiers is by combining application knowledge with available training data. In this paper, we extend BoosTexter to combine and balance pragmatic knowledge of the application with the statistics of the training data. The objective is to encode rough application knowledge and human judgment by a set of hand-crafted rules, and to refine, reinforce and adjust these rules by the statistics of available training data.

We present two sets of experiments using BoosTexter, one for a customer care application and the other for a Help Desk application. Our results show that including expert knowledge when data is limited can help us reduce the amount of data necessary for building these applications by factors of 2 to 4. We further show that even in the presence of no data, the same paradigm can be applied to improve classification performance in the event that new classes are being added to the system.

2. BOOSTEXTER

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. 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 $+1$.

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* [2, 6]. In particular, we use a variant of Schapire and Singer's BoosTexter system [8].

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 [7], 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."

The real-valued predictions of the final classifier f can be converted into probabilities by passing them through a logistic function; that is, we can regard the quantity $(1 + e^{-f(x)})^{-1}$ as an estimate of the probability that x belongs to class $+1$. In fact, the boosting procedure here described is designed to minimize the negative conditional log likelihood of the data under this model, namely,

$$\sum_i \ln(1 + e^{-y_i f(x_i)}). \tag{1}$$

The extension of BoosTexter to the multiclass problem is described in [7, 8].

2.1. Incorporating human knowledge

Boosting, like many machine-learning methods, is entirely data-driven in the sense that the classifier it generates is derived exclusively from the evidence present in the training data itself. When data is abundant, this approach makes sense. However, in some applications, data may be severely limited, but there may be human knowledge that, in principle, might compensate for the lack of data.

In its standard form, boosting does not allow for the direct incorporation of such prior knowledge. In this section, we describe a modification of boosting that combines and balances human expertise with available training data. We aim for an approach that allows the rough human judgments to be refined, reinforced and adjusted by the statistics of the training data in a well controlled manner.

As before, we limit our attention to binary classification; extension to multiclass problems can be made along the lines of Schapire and Singer [7, 8]. In our approach, a human expert must begin by constructing a rule p mapping each instance x to an estimated probability $p(x) \in [0, 1]$ which is interpreted as the guessed probability that instance x belongs to class $+1$. We discuss below some methods for constructing such a function p.

To apply boosting using p and a training set, we create a new *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 $\eta p(x_i)$ and $\eta(1 - p(x_i))$, respectively, where η is a parameter of the algorithm controlling the confidence in encoding knowledge. During training, these weights w_0 are 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).

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

$$h_0(x) = \ln\left(\frac{p(x)}{1 - p(x)}\right)$$

and include h_0 in computing the final classifier f.

Essentially, these modifications have the effect of changing the objective function in (1) to one that incorporates prior knowledge, namely,

$$\sum_i \ln\left(1 + e^{-y_i f(x_i)}\right) + \eta \sum_i \mathrm{RE}\left(p(x_i) \ \| \ \frac{1}{1 + e^{-f(x_i)}}\right)$$

where $\mathrm{RE}\,(p \ \| \ q) = p \ln(p/q) + (1 - p) \ln((1 - p)/(1 - q))$ is binary relative entropy. Thus, we balance the conditional likelihood of the data against the distance of the data-generated model from the model provided by the human. The relative importance of the two terms is controlled by the parameter η.

3. PRIOR KNOWLEDGE

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", or "fine", etc. is highly indicative of a positive confirmation. We can formally express this by a rule with an estimated probability, of say 0.9:

- yes|okay|correct|all right|fine $\rightarrow$ class(Yes, .9).

Another example of applying prior knowledge is for classifying users requests to be connected to an operator/service agent. This can be expressed by the following rule:

- speak & (human|operator|(service & agent)) $\rightarrow$ class(Agent, .95).

In general, we encode knowledge using rules with three logical operators, OR, AND and NOT. Rules are mapped onto a set of BoosTexter 0-th base classifiers h_0. Each of these classifiers is assigned a $p(x)$ value which is interpreted as our confidence that a given rule belongs to class +1. The values of $p(x)$ are then used to compute $h_0(x)$ as explained in section 2.1.

4. EXPERIMENTS

In this section, we describe and analyze the experiments we performed using BoosTexter for text categorization with in-domain knowledge provided by the application guidelines.

We ran experiments with data from two different applications. The first database used is the *How May I Help You?*^{sm} database [3]. In this task, there are 15 different classes. We did experiments with 50 to 1600 sentences in the training set and 2991 sentences in the test set. The second database is for an application called "*TTS Help Desk*". This application provides information about AT&T Text-to-Speech engine. There are 22 different classes. We trained models on 100 to 2675 sentences and tested on 2000 sentences.

In the first experiment that was performed on the *How May I Help You?*^{sm} database, we measured the classification accuracy as a function of the number of examples used during training. The classification accuracy is the percentage of sentences with a correctly predicted label. There is no rejection and the label with the highest score is kept as the predicted label. We compared models built only with some training examples and models built with both hand-crafted rules (prior knowledge) and training examples. We used approximately one rule per class where a rule is a combination of many words or phrases. 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

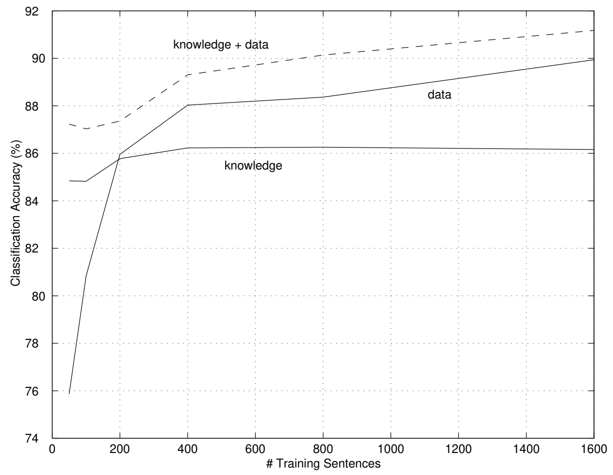

Figure 1: Comparison of performance using data and knowledge separately or together on the How May I Help You task.

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 Figure 1 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 becomes 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.

The second experiment was performed on the Help Desk task. Figure 2 shows the comparison between models built with hand-crafted rules and training examples and models built only with training examples. In this experiment there is also about 1 rule per class where a rule is again a combination of many words and phrases. We trained the models on 100, 200, 400, 600, 800, 1000 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 to 1600 and to 0.01 otherwise.

Figure 2 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

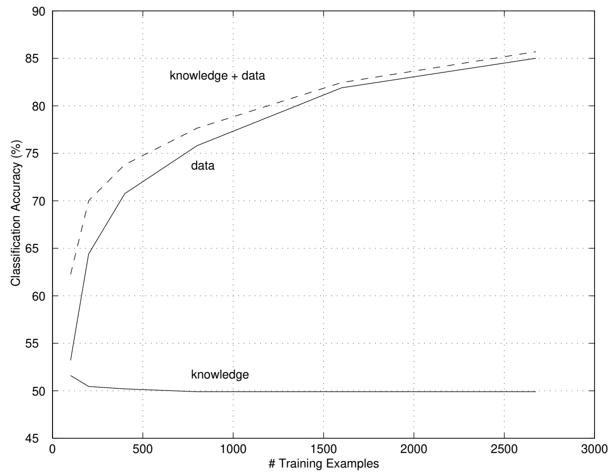

Figure 2: Comparison of performance using data and knowledge separately or together on the Help Desk task.

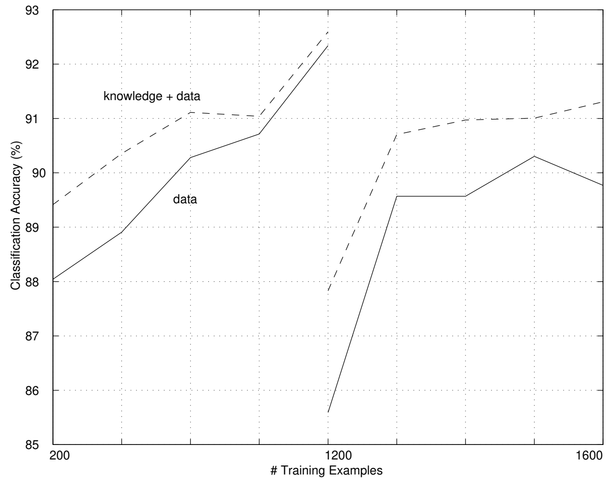

Figure 3: Adding new semantic classes following model training.

drops to 0.5% when more data becomes available.

We can notice from the experiments that the number of rounds and the choice of the parameter η are dependent on the number of training examples available. We can also notice that the knowledge-only curves are not perfectly flat. This comes from the fact that the models from the knowledge take into account the class distribution of the available training examples.

A further experiment was performed to evaluate the accuracy of our classifier when new semantic classes are being added following system training. This is the situation when new functionalities are needed following system deployment but with no data available. Figure 3 shows the classification accuracy when four additional semantic classes are added to the *How May I Help You?*^{sm} 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.

5. SUMMARY

The use of BoosTexter for text categorization in natural-language understanding was described in this paper. We presented an extension to BoosTexter that incorporates human knowledge of the application in the form of hand-crafted rules. Each set of rules, associated with estimated probabilities of the distribution of the class, is refined and enhanced by the statistics of the training data. A new objective function was proposed which has an additional relative entropy term that balances the conditional likelihood of the data against the distance of the data-generated model from that obtained using human knowledge.

Two experiments were conducted on speech data collected from natural-language dialogue applications. Both experiments demonstrate that introducing prior knowledge of the domain can significantly cut the amount of data needed for building the application. For both applications, exploiting human expertise helped to reduce reliance on the data by factors of two to four. A further investigation showed that the proposed paradigm using human judgment helps to improve classification performance when new functionalities are needed during service evolution.

References

[1] Hiyan Alshawi and Shona Douglas. Variant transduction: A method for rapid development of interactive spoken interfaces. In *Proc. SIGDial Workshop on Discourse and dialogue*, 2001.

[2] 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.

[3] A.L. Gorin, G. Riccardi, and J.H. Wright. How May I Help You? *Speech Communication*, 23:113–127, 1997.

[4] E. Levin, , S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. The AT&T darpa communicator mixed-initiative spoken dialogue system. In *ICSLP*, 2000.

[5] G. Riccardi and S. Bangalore. Automatic acquisision of phrase grammars for stochastic language modeling. In *ACL Workshop on Very Large Corpora Proc.*, pages 188–196, 1998.

[6] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

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

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