Text Mining to Detect Insults in Online Discussion

Ryan Foley
rfoley@stanford.edu

Ben Kasper
bkasper@stanford.edu

Robert MacNguyen
rmacngu@stanford.edu

December 14, 2012

Abstract

The anonymity of online communication makes it particularly prone to hostility. Given text from online discussions [9], we train a variety of supervised learning algorithms to classify insults directed at other forum members. After meticulous text processing to generate features, we fit naive Bayes and regularized logistic regression to establish a baseline. Then we attempt to maximize accuracy with more complex models: random forests, SVM, and boosted regression trees. In order to avoid overfitting, we emphasize thorough model tuning and evaluation via 10-fold cross validation, while maintaining a separate test set throughout. We evaluate models primarily via ROC curves and AUC, paying special attention to performance when strictly constraining false positive rate.

1 Preprocessing

Before applying learning algorithms, we must generate meaningful features from the text. We begin with 3,947 strings, where 1,049 are labeled as insults. To clean the data we tokenized, converted to lowercase, stemmed (using a Porter Stemmer), and removed one letter words, numbers, and symbols. Primarily, we count the number of occurrences of each tokenized value. Then we include bigrams and trigrams to capture local context. We retain only those features that occur at least twice, leaving us with 25,671 total. With so many more features than observations, we recognize the need to regularize and employ methods that are resistant to the inclusion of irrelevant features. Finally, we split the data into a training and test set at random, with 80% of the observations belonging to the former.

2 Methodology

In this section we describe the model fitting and tuning process. Our general strategy is to tune each model on the training set with respect to some performance criterion (usually some measure of generalization error), while keeping the test set hidden.

2.1 Naive Bayes

We train a naive Bayes multinomial event model, which assumes conditional independence among the features. This is a canonical text classification model, and we find it has impressive performance for its simplicity, achieving a test set AUC of .8630. However, we find that models not relying on strong distributional assumptions will be able to improve upon this result.

2.2 Elastic Net Regularized Logistic Regression

We apply elastic net penalized logistic regression. This adds to logistic regression the penalty $P_{\alpha,\lambda}(\beta) = \lambda\left[(1-\alpha)\frac{1}{2}||\beta||_{\ell_2}^2 + \alpha||\beta||_{\ell_1}\right]$, where β is the vector of coefficients. Note that $\alpha = 0$ corresponds to a pure ridge penalty, and $\alpha = 1$ to a pure lasso penalty. Thus, for $\alpha \in (0, 1)$ the elastic net can be seen as a compromise between ridge and lasso. It has been shown to outperform the lasso (especially with correlated variables), while maintaining its desirable sparsity properties [8]. In particular, the elastic net excels when the number of variables exceeds the number of observations.

For a given α, the parameter λ parameterizes a path of solutions, where smaller λ corresponds to a harsher penalty. We compute these solutions using pathwise coordinate descent as described in [3], and perform a grid search maximizing area under the ROC curve (AUC) to pick optimal values of α and λ.

For each choice of α, we pick the optimal λ by the 'one standard error' rule; that is, the largest λ that is within one standard deviation of λ_{opt}. After

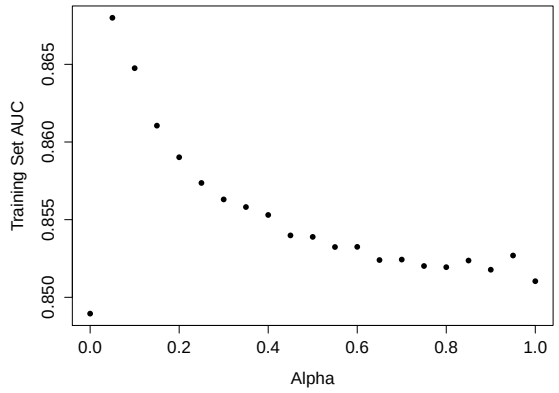

Figure 1: Logistic Regression: Grid search to optimize AUC over elastic net penalty α.

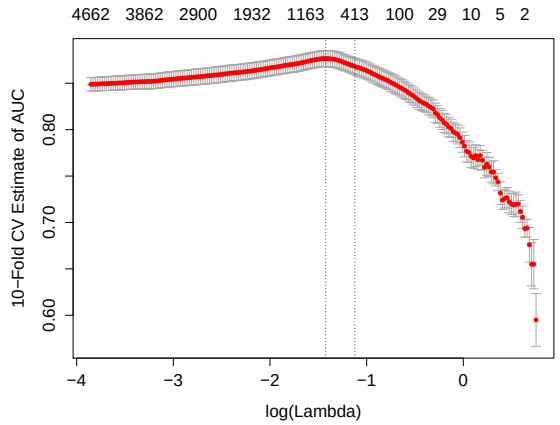

Figure 2: Logistic Regression: Grid search to optimize 10-fold CV estimate of AUC over λ (when $\alpha = 0.05$)

picking the best λ for each α, the α with the best training set AUC is chosen. The grid search yeilds $\alpha^* = 0.05, \lambda^* = 0.01827903$, and a final model with just 413 features. With a test set AUC of 0.8613, this is the worst performing of our models. We suspect that even with regularization, logistic regression suffers from the high-dimensional feature space.

2.3 Random Forests

We now pursue more complex models in an attempt to maximize performance. The first is a random forest, in which decision trees are bagged to reduce the model's variance. Each tree is constructed by picking m features at random at each node amongst which a split is chosen. The out-of-bag (OOB) error of a random forest is an unbiased estimate of the generalization error and is guaranteed to converge as more and more base learners are constructed [1]. Thus, we need only train the random forest until OOB error converges. The only parameter to be tuned, then, is m. We perform a grid search minimizing OOB error, and find $m^* = 32$. The final random forest has 0.8713 test set AUC, a noticeable improvement over the simpler models.

2.4 Support Vector Machine

We now build a support vector machine (SVM) classifier. We use a radial basis kernel and show that, on our data, it outperforms SVMs with several other kernels.

First, we use a radial basis kernel and perform 10-fold cross validation on the training set. Performing a $\log_2$-scale grid search across parameter values shown in Figure 3, we find that the minimum misclassication error on the test set is 0.1244058 with the soft margin cost parameter of 2 and the radial basis kernel parameter $\gamma = .0078125 = 2^{-7}$.

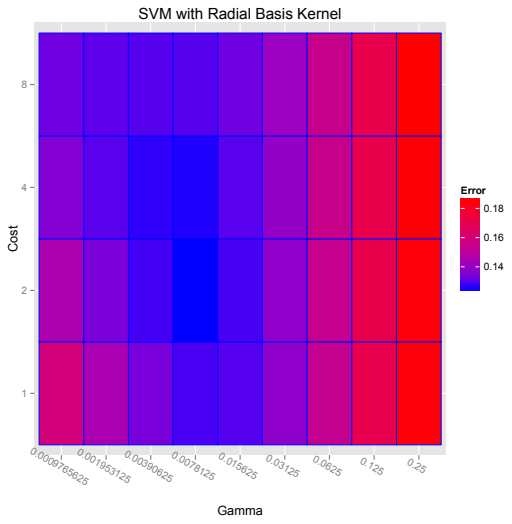

Figure 3: SVM Tuning: Error as a function of the parameter gamma and soft margin cost for the radial basis SVM.

Next, for comparison we replace the radial basis kernel with a linear kernel. Performing 10-fold cross validation again and varying the value of the cost parameter, we have the resulting misclassication error rates shown in Figure 4.

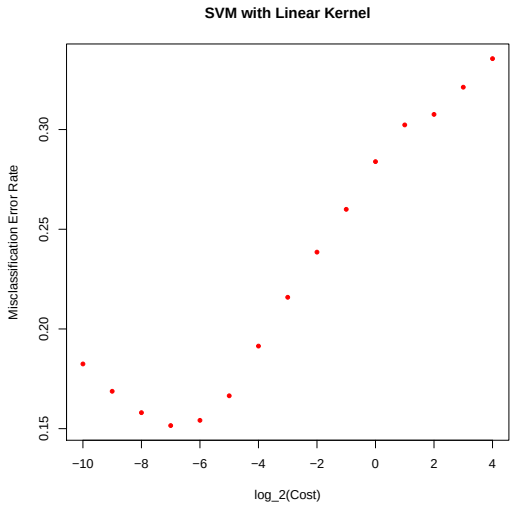

Figure 4: SVM Tuning: Error as a function of soft margin cost parameter for linear kernel SVM.

Kernel	AUC
Polynomial	0.7770
Linear	0.8011
Sigmoid	0.8332
Radial	0.8800

Table 1: SVM results with various kernel choices

In Figure 4, we see that the linear kernel has a minimum misclassication error of 0.1526587 with the cost parameter set to 2^{-7}.

With this tuning procedure repeated, Table 1 shows the AUC of the test set predictions from the SVM with several different kernels.

From the table we see that the radial basis kernel significantly outperforms the others. The radial basis function, $K(u, v) = \exp\left(-\gamma\|u - v\|^2\right)$, for the SVM creates a decision boundary from weighted Gaussians at the support vectors. Furthermore, it allows for more flexibility in the decision boundary as it can be nonlinear and more spherical at particular training points.

2.5 Boosted Regression Trees

We implement boosted regression trees using the Adaboost exponential loss function. However, in contrast to traditional AdaBoost, the optimization is carried out through Friedman's stochastic gradient boosting machine [4]. Most notably, this means that at each iteration we randomly sample half of the training set (without replacement) on which to fit the next base learner. This combined with a small shrinkage coefficient reduces the variance of the ensemble in much the same way as bagging. We find empirically that setting the shrinkage coefficient as small as possible results in the best performance, though this comes at the cost of more iterations. Our best run uses a shrinkage coefficient of 0.005.

We use regression trees as base learners, despite the fact that this is a classification problem. This means that predictions are unbounded real numbers, but are thresholded to generate predictions. Our tuning process (again by grid search minimizing 10-fold cross validation error) focuses on the complexity of these base learners. There are two parameters at our disposal: the maximum interaction depth each tree is allowed to have (p), and the minimum number of observations required in a node to allow continued splitting (n). We execute a two-dimensional grid search over $p = \{1, \ldots, 5\}$ and $n = \{2, 4, 6, 7, 8, 9, 10, 12, 15, 25\}$, for each run tuning the number of iterations by 10-fold cross validation. We find the best parameter pair to be $p = 5$ and $n = 7$.

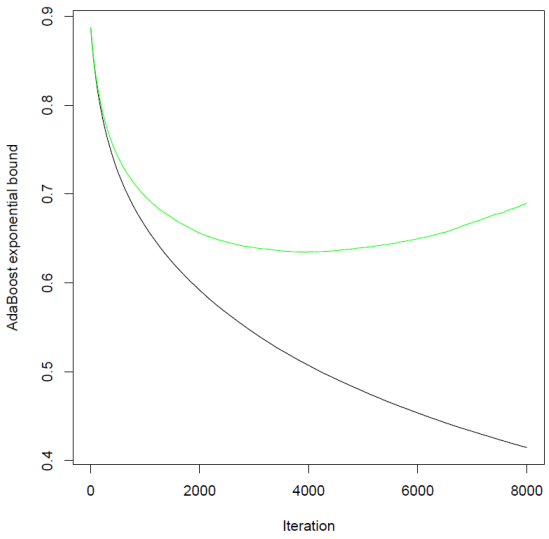

Figure 5: Boosting: Tuning model complexity. Training set error (black) and 10-fold CV error (green).

Following tuning, this model has a test set AUC of 0.8950, beating our previous models by a substantial margin. We attribute the success of this model in part to the well-known robustness of tree-based methods to irrelevant features. Furthermore, we find it enlightening that an interaction depth of 5 produced the best model. Our experience is that including high order interactions tends to overfit, but here

the prediction accuracy benefits under 10-fold cross validation and generalizes to the test set. We suspect that this is because our features are word counts, so the high-order interactions may be capturing the underlying complexity of language.

2.6 Ensemble

As one final attempt to maximize accuracy, we construct an ensemble of the previous models. To do this, we fit ridge regression using as features the training set predictions of each model. We choose a ridge penalty because we expect the predictions to exhibit multicollinearity. On the test set, our ensemble achieves an AUC of 0.8970.

3 Evaluation

We evaluate each model on the randomly chosen test set consisting of the unseen 20% of the original data. When evaluating the trained models, we wish to observe the tradeoff between true positive and false positive rates when classifying on the test set. To do so, we plot their receiver operating characteristic (ROC) curves. To summarize performance, we calculate the area under the ROC curve (AUC), a common measure of classifier performance.

Figure 6 shows the ROC curve corresponding to each model. Note in particular that boosting and the ensemble appear to outperform the other models. This result is corroborated by the summary of AUC values in Table 2, where we see that boosting and the ensemble are the top two performers by this metric.

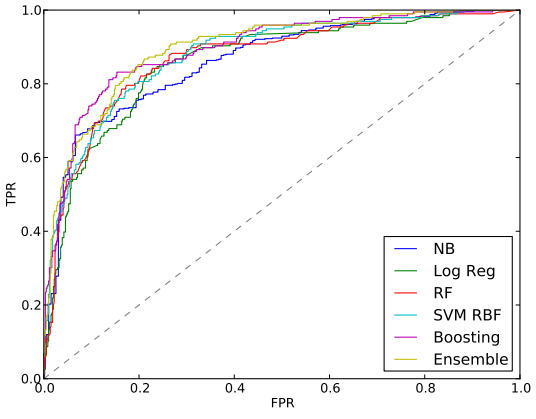

Figure 6: ROC curves for each model.

Model	AUC
Log. Reg.	0.8613
Naive Bayes	0.8630
Random Forest	0.8713
SVM	0.8800
Boosting	0.8950
Ensemble	0.8970

Table 2: Comparison of AUC between models

However, practical applications often require constraining false positive rate. In the context of detecting insults on the web, a high false positive rate would result in a high rate of unwarranted censorship, potentially angering forum members and discouraging users from posting. Thus, we are more concerned with each model's performance whe false positve rate is low.

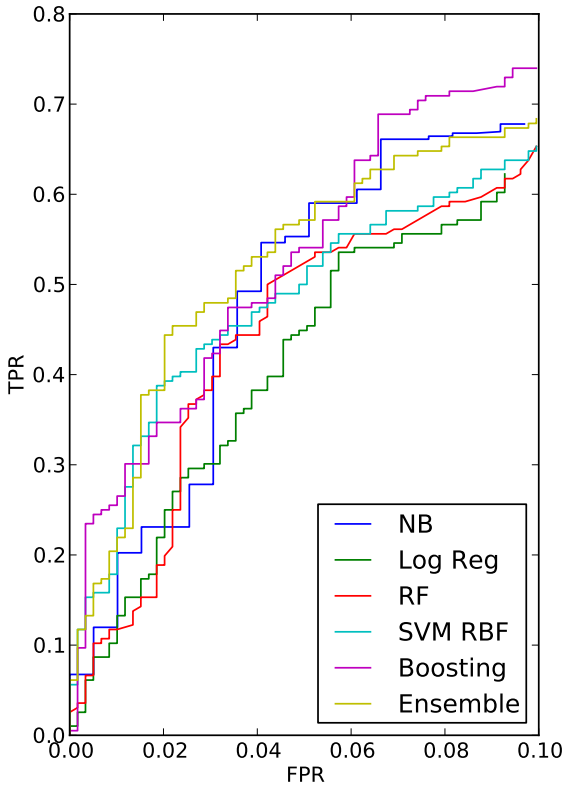

Figure 7: Zoomed in view of ROC curves for each model.

Figure 7 displays the behavior of the ROC curves at low false positive rates. We see a particularly en-

couraging curve for boosting, which correctly identifies almost 25% of the insults while making very few mistakes. Table 3 provides the true positive rates for each model at false positive rates of 0.1 and 0.01. Although the ensemble rises to the top for much of the plot in Figure 7, we observe that boosting has the highest true positive rate at each cutoff. We conclude that boosting and the ensemble are the top performers for our objective of detecting directed insults in online discussions, even when restricted to low false positive rate.

Model	TPR @ FPR = 0.1	TPR @ FPR = 0.01
Log. Reg.	0.6276	0.1020
Naive Bayes	0.6779	0.1197
Random Forest	0.6531	0.1173
SVM	0.6531	0.1786
Boosting	0.7398	0.2551
Ensemble	0.6837	0.2041

Table 3: True positive rates for each model at constrained values of false positive rate.

4 Further Work

We tried several other methods of extracting features from the data, but had poor or insignificant results. We tried removing stopwords (but leaving pronouns, since these are likely to be present in directed insults), but doing so dramatically decreased the performance of our models. This may have removed some meaningful context of the comment. Additionally, we found that including word dependencies - either as a substitute for bigrams or as individual features of two words and their dependency - made a very slight but negligible performance gain. We tried synonym set counts and squared counts (so as to remove any linear dependence with word counts) but did not see a performance gain. Lastly, we varied the degree of n-grams and found 3 to have the best performance.

We think the most fruitful area for future work is in gathering new types of features. For example, we could get user-based features such as location, post history, and past comments. This might allow us to construct a profile of each user, to classify users themselves as toxic. Additionally, it would be useful to follow a conversation in its entirety to look at characteristics like thread length, number of replies to a comment, and nested comments.

Regardless, we have achieved promising results. Our detection rate, even when enforcing a strict limit on false positive rate, is encouraging. Our results gives hope for automated moderation of online discussion.

References

[1] Breiman, Leo (2001). Random Forests. *Machine Learning* 45(1):5-32.

[2] Forman, George (2003). *An extensive empirical study of feature selection metrics for text classification.* The Journal of Machine Learning Research 3:1289-1305.

[3] Friedman J.H., Hastie, T, and Tibshirani, R (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1-22.

[4] Friedman, J.H. (2002). Stochastic Gradient Boosting. *Computational Statistics and Data Analysis*, 38(4):367-378.

[5] Nadig, Raghuvar, J. Ramanand, and Pushpak Bhattacharyya (2008). *Automatic Evaluation of Wordnet Synonyms and Hyper Nyms.* Proceedings of ICON-2008: 6th International Conference on Natural Language Processing.

[6] Tong, Simon, and Koller, Daphne (2002). *Support Vector Machine Active Learning with Applications to Text Classification.* Journal of Machine Learning Research. 2:45-66

[7] Wang, Sida, and Manning, Christopher D. (2012). *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification.* ACL '12 Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers. 2:90-94.

[8] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *J. Royal. Stat. Soc. B.*, 67(2):301-320.

[9] http://www.kaggle.com/c/detecting-insults-in-social-commentary