EXPLOITING UNLABELED DATA USING MULTIPLE CLASSIFIERS FOR IMPROVED NATURAL LANGUAGE CALL–ROUTING

Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Vaibhava Goel and Yuqing Gao
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{sarikaya,hkuo,vgoel,yuqing}@us.ibm.com

ABSTRACT

This paper presents an unsupervised method that uses limited amount of labeled data and a large pool of unlabeled data to improve natural language call routing performance. The method uses multiple classifiers to select a subset of the unlabeled data to augment limited labeled data. We evaluated four widely used text classification algorithms; Naive Bayes Classification (NBC), Support Vector machines (SVM), Boosting and Maximum Entropy (MaxEnt). The NBC method is found to be poorest performer compared to other three classification methods. Combining SVM, Boosting and MaxEnt resulted in significant improvements in call classification accuracy compared to any single classifier performance across varying amounts of labeled data.

1. INTRODUCTION

One of the many applications of speech understanding is natural language call–routing. The goal of call–routing is to understand the speaker's request and take the appropriate action. Typically, natural language call–routing requires two statistical models. The first performs speech recognition to transcribe what the caller says. The second is the Action Classification (AC) model that takes the spoken utterance generated by the speech recognizer and predicts the correct action to fulfill speaker's request. Actions are the categories into which each caller request are mapped. Building a robust and highly accurate call–routing system requires large amounts of manually transcribed and labeled data from the domain of the application. The human labeler assigns one or more of the predefined classes (call–types) to each utterance. This requires full understanding of the domain of application. Naturally, it is tedious and expensive to manually label large amounts of data.

Machine learning algorithms are quite successful in text classification when provided with sufficient labeled data. However, as the complexity of the task increases required training data for reasonable performance can become large. This increases the cost and time to deploy the natural language understanding system. In order to solve this problem we look for ways to use the limited labeled data efficiently to exploit a large pool of unlabeled data. Active learning methods deal with this same problem in an alternative way [12]. Active learner selects utterances for which they require true class labels. The key issue is to identify those utterances that are most informative to the classification task.

Typically labeled data is scarce, whereas unlabeled data is easily obtainable in large amounts. There are a number of studies attempting to exploit unlabeled data for improved classification performance. In [6], Expectation Maximization algorithm is used in conjunction with the Naive Bayes classifier to take advantage of a large pool of unlabeled data. In [2], automatically labeled examples are combined at the model and data level with the initial manually labeled examples to improve classification. For data–level combination only those utterances that obtained a confidence score above a pre–determined threshold are used to augment the initial training data. As for the classifier they used a Boosting–style algorithm [3]. In [1], a Bayesian classifier and a Boosting–style classifier are combined in various ways to improve AC accuracy using unlabeled data.

In this study, we motivate a multiple classifier (≥ 3) combination strategy for automatic learning of the labels that would otherwise require costly human labor. The input to this method is a large amount of unlabeled data and a small amount of labeled data to inform the classifiers about the specific task at hand. Bootstrapping initializes each classifier with the limited labeled data. It then combines the hypothesis from the classifiers in a voting scheme to select a subset of the unlabeled sentences that are reliably classified. These utterances with their predicted labels are used to augment the labeled data. Multiple classifiers employing different learning algorithms are effective in making fairly independent errors. Previously, we used a similar framework for Semi–automatic semantic annotation of the parse trees for speech understanding [4]. There, we found that combining multiple classifiers is always better than using one or two classifiers with confidence thresholding. We considered four classifiers in this study. Besides SVM, MaxEnt and Boosting we also implemented Naive Bayes Classifier employing EM to take advantage of unlabeled data. NBC was shown to be an effective method in combining evidence from labeled and unlabeled data for text document classification [6, 7].

This paper is organized as follows: Section 2 describes the four classification algorithms used in this paper. Section 3 presents the experimental results and discussion followed by the conclusions in Section 4.

2. CLASSIFICATION METHODS

We investigated four widely used classification algorithms for text classification; Boosting, Support Vector Machines, Maximum Entropy, and Naive Bayes classification with Expec-

tation Maximization. Now, we briefly describe these algorithms.

2.1. BOOSTING

Boosting is an iterative method for improving the accuracy of any given learning algorithm. The premise of Boosting is to produce a very accurate prediction rule by combining moderately inaccurate (weak) rules [3]. The algorithm operates by learning a weak rule at each iteration so as to minimize the training error rate.

A specific implementation of the Boosting is AdaBoost. The pseudo code for AdaBoost for binary classification is given below [3]:

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$ (X is some domain or instance space and Y is label set).

Initialize the distribution, $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:
1. Train weak learner using distribution D_t.
2. Get weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ with error $\epsilon_t = P_{i \sim D_t}[h_t(x_i) \neq y_i]$
3. Choose $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
4. Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned} \quad (1)$$

where Z_t is a normalization factor ensuring D_{t+1} is a distribution. The output of the final classifier is given below.

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (2)$$

2.2. MAXIMUM ENTROPY

The MaxEnt method is a flexible statistical modeling framework that has been used in widely in in many areas of natural language processing [9]. The MaxEnt allows the combination of multiple overlapping information sources [10, 9]. The information sources are combined as follows:

$$P(C|W) = \frac{e^{\sum_i \lambda_i f_i(C,W)}}{\sum_{C'} e^{\sum_j \lambda_j f_j(C',W)}}, \quad (3)$$

which describes the probability of a particular class C (e.g. action class) given the word sequence W spoken by the caller. Notice that the denominator includes a sum over all classes C', which is essentially a normalization factor for probabilities to sum to 1. The f_i are indicator functions, or *features*, which are "activated" based on computable features on the word sequence, for example if a particular word or word pair appears, or if the parse tree contain a particular tag, etc. For simplicity, we only use unigram (single word) features in this paper, also commonly known as a "bag of words" model. The MaxEnt models are trained using the improved iterative scaling algorithm [10] with Gaussian prior smoothing [9] using a single universal variance parameter of 2.0.

2.3. SUPPORT VECTOR MACHINES

The SVMs are derived from the theory of structural risk minimization [5]. The SVMs learn the boundaries between samples of the two classes by mapping these sample points into a higher dimensional space. In the high dimensional space a hyperplane separating these regions is found by maximizing the margin between closest sample points belonging to competing classes.

The theory behind SVM is briefly summarized (for linearly separable case) below. An in-depth discussion of SVMs can be found in [5]. Given: $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)$ where $\mathbf{x_i} \in \mathbf{X}, y_i \in Y = \{-1, +1\}$ ($\mathbf{X}$ training sample space and Y is label set). The hyperplane $(\mathbf{w} \cdot \mathbf{x}) + b$ corresponding to decision functions

$$f(x) = \text{sgn}((\mathbf{w} \cdot \mathbf{x}) + b) \quad (4)$$

separates the classes iff

$$(\mathbf{w} \cdot \mathbf{x_i}) + b > 0 \text{ if } y_i = 1 \quad (5)$$

$$(\mathbf{w} \cdot \mathbf{x_i}) + b < 0 \text{ if } y_i = -1 \quad (6)$$

Rescaling $\mathbf{w}$ and b such that the points closest to hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x_i}) + b| = 1$, we obtain a canonical form $(\mathbf{w}, b)$ of the hyperplane that satisfies:

$$y_i \cdot ((\mathbf{w} \cdot \mathbf{x_i}) + b) \geq 1 \quad (7)$$

In order to construct the optimum hyperplane we have to solve the following optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}|\mathbf{w}|^2 \quad (8)$$

subject to Eqn. 7. Eqn. 8 can be solved through its Lagrangian dual:

$$\max_{\alpha_i \geq 0} (\min_{\mathbf{w},b} (\frac{1}{2}|\mathbf{w}|^2 - \sum_i \alpha_i (y_i \cdot ((\mathbf{w} \cdot \mathbf{x_i}) + b) - 1) \quad (9)$$

The dual problem can be simplified to finding the multipliers α_i [5]:

$$\max_{\alpha_i} (\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}) \quad (10)$$

Here, α_i are Lagrange multipliers belonging to each training sample point. The above equation is subject to the following constraint:

$$\begin{aligned} \alpha_i &\geq 0 \\ \sum_i \alpha_i y_i &= 0 \end{aligned} \quad (11)$$

Those training samples for which α_i are nonzero are called support vectors.

Much of the flexibility and classification power of SVM's resides in the choice of kernel. Some of the commonly used kernels are linear, polynomial and radial basis functions. In this work, we chose linear kernels to train the SVM since computationally it is faster compared to other kernels yet there is no significant difference in performance for the current application.

2.4. NAIVE BAYES CLASSIFIER WITH EM

The Naive Bayes classification (NBC) algorithm is a generative probabilistic modeling technique. In this method documents are assumed to be generated by a mixture of multinomials. Its appeal comes from the fact that it can be combined with the Expectation Maximization algorithm [8] to exploit unlabeled data. NBC has been used extensively for document classification [6, 7]. For some document classification tasks combining it with EM (NBC–EM) resulted in additional improvements over NBC. The defining equations of NBC–EM for the simple case of single mixture per class are given below. The estimated probability of word (w_t) in the document given the class the document belongs to is as follows [6, 7]:

$$\mathrm{P}(w_t|c_j;\hat{\theta}) = \frac{1+\sum_{i=1}^{|D|} N(w_t,d_i)\mathrm{P}(c_j|d_i;\theta)}{|V|+\sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N(w_s,d_i)\mathrm{P}(c_j|d_i;\theta)} \quad (12)$$

where $|V|$ is the vocabulary size, $|D|$ is document count, $N(w_t, d_i)$ is the count of the number of times w_t occurs in document d_i and $\mathrm{P}(c_j|d_i)$ is the posterior probability of document class c_j given document d_i. The class prior probabilities, $\hat{\theta}_{c_j}$ are computed similarly but without smoothing:

$$\hat{\theta}_{c_j} = \frac{\sum_{i=1}^{|D|}\mathrm{P}(c_j|d_i)}{|D|} \quad (13)$$

Given the estimates of word and class probabilities obtained using the training data, application of Bayes' rule turns the generative model around to estimate the probability of a class model given a document.

$$\begin{aligned}\mathrm{P}(c_j|d_i;\hat{\theta}) &= \frac{\mathrm{P}(c_j|\hat{\theta})\mathrm{P}(d_i|c_j;\hat{\theta})}{\mathrm{P}(d_i|\hat{\theta})} \\ &= \frac{\mathrm{P}(c_j|\hat{\theta})\Pi_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_j;\hat{\theta})}{\sum_{r=1}^{|C|}\mathrm{P}(c_r|\hat{\theta})\Pi_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_r;\hat{\theta})} \quad (14)\end{aligned}$$

The class that achieved the highest posterior probability for the test document d_i is selected. For NBC–EM, the E–step corresponds to estimating probabilistic labels using Eqn. 14, the M–step, maximizing the likelihood corresponds estimating a new maximum a posteriori estimate for the parameters, $\hat{\theta}$, using the current estimates, $\mathrm{P}(c_j|d_i;\hat{\theta})$, and Eqn. 12 and 13. Extensions of these equations to multiple mixtures per class can be found in [7].

3. EXPERIMENTAL RESULTS AND DISCUSSION

The corpus used in our experiments is from a call–center customer hotline technical assistance for a Fortune–500 company [11]. The natural language call–routing system selects one of the 35 call–types. The training data has 27K utterances amounting to 178K words. This data is split into {1K, 2K, 3K, 4K, 5K, 6K, 7K, 8K, 9K, 10K} and 17K sets. The first ten sets are used as labeled examples and the 17K set is treated as unlabeled data. A separate data set containing 5644 utterances is used as the test set. All of these data are hand–labeled with call–types. In all the classification methods employed here we used word unigrams as the feature for modeling.

Labeled Data	Mixture Count	Accuracy
1K	35	58.14
2K	35	63.56
	38	68.46
3K	35	67.32
	39	70.72
4K	35	69.13
	49	70.41
5K	35	71.12
	50	70.41
6K	35	72.70
	53	74.63
7K	35	74.16
	59	73.52
8K	35	75.59
	66	74.59
9K	35	76.55
	66	75.29
10K	35	77.20
	68	75.99

Table 1: Action Classification (AC) Accuracy for Naive Bayes Classifier (%)

As mentioned earlier NBC lends itself as an interesting method since it can be integrated with the EM–algorithm to take advantage of unlabeled data. NBC–EM was previously applied successfully to text (document) classification of several tasks including, 20–Newsgroup, WebKB, Reuters [6, 7]. The main difference between these tasks and our task is the definition of a document. There, documents in general are composed of tens of sentences whereas here each sentence is assumed to be a document. We believe this study is the first application of NBC–EM to such a limited domain utterance classification task. The results are given in Table 1, show that as labeled data increases performance improves. These results are obtained on the 17K heldout data. We implemented NBC with single as well as multiple mixtures per class. The number of mixture components are determined empirically based on the number of sentences belonging to each class. Using multiple mixtures per class helps to improve the results mostly from 1K–4K labeled data and degrades the results for others. There may be many factors contributing to this outcome. For example, data distribution for each class may not be multi–modal. Data sparsity is a critical issue for robust model estimation. Unlike NBC, SVM, MaxEnt and Boosting are considered as discriminative modeling techniques. NBC obtained poorer performance compared to the other three classifiers (Table 2). We further implemented the NBC–EM, but application of EM degraded AC accuracy while improving the model likelihood. There may be several reasons for this such as the assumptions of the generative model, which are not realistic for this application, or data sparseness that severely impacts model robustness [7].

In Table 2, we present the results for SVM, MaxEnt, Boosting classifiers using the test data. Each classifier is trained using the labeled data amount given in the first column. Then, classifiers are used to label the 17K unlabeled set. If all three classifiers predicted the same label those sentences are selected to be combined with the labeled data. The classifiers

Action Classification (AC) Accuracy for SVM, MaxEnt and Boostexter Classifiers and Their Combinations (%)											
Labeled Data	SVM Base	SVM Rover	SVM Bound	MaxEnt Base	MaxEnt Rover	MaxEnt Bound	Boosting Base	Boosting Rover	Boosting Bound	Accepted Error	Accepted Sents
1K	77.88	79.76	89.63	75.99	78.69	88.71	79.63	80.20	88.31	1580	13177
2K	82.21	84.47	89.72	80.37	82.03	88.87	83.60	84.31	88.29	1245	13703
3K	84.31	85.77	89.77	82.19	84.07	89.07	85.12	85.37	88.45	1146	14037
4K	85.27	86.33	89.74	83.52	84.69	88.96	84.61	85.64	88.49	1091	14118
5K	86.21	87.10	89.90	84.64	85.67	89.01	85.92	86.95	88.59	1038	14358
6K	86.95	87.34	89.93	85.47	86.34	89.14	86.27	87.28	88.47	931	14397
7K	87.66	88.11	89.98	86.20	86.75	89.28	86.26	87.44	88.52	895	14459
8K	88.00	88.51	90.20	86.46	87.26	89.28	87.21	87.76	88.88	854	14550
9K	88.51	88.92	90.45	87.21	87.79	89.53	87.50	88.03	88.81	872	14643
10K	88.53	89.06	90.34	87.62	88.00	89.69	87.66	88.08	88.08	836	14681

Table 2: AC Accuracy for Rover–Based Classifier Combination.

are retrained using the updated training material. In Table 2 "Base" (i.e. SVM–Base) refers to AC accuracy obtained on the test data using only limited labeled data. "Rover" (i.e. SVM–Rover) refers to AC accuracy results on the test data where classifiers are trained using the augmented training material. We also present the upper–bounds for the possible performance limits that are obtained by lumping labeled data and unlabeled data with their true labels. This is shown with "Bound" (i.e. SVM–Bound) in the table. For example, the upper–bound for 1K labeled data and 17K unlabeled data is obtained using 18K (1K+17K) labeled data. SVM–Bound changes from 89.63% for 1K (18K) to 90.34% 10K (28K) labeled data. For MaxEnt the corresponding figures are 88.71% and 89.69%, respectively. Unlike SVM and MaxEnt, for Boosting the figure for 1K labeled data is higher (88.31%) than that of the 10K labeled data (88.08%). On the right–most column we presented the total number of utterances where all three classifiers resulted in the same call–type. Since these sentences are accepted to augment the labeled data they are tagged as "Accepted Sentences". The second column from the right shows the number of errors in the accepted sentences. The error rate in the accepted sentences range from 12% for the 1K labeled data to 5.7% for the 10K labeled data.

The SVM classifier obtained 77.88% accuracy as the baseline (SVM–Base) using 1K labeled data. Using classifier combination (SVM–Rover) improved the accuracy to 79.76% (1.8%). The corresponding improvement for MaxEnt is 2.7%. Boosting improved the least by about 0.6%. For 10K labeled data the corresponding numbers are 0.5%, 0.4%, 0.4%, for SVM, MaxEnt and Boosting, respectively. For Boosting, we also observed that its starting performance (for 1K) is better than SVM and MaxEnt. However, as we use more data SVM caught up and then outperformed Boosting whereas MaxEnt narrowed the performance gap. We observed the same behavior for Boosting upper–bound where the upper–bound for 1K was better than that of the 10K. We also experimented with thresholding the scores from MaxEnt and Boosting to be more selective among the sentences that are agreed. However, this resulted in slight improvement on some classification methods while degraded the others. Therefore, we used all the sentences that are agreed on the call–type.

4. CONCLUSIONS

We presented an unsupervised method to take advantage of a large pool of unlabeled data to improve the performance of a natural language call–routing system trained used limited labeled data. This method employed multiple widely used classification algorithms to select a subset of the unlabeled data to augment limited labeled data. The classification algorithms evaluated in the experiments were Support Vector Machines (SVM), Maximum Entropy (MaxEnt), Boosting and Naive Bayes Classification (NBC). Experimental results demonstrated that NBC did not perform as well as the other classifiers for this task. For this reason we left out NBC and combined the remaining three classifiers. Combining the classifiers improved action classification accuracy compared to single classifier performance consistently across a wide range of labeled data amounts. In general the improvement were slightly larger for SVM and MaxEnt compared to Boosting.

References

[1] M. Karahan, D. Hakkani-Tur, G. Riccardi, G. Tur, "Combining Classifiers for Natural Language Understanding", *IEEE ASRU-2003*, US Virgin Islands, Dec. 2003.

[2] G. Tur, D. Hakkani-Tur, "Exploiting Unlabeled Utterances for Spoken Dialog Understanding", *Eurospeech-2003*, Geneva, Switzerland , Sept. 2003.

[3] R. E. Schapire and Y. Singer, Boostexter: A boosting based system for text categorization, *Machine Learning*, vol. 39, no. 2/3, pp. 135168, 2000.

[4] R. Sarikaya, Y. Gao and P. Virga, "Fast Semi-Automatic Semantic Annotation for Spoken Dialog Systems", *ICSLP-2004*, Jeju Island, South Korea.

[5] V. Vapnik, "The Nature of Statistical Learning Theory", *Springer–Verlag*, NY, USA, 1995.

[6] K. Nigam, A. McCallum, S. Thrun and T. Mitchell, "Text classification from labeled and unlabeled documents using EM", *Machine Learning*, 39, 2000.

[7] K. Nigam, "Using unlabeled data to improve text classification", *Ph.D. Thesis*, Carnegie Mellon University, 2001.

[8] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM Algorithm", *J. Royal Stat. Soc.* Series B 39:1–38, 1997.

[9] S. Chen and R. Rosenfeld, "A survey smoothing techniques for ME models", *IEEE Trans. SAP*, 8(1):37–50, 2001.

[10] S. D. Pietra, V. D. Pietra and J. Lafferty, "Inducing features of random fields", *IEEE Trans. Pattern. Analysis Mach. Int.*, 19(4):380–93, 1997.

[11] V. Goel, H-K.J. Kuo, S. Deligne and C. Wu, "Language model estimation for optimizing end–to–end performance of a natural language call routing system", *ICASSP-2005*, Philadelphia, PA 2005.

[12] G. Tur, R.E. Schapire and D. Hakkani-Tur, "Active Learning for Spoken Language Understanding", *ICASSP-2003*, Hong Kong, May. 2003.