Sentiment Mining Using Ensemble Classification Models

Matthew Whitehead and Larry Yaeger
Indiana University
School of Informatics
901 E. 10th St.
Bloomington, IN 47408
{mewhiteh, larryy}@indiana.edu

Abstract

We live in the information age, where the amount of data readily available already overwhelms our capacity to analyze and absorb it without help from our machines. In particular, there is a wealth of text written in natural language available online that would become much more useful to us were we able to effectively aggregate and process it automatically. In this paper, we consider the problem of automatically classifying human sentiment from natural language written text. In this sentiment mining domain, we compare the accuracy of ensemble models, which take advantage of groups of learners to yield greater performance. We show that these ensemble machine learning models can significantly improve sentiment classification for free-form text.

I. INTRODUCTION

The amount of information being generated and stored on computers is growing rapidly [1], [2]. The explosion of the popularity of the internet and world wide web has all but eliminated the barrier for entry into publishing. For virtually no cost, just about anyone can record their thoughts on blogs, message boards, and personal web pages. More recently, many so-called Web 2.0 business models rely solely upon users to generate all of their meaningful content. Many of these websites do little more than make content publishing, organization, and access as simple as possible. The result is a massive increase in the amount of widely available human-generated, natural language data.

This wealth of data has the potential to significantly alter the way that we access, process, and use information. Increasingly, the challenge has become one of trying to make sense of the information available and organize it in such a way that it can used to maximum benefit.

Much of the difficulty is that so much of the useful data being generated by users online is generated in a communication medium that is easiest for humans to create and process, namely natural language. Because of this, much of the challenge lies in developing computer software that can process written natural language, aggregate it, organize it, and present it back to humans in meaningful and, often, more succinct ways.

A subset of this very difficult problem of natural language processing that is of particular interest is determining the sentiment of a piece of written text. The world wide web has numerous sites dedicated to collecting user opinions about a wide variety of subjects. There are sites for political opinions, movie reviews, restaurant reviews, music reviews, and more. A system that could automatically determine user opinions from a number of sources and then present a report showing the results in aggregate would be highly useful.

Consider the problem of trying to find the definitive performance of Bach's cello suites. Without a useful sentiment mining tool that aggregates user reviews, a user must read through numerous opinions of cello CDs to reach a final conclusion about which one to buy. This is time-consuming and inefficient. If the user could instead glance at a chart of cello CDs that shows how positive or negative the reviews were for that CD, then the search process would be much simplified. The system could present opinion results for thousands of reviews in just seconds.

In order for a tool like this to be useful, it must maintain a high level of classification accuracy. If reviews used in the system are not properly identified as being either positive or

negative, then the system quickly becomes more trouble than it is worth.

Machine learning algorithms are a natural fit for solving this sort of problem and building a useful classification system. Using a bag of words representation for user reviews allows the problem to be cast as a binary classification problem. In this case, the two categories are *positive* and *negative* and they reflect the sentiment of the accompanying user review. Of course reviews need not be entirely positive or negative, so we could also classify reviews using real numbers to show the degree to which each review is positive or negative. We chose to consider the simpler binary classification setup because we sampled real-world data and found that, at least for some bodies of data, very few reviews were truly mixed. For the datasets used in this study, only about 10% of sampled reviews had neither a strong positive nor a strong negative opinion.

In the recent past there has been considerable interest in the use of ensemble techniques in machine learning. Ensemble techniques, such as bagging [3] and boosting [4], use groups of learners to outperform a single learner.

In this paper, we show how a variety of ensemble methods perform in the sentiment mining domain. We believe that the resulting classification accuracies are high enough to be usable in practical settings.

II. RELATED WORK

Sentiment Mining

The area of sentiment mining (also called sentiment extraction, opinion mining, opinion extraction, sentiment analysis, etc.) has seen a large increase in academic interest in the last few years. Researchers in the areas of natural language processing, data mining, machine learning, and others have tested a variety of methods of automating the sentiment analysis process.

Pang et al. [5] researched sentiment mining using a binary unigram representation of patterns. In this representation, training patterns are represented by the presence/absence of words instead of by the count of total word occurrences. They tested a variety of algorithms for classification and found that a support vector machine had the highest accuracy of 82.9% using a movie reviews dataset. In later work, Pang and Lee [6] report improvement by adding a preprocessing filter to remove objective sentences which allowed the classifier to focus only on subjective sentences, raising the accuracy to 86.4%.

Whitelaw et al. [7] proposed improving sentiment mining pattern representations by using appraisal groups. They define appraisal groups as "coherent groups of words that express together a particular attitude, such as 'extremely boring', or 'not really very good'." By combining a standard bag-of-words approach with appraisal groups they report a 90.2% classification accuracy.

Snyder and Barzilay [8] describe an algorithm that breaks up reviews into multiple aspects and then provides different numerical scores for each aspect. This would be helpful for mixed reviews that explicitly describe those aspects which are good or bad. For example, a movie reviewer may like a movie's acting and special effects, but find its plot poorly conceived.

Ensemble Learning

Ensemble learning techniques have been shown to increase machine learning accuracy by combining arrays of specialized learners. These specialized learners are trained as separate classifiers using various subsets of the training data and then combined to form a network of learners that has a higher accuracy than any single component.

Ensemble techniques increase classification accuracy with the trade-off of increasing computation time. Training a large number of learners can be time-consuming, especially when the dimensionality of the training data is high. Ensemble approaches are best suited to domains where computational complexity is relatively unimportant or where the highest possible classification accuracy is desired.

A number of different approaches to building ensemble learners have been proposed. There are

numerous ways to build ensemble systems and there are several decisions to be made which affect the performance of the final model [9]:

- How are subsets of the training data chosen for each individual learner? Training subsets can be chosen by random selection, by examining which training patterns are difficult to classify and focusing on those, or by other means.
- How are classifications made by the different individual learners combined to form the final prediction? They can be combined by averaging, majority vote, weighted majority vote, etc.
- What types of learners are used to form the ensemble? Do all the learners use the same basic learning mechanism or are there differences? If the learners use the same learning algorithm, then do they use the same initialization parameters?

One of the first ensemble machine learning techniques was bootstrap aggregating, also called bagging [3]. Bagging requires the construction of a number of classifiers using randomly chosen subsamples of the training data. By choosing different random subsamples for each classifier, some input patterns are repeated and some are not chosen at all. This allows each particular classifier the ability to focus on its training subset and the resulting ensemble is less likely to suffer from being stuck in a local optimum.

[10] describes another ensemble technique called the random subspace method. Instead of choosing subsets of training patterns like in bagging, the algorithm randomly chooses subsets of training features.

Another popular ensemble technique is boosting [4], which also has many variants [11], [12]. Boosting is an iterative process where each successive classifier's training subset is chosen based on the performance of the previously trained classifier. If the previous classifier had difficulty properly classifying a particular training pattern, then that pattern is more likely to be chosen to be included in the current classifier's training set. This allows the system to build learners which focus on those difficult training patterns. This method forces each learner to act as a specialist for classifying its particular region of the data space.

III. EXPERIMENTAL SETUP

In order to gauge the performance of ensemble techniques in the domain of sentiment mining, we set up classification accuracy tests to compare ensembles against standard, single machine learning models. If ensemble techniques are useful in this domain, then we would expect a higher level of classification accuracy. If classification accuracy does not increase, or only increases a negligible amount, then the added complexity and computational overhead of using an ensemble of classifiers would outweigh the benefit.

Datasets

We chose to perform our tests on several datasets that we collected and one used by Snyder and Barzilay [8]. Their dataset consists of restaurant reviews written by many different users along with labels denoting whether each review had positive or negative sentiment. The full dataset used by Snyder and Barzilay has 3488 reviews, but has a preponderance of positive reviews, so for this binary classification task we chose to use a subset of 1476 reviews containing exactly half positive reviews and half negative reviews. The full dataset also has separate ratings for the various aspects of the restaurant being reviewed (e.g. different ratings for service and food quality), but we only used each restaurant's overall rating. We also tested with four other datasets that we collected from Amazon.com, lawyerratingz.com, and tvratingz.com. These datasets are available online (http://www.cs.indiana.edu/~mewhiteh/html/opinion_mining.html).

Review Text	Sentiment
This is a truly, truly great restaurant...	Positive
I found the CD boring to listen to.	Negative
This laptop is cheap, but slow.	Negative

Table 1: Example raw training patterns

Since the datasets consist of natural language reviews (see Table 1), they needed to be converted to numeric vector representations before any machine learning algorithm would be able to use them to learn. We chose to use a unigram (bag-of-words) conversion. Using this kind of conversion, each machine learning input vector has one input value per word in the lexicon. We created the lexicon by including every unique word seen in any review and then removing stopwords and words that occurred only once.

To reduce the large size of the lexicons, we used the odds ratio method to select those words which were likely to help in classification. For each term in the full lexicon, the following odds ratio value was calculated:

$$\text{odds ratio} = \frac{p(1-q)}{q(1-p)}$$

where p was the fraction of positive reviews containing the target word and q was the fraction of negative reviews containing the target word. We then selected the words which had the highest absolute value odds ratios, a good measure of effect size for binary classification, under the assumption that those words would be most useful for distinguishing positive reviews from negative reviews. This allowed us to produce smaller lexicons that were only about 10% the size of the originals.

Each value in an input vector is the count of occurrences of a particular word in a certain review mapped to fit the range [-1.0, 1.0]. The maximum count of a word in any single review over all reviews maps to the value 1.0 and the minimum count to the value -1.0. Note that many of the values in an input vector are -1.0 since there are typically many words in each lexicon that do not occur in a particular review. These vectors of word occurrence counts coupled with converted class labels form the final training dataset.

Procedure

First, we ran a single support vector machine (SVM) learning algorithm on the full training datasets. To model the support vector machines, we used the libsvm library [13]. The results from these tests act as a baseline to compare with the results from our ensemble tests.

Next, we wrote our own implementations of several popular ensemble algorithms: bagging, boosting, random subspace, and bagging random subspaces. Each of these ensemble methods used SVMs for their base models.

For the bagging ensemble models, we trained groups of 50 individual models each with different training subsets. Individual model classifications from all models in the ensemble were combined by averaging to produce the final result for each test pattern.

We implemented the AdaBoost boosting algorithm [4] since it has been shown to have a high level of performance for many different problems. We set the number of iterations, T, to be 50, but also used the boosting variation of ending the process early given a high enough error from the last trained model.

For all the tests, 50 models were used for both the random subspace ensembles and the bagging random subspace ensembles. In the subspace tests, each model was given a randomly chosen subset of pattern features. In the bagging random subspace ensemble, each model was given a subset of training patterns and each pattern had a subset of training features.

Algorithm	camera	laptop	lawyer	restaurant	tv
Single Model	83.40	88.94	83.11	83.65	82.53
Bagging	84.69	89.11	83.45	85.41	82.98
Boosting	82.23	86.66	85.04	81.49	81.74
Subspace	85.80	90.00	84.56	85.95	84.26
Bagging Subspace	86.80	90.00	84.36	86.62	84.08

Table 2: Classification accuracies (% correctly classified)

To compare classification accuracy, we used K-fold cross validation with 10 folds to provide statistically stable accuracy estimates. This was done by first splitting up the dataset into 10 randomly chosen subsets. Then 10 trials were performed with each trial being a test on a single subset using the other 9 subsets as training data. The results of all 10 trials were then averaged to produce the final classification accuracies as reported.

IV. RESULTS

Table 2 shows the accuracy of the various setups. In general, the ensemble methods performed well and often outperformed the single model SVM.

The bagging ensemble was consistently equal to or better than the single SVM model across all the datasets. This is an encouraging result since this means a bagging ensemble can be used with the reasonable assumption that it will not hurt performance on sentiment mining datasets. It did not seem to have problems overfitting and losing generalization capabilities.

The random subspace and bagging random subspaces ensembles did even better than the bagging ensembles, often achieving the best overall accuracies of any of the methods. These techniques that use subsets of available features seem to be quite effective on the tested sparse datasets containing a large number of available features.

The boosting ensembles did not perform as well as the other ensemble methods with the exception of its performance on the lawyer dataset. The sparsity and noisy structure of the datasets appears to have caused the boosting models to overfit and degrade generalization performance.

When deciding upon which model to use for a real-world application, one should consider the accuracy requirements and time constraints. If time and computational resources are not an issue or the highest possible classification accuracy is desired, then the bagging subspace model seems to be the best choice.

V. CONCLUSION

We have shown that ensemble learning techniques can increase classification accuracy in the domain of sentiment mining, though the choice of ensemble method affects accuracy. The random subspace and bagging subspaces models typically had the highest classification accuracies. If one wants to create a model with the highest possible classification accuracy for sentiment mining, then ensemble methods should be considered.

The only drawback of these methods is that they increase the computational time required for training and classification. For all the ensemble methods, a group of many different learners must be trained as opposed to a single learner that is used to make all classifications. This makes the training time of ensemble techniques roughly $S \cdot t$, where S is the size of the ensemble and t is the time it takes to train a single learner. This time increase may be prohibitive in cases where t is already very large, so a computation time vs. accuracy decision would have to be made. However, the bulk of the computational expense is a one-time occurrence, applying only to the learning phase, and ensemble learners can be sufficiently fast at classification that the up-front expense is well justified for the increased

accuracy in actual use.

In the future we intend to empirically test other ensemble models, including ways to increase computational efficiency while still maintaining the benefits of using a combined group of classifiers. It would also be interesting to investigate different ways of representing the user reviews (e.g. using other n-gram models, or building a lexicon that consists entirely of polarity words) so that the various machine learning algorithms would have less noise with which to contend.

REFERENCES

[1] Lyman, P., and Varian, H. 2000. How much information. http://www.sims.berkeley.edu/how-much-info.

[2] Lyman, P., and Varian, H. 2003. How much information 2003. http://www.sims.berkeley.edu/how-much-info-2003.

[3] Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2):123–140.

[4] Schapire, R. E. 2002. The boosting approach to machine learning: An overview.

[5] Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, 271–278.

[6] Pang, B.; Lee, L.; and Vaithyanathan, S. 2002. Thumbs up? sentiment classification using machine learning techniques. *CoRR* cs.CL/0205070.

[7] Whitelaw, C.; Garg, N.; and Argamon, S. 2005. Using appraisal groups for sentiment analysis. In Herzog, O.; Schek, H.-J.; Fuhr, N.; Chowdhury, A.; and Teiken, W., eds., *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, 625–631. ACM.

[8] Snyder, B., and Barzilay, R. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of NAACL HLT*, 300–307.

[9] Polikar, R. 2006. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine* Third Quarter:21–45.

[10] Ho, Tin Kam. 1998. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8): 832-844.

[11] Demiriz, A., and Bennett, K. P. 2001. Linear programming boosting via column generation.

[12] Domingo, and Watanabe. 2000. Madaboost: A modification of adaboost. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

[13] Chang, C. C., and Lin, C. J. 2001. *LIBSVM: a library for support vector machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm.