Combining Multiclass Maximum Entropy Text Classifiers with Neural Network Voting

Philipp Koehn
koehn@isi.edu

[1] Whizbang! Labs, Provo, UT 84604, USA
[2] Information Sciences Institute, USC, Marina del Rey, CA 90292, USA

Abstract. We improve a high-accuracy maximum entropy classifier by combining an ensemble of classifiers with neural network voting. In our experiments we demonstrate significantly superior performance both over a single classifier as well as over the use of the traditional weighted-sum voting approach. Specifically, we apply this to a maximum entropy classifier on a large scale multi-class text categorization task: the online job directory Flipdog[3] with over half a million jobs in 65 categories.

1 Ensemble Learning

For classification problems, supervised learning methods train a classifier on a set of labeled training examples which fall into several classes. The classifier can then be used to predict the class of a new instance. Each instance is represented by a set of features, which have to be carefully chosen.

For example: The task may be to classify job descriptions into several categories such as Chemical Engineering or Hospitality/Recreation (see Figure 1). In this case, the words in the description can be used as features. The classifier tries to learn which words (or combination of words) predict the category of the job description.

This type of problem has been called **Text Categorization**. An excellent overview of this field is presented by Sebastiani [12]. This problem has previously been addressed with Maximum Entropy Classification [9]. In this paper we will point out a weakness of this method and show how to overcome it by using an ensemble of maximum entropy classifiers trained on different feature sets.

A recent thread in machine learning research concerns itself with **Ensemble Learning**: instead of training a single classifier, a set (or ensemble) of classifiers is used. The classifiers are trained on different sets of training examples, most often using the same learning algorithm. Subsequently, the classifiers are combined by voting.

This seemingly simple idea has been applied to a variety of problems and learning methods. Consistently, superior results are obtained opposed to using just one classifier trained on all the training examples.

[3] Available online at http://www.flipdog.com/. Job descriptions are collected from company web sites.

Slot Technician Graduate of mechanical program and one to two years electronics experience preferred. Applies electrical theory and related knowledge to test and modify electrical gaming machinery and equipment. Requires frequent standing, walking, reaching, stooping and crouching; excellent hand-eye coordination and fine motor hand and wrist movements. Good close, color and peripheral vision required. Must be capable of lifting/carrying weights of up to 100 pounds. Moderate to loud noise level conditions.

Category: Hospitality / Recreation

Fig. 1. Example of an job description, as used in the experiments of this paper.

Some general strategies in this approach have emerged: In **Bagging** [2], subsets of the training data are constructed by randomly selecting training examples. For each of the subsets a classifier is trained. The classifiers are combined by averaging the predictions of the single classifiers.

Boosting [5] iteratively defines new training sets and trains a classifier on these. This classifier is typically called a **Weak Learner** – we will also use this terminology in this paper, even though the maximum entropy learner we use is clearly not a weak learning algorithm. First, the weak learner is trained on the entire training set. Then, an instance weight is assigned to each training example, which is higher for examples that have been misclassified, although they are part of the training set. A second weak learner is trained on this weighted training set. The weighting of the training set and the training of a new weak learner is done for a number of rounds. Ultimately, a final classifier is formed by combining the weak learners. Boosting has been used with many learning algorithms acting as a weak learner, ranging from simple feature detectors [13] to decision trees [11].

In addition to bagging and boosting, there has also been research on combining a more heterogeneous set of weak learners. Different training algorithms may be combined – see the learning scheme by Goldman and Zhou [6], or work by Zhang et al. [15] as well as Larkey and Croft [7]. Herein, we propose to combine classifiers trained by the same algorithm – maximum entropy – but using a different feature set for each classifier.

The formula for **Voting** – combining the results of classifiers – usually takes the form of a weighted sum (or weighted linear combination):

$$confid(class) = \sum_i weight_i * confid_i(class) \tag{1}$$

Each weak learner i provides a confidence value $confid_i(class)$ for its prediction of a certain class. For each class, these confidence values are weighted by a factor $weight_i$ and combined to an overall confidence value $confid(class)$. Finally, the class with the highest combined confidence is the predicted class of the combined classifier.

2 Neural Network Voting

Our approach views voting as a separate classification task: the correct class is to be determined by a number of input values, namely the confidence values from each weak learner for each class. Wolpert [14] introduces this view on combining classifiers, calling it **Stacked Generalization**.

Many different machine learning methods could be used for combining weak learners. Domingos [3] proposes Bayesian averaging, where the weights for the weighted sum is computed based on the probability of a weak learner given an example with its class assignment. Pennock et al. [10] provide a theoretical treatment of this issue, arguing for weighted sum voting. Zhang et al. [15] use a neural network to combine a statistical model, a memory-based learner and a neural network for protein secondary structure prediction. An overview to the issues in combining classifiers is presented by Alpaydın [1].

We chose neural network back-propagation as the learning algorithm for this task, due to its ability to work well with real-numbered values and its ability to cope with a large number of inputs without need for model simplifications.

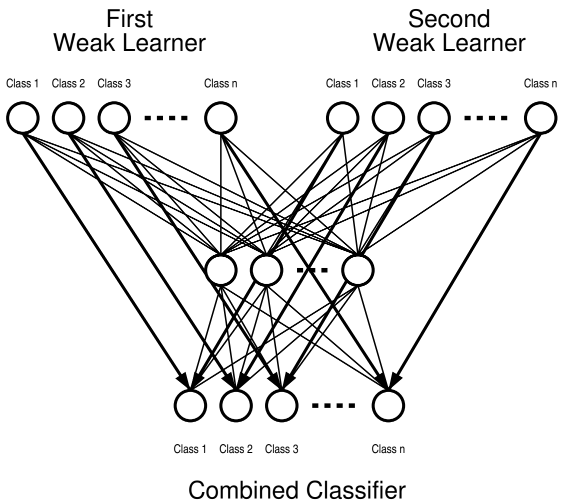

Fig. 2. The architecture of the neural network that combines the classification results of the weak learners.

Figure 2 shows the network architecture. We use a standard multi-layered network with a simple addition. The confidence values for each class and weak learner are used as input nodes. The output layer provides confidence values for each class. The input layer is fully connected with a hidden layer, which in turn is fully connected with the output layer. In addition to this, each output node receives incoming connections from the input nodes of the corresponding class. This addition results in increased accuracy of 1.0%.

Training in this framework requires splitting the labeled data into three sets:

- a **Primary Training Set** is used for training the weak learners.
- a **Secondary Training Set** is classified by each weak learner, providing the training data for neural network voting.
- a **Testing Set** is used for evaluating neural network voting on the weak learners.

The split of the training set into a primary and secondary training set is motivated by the following: The maximum entropy classifiers already reach almost perfect performance on the training sets. If we would train the neural network on these classification decisions for the same training examples, it would not likely learn anything useful. By using a previously unseen secondary training set, the neural network will have a better chance to learn when to trust which classifier, since they disagree more often.

3 Maximum Entropy Classification

We use a maximum entropy classifier as the weak learner for our text classification problem. A good introduction to maximum entropy classification is presented by Manning and Schütze [8]. Generally speaking, maximum entropy learning ensures the expected frequency with which a feature occurs in a class, as seen in the training data, to be the same as the corresponding probability in the model of the training data.

While observing the constraints of matching these empirical expectations, the classification model makes no further assumptions. In other words, it maintains maximum entropy.

Maximum entropy classification is similar to Naive Bayes. However, it overcomes some effects of Naive Bayes' problematic independence assumption[4]. To illustrate this point: Two correlating features in Naive Bayes learning would falsely contribute with doubled strength to classification decisions. Maximum entropy, on the other hand, ensures the correct frequency of both features in the model by distributing probability mass between them.

The classifier applied herein is described in detail by Nigam et al. [9]. We use as features words, bigrams, and trigrams occurring in the document and in the title. We also tried to introduce syntactic features such as subject-object relationships, but we found no improvement.

[4] This independence assumption may not be very harmful [4]

4 Experiments

We will now describe our experiments which show the performance gains of neural network voting. Using 130,000 training examples, we classify job descriptions in 65 categories. We have to deal with mislabeled examples and borderline cases that lie between categories. Based on a small-scale analysis to assess inter-annotator agreement, we estimate that the best performance a classifier (human or machine) could achieve is roughly in the mid-80 percent accuracy range.

For the given task, the maximum entropy classifier already achieves very high accuracy. It performs superior to Naive Bayes, Error Correcting Output Codes layered over binary boosted decision trees, sequential covering rule learner and other algorithms[5]. The performance of the classifier is at 72.3% performance accuracy on a 30,000 example testing set. The 95% confidence interval for statistical significance is 72.3±0.5%.

An error analysis revealed that about half of the errors derive from mislabeled testing data or borderline cases. However, there is also a large number of examples where the title of the job description gives a clear indication of the job's category, while confusing words in the text of the description mislead the classifier. Apparently the classifier was not able to give sufficient weight to the job title.

Clearly, there is a much larger number of words in the text than in the title. Still, the inability to pick the title words as important features, even when they are marked up as such, is a surprising weakness of the maximum entropy classifier.

One idea would be to use boosting to address the discovered weakness. However, the classifier performs over 99 percent accurate on the training data. Since boosting increases the weight of mislabeled examples, and there are hardly any, not much can be expected from re-weighting the examples.

Instead, we train new classifiers on different feature sets. The most obvious, of course, is to train a classifier just on the job title. We are then faced with the issue of combining the original classifier with the classifier trained just on the title words. This is where we apply our neural network voting strategy, as described in Section 2.

For neural network voting, we split the original training set into a primary training set (115,000 training examples) and a secondary training set (15,000 training examples). Training of the neural network uses the secondary testing set, as described in Section 2.

Figure 3 shows the learning curve of the neural network algorithm over time. While the accuracy for the secondary training set is increasing consistently, the accuracy for the testing set increases initially, but then decreases slightly when over-fitting sets in.

We compare neural network voting and the use of a weighted sum for combining classifier. When using a weighted sum for combining the classifiers, we determined the weights by hand through experimentation. We tried a variety of

[5] According to private communication with Dallan Quass and Andrew McCallum

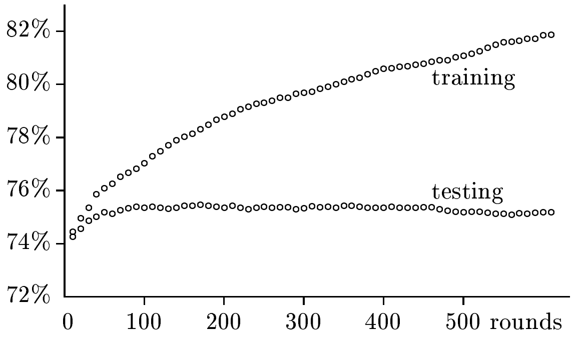

Fig. 3. Learning curve for neural network voting – both for the secondary training set and the testing set.

weight setting and found a large range of settings with very similar results on the testing set.

The results of our experiments on combining different types of classifiers are summarized in Figure 4. By adding a second classifier trained on titles alone using a weighted sum, we improve accuracy over the original classifier by 0.8%. Using neural network voting instead results in a 3.6% accuracy boost.

Classifiers	Voting Method	Accuracy
title	-	61.5%
original	-	72.3%
original + title	weighted sum	73.0%
original + title	neural network voting	75.9%

Fig. 4. Overview of Results: Single classifiers vs. combination of classifiers with neural network voting, weighted sum of confidence values.

5 Conclusions

The results of our experiments are twofold: Firstly, we discovered a weakness of maximum entropy learning for text classification and showed how to overcome it with an ensemble learning method. Secondly, we showed that the use of neural network voting is superior to the traditional use of a weighted sum for this type of model combination.

Acknowledgment

I would like to thank everybody at Whizbang! Labs for creating a great environment that enabled this research.

References

1. Ethem Alpaydın. Techniques for combining multiple learners. In *Proceedings of Engineering of Intelligent Systems*, volume 2, pages 6–12, 1998.
2. Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
3. Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In *Proceedings of the 17th International Conference on Machine Learning*, pages 223–230, 2000.
4. Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning*, pages 105–112, 1996.
5. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.
6. Sally Goldman and Yah Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning*, pages 327–334, 2000.
7. Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR 96)*, pages 289–297, 1996.
8. Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
9. Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI Workshop on Information Filtering*, 1999.
10. David M. Pennock, Pedrito Maynard-Reid, C. Lee Giles, and Eric Horvitz. A normative examination of ensemble learning algorithms. In *Proceedings of the 17th International Conference on Machine Learning*, pages 735–742, 2000.
11. J. R. Quinlan. Boosting first-order learning. In *Proceedings of the 7th International Workshop on Algorithmic Learning Theory*, pages 143–155, 1996.
12. Fabrizio Sebastiani. Machine learning in automated text categorisation. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999. Submitted for publication to *ACM Computing Surveys*.
13. Robert E. Shapire and Yoram Singer. Boostexter: A system for multiclass multi-label text categorization. Technical report, AT&T Labs – Research, 1998.
14. David H. Wolpert. Stacked generalization. *Neural Network*, 5:241–259, 1992.
15. X. Zhang, J. P. Mesirov, and D. L. Waltz. Hybrid system for protein secondary structure prediction. *Journal of Molecular Biology*, 225:1049–1063, 1992.