CIS 520: Final Report
Team OnlyLinearModels

Joseph K. Aicher
jaicher@upenn.edu

Lei Shi
shilei@upenn.edu

Jiachen Wang
jiacwang@upenn.edu

11 December 2017

1 Introduction

For this project, we were asked to develop three different models to classify the mood of each tweet in a given set of tweets as either `joy`, `sadness`, `surprise`, `anger`, or `fear`, and to develop one specifically so that we would minimize a weighted loss function as defined in table 1. The three different models were required to be of different types. Specifically, we were asked to develop a generative method, a discriminative method, and an instance-based method, which we implemented using Naïve Bayes, logistic regression, and *k*-nearest neighbors, respectively. We ended up refining the logistic regression method to minimize the weighted loss function, borrowing ideas from Naïve Bayes, leading to our final model that achieved first place in the in-class competition.

	joy	sadness	surprise	anger	fear
joy	0	3	1	2	3
sadness	4	0	2	3	2
surprise	1	2	0	2	1
anger	2	1	2	0	2
fear	2	2	2	1	0

Table 1: The cost matrix defining our problem's weighted loss function. The first column represents ground truth, and the first row represents the predicted labels.

In this report, we first describe the three final models we submitted. We then describe the process with which we developed and evaluated candidate models, leading to our final model. We conclude with a brief discussion of our performance in the contest and possible future directions for improved performance.

2 Description of Final Models

We implemented the following models for our `proj_final` submission:

- A generative method: Naïve Bayes
- A discriminative method: Logistic Regression
- An instance-based method: *k*-Nearest Neighbors

The Naïve Bayes model we submitted was extraordinarily simple and was not tuned as previously discussed in the literature to perform better in the context of document/sentiment classification[1]. We simply used a multinomial model with counts truncated to 1 with a pseudocount of 1 (e.g. Laplace smoothing). The resulting probability estimates were used to minimize expected cost. This was the first model we submitted to the leaderboard, where its performance on the validation set was 0.9464 (just under the first baseline). For more details, please see the previously submitted README.

The logistic regression model was inspired by the support vector machine (SVM) with Naïve Bayes (NB) features model in [2]. We used ridge-penalized logistic regression instead of SVM to directly make probability estimates and generalized to the multiclass setting by taking a one-vs-all-but-one approach. This can be seen as using Naïve Bayes to determine an appropriate scale for the ridge penalty on the different features. Input features to this model were aggregate, stemmed words obtained using the `SnowballStemmer` as implemented in the Natural Language Toolkit for Python[3].This was the third model we submitted to the leaderboard, where its performance on the validation set was 0.9052. We subsequently submitted it as our final model, where its performance on the test set was 0.92556, which came first place in the competition. For more details, please see the previously submitted README.

The k-nearest neighbors model was used to provide an instance-based probability model. Features were initially transformed using PCA to reduce the dimensionality of the Euclidean distance calculations. Probability estimates were estimated using relative proportions of each label and used to predict the label with smallest expected cost. Generating predictions from this model was prohibitively slow, and performance as estimated by 10-fold cross-validation was 1.211, so we did not submit it to the leaderboard to evaluate its performance on the validation set. For more details, please see the previously submitted README.

3 Development and Evaluation of Candidate Models

We iterated through numerous candidate models as a team which would be too numerous to fully enumerate in this report. However, many of these models represent the same pattern of trying various preprocessing steps on some single class of model to find a pattern that performed better. In this section, we describe the general approach we took in the development and evaluation of candidate models. In particular, we will first describe how we evaluated our models using cross-validation. We will then describe different feature transformations we tried. Finally, we will describe our search through different algorithms leading to our leaderboard submissions, providing our estimates of performance as evaluated by cross-validation, and, when available, performance on the validation and final test sets.

All candidate models were evaluated using 10-fold cross-validation. With the exception of the k-nearest neighbors, these methods were all evaluated using the same set of folds as generated by the built in MATLAB function `crossvalind`. Performance on each fold was evaluated using the provided `performance_measure.m` function and summarized for the folds by taking the mean and standard deviation of the ten resulting scores. For this report, our results/performance estimates are from this procedure using folds generated by `crossvalind('Kfold', 18092, 10)` with the random seed `rng(42)` except where otherwise mentioned (i.e. k-nearest neighbors).

In the development and evaluation of candidate models, we considered various feature transformations as described in table 2. We arbitrarily chose the `SnowballStemmer` over alternatives in the NLTK because it appeared to be the most popular stemmer and did not compare its performance to other stemmers because of the inconvenience of doing stemming directly in MATLAB. Meanwhile, we chose pseudocount and number of components parameters in these preprocessing steps using cross-validation.

We were largely unsuccessful before the deadline for baseline 1 but began learning important lessons on feature transformation. We initially tried using multi-class logistic regression as implemented in `liblinear` on original features to predict probabilities and then cost-sensitive hard-decisions on the labels. Coarse search over cost parameters $C \in \{0.1, 0.5, 1, 5\}$ identified $C = 0.5$ as least poorly performing (Note that cost parameter C is inversely proportional to λ in usual formulation of ridge penalty). Since it was easy to try SVM using the same code, despite the inability to directly compute probabilities, we also trained a linear SVM on the untransformed data using the same cost parameters, using the maximum score as our decision rule. As we expected due to the lack of an easy way to integrate cost-sensitivity into training or prediction, this underperformed the basic logistic regression model and we proceeded to ignore SVM as a model for the rest of our search. We then tried logistic regression with elastic net penalty, using `lassoglm` in MATLAB to pick λ for a given α. We had similar performance on the original features, which improved with transformation of features by `tf-idf` followed by SVD. Chosen value for α approached a ridge penalty, suggesting that the feature transformation was responsible for improvement. The performance was still substantially higher than the first baseline, and with increased training time and an approaching deadline, we abandoned further investigation of this model at that time. We briefly attempted a random forests classifier on the same transformed features, using the `TreeBagger` implementation in MATLAB with cost-sensitive predictions. Unfortunately, the model was slow to train and predict, and it did

Transformation	Hyperparameter(s)	Description
Intercept	None	Add column of ones as a feature.
Number of words	None	Add column corresponding to the number of words used in a document.
SVD/PCA	Number of components *k*	Use principal components of low-rank approximation of original features matrix with respect to Frobenius norm.
Stemming	Algorithm choice (i.e. `SnowballStemmer`)	Summed aggregation of similar words based on similar word stems[4].
`tf-idf`	Pseudocount	Scale features in some proportionate way to the inverse of the frequency in which they appear to weigh less commonly occurring features more highly[5].
NB features	Pseudocount	Use binary labels to learn reweighing of features using Naive Bayes log probability ratios as described in [2].

Table 2: List of feature transformations we considered, along with hyperparameters/user choices to be chosen either arbitrarily or by cross-validation and a brief description of what the transformation is.

not show improvement over the previous classifier. At this point, we also tried *k*-nearest neighbors, which ended up being the instance-bsaed classifier we submitted. Cross-validation over grid-search on the number of neighbors and the number of principal components did not yield a satisfactory estimate of performance. At this point we missed the first baseline's deadline, and we waited a few hours to rest and ask the instructors for suggestions. These first candidate models were largely unsuccessful. What worked to improve performance in these models was transforming the data. Due to time constraint related to the surrounding holiday weekend, we did not fully troubleshoot these models before moving on.

Conversation with Dr. Shivani Agarwal pointed out that we had not attempted any generative models. She suggested to try Naïve Bayes, which we implemented as discussed in the previous section. Performance on cross-validation with default Laplace smoothing was sufficient to warrant submission to the leaderboard, which allowed us to pass the first baseline 14 hours after the deadline elapsed. Subsequent attempts to improve on this model's performance relied on more feature transformation and bootstrap aggregation. Using a stemmed vocabulary, we applied bootstrap aggregation to the original NB classifier, searching over the optimal proportion of samples and features to use in the context of 100 predictors. Performance was sufficiently improved on cross-validation to justify submitting to leaderboard; unfortunately, performance on the validation set was worse. At this point, we returned to logistic regression, this time using NB features as described in the previous section. With initial parameters, the cross-validation estimate easily outperformed the previous models. Search was performed on the cost parameter to optimize the cross-validation estimate, leading to our final model that allowed us to beat baseline 2 and subsequently place first in the competition.

4 Discussion and Future Directions

We performed a combinatorial search over different candidate algorithms to perform cost-sensitive multiclass classification of tweets as outlined in table 1. We ultimately started and ended our search with the same fundamental algorithm: (ridge-penalized) logistic regression. The performance difference between those first and last times is stark, demonstrating the relevance of using appropriate features as alluded to in table 2. Our least successful algorithms were scale-sensitive methods where we either did not transform the *scale* of the data or used only SVD, which preserves Euclidean distance (approximately). We hypothesize that these features were ineffective due to the sparsity of our data and the comparable number of features to observations (e.g. $p \sim n$). Transforming the data with `tf-idf` showed improvements with these methods, although we did not pursue that further due to time constraints. The most successful algorithms we tried involved the use of Naïve Bayes (table 3). The multinomial model with truncated counts that we used, either directly or to transform our data, is scale-invariant.

Candidate model	Transform	Hyperparameters	Cross-validation estimate	Validation set performance
Ridge-penalized logistic	None	$C = 0.5$	1.332 ± 0.023	
SVM	None	$C = 0.5$	1.339 ± 0.022	
Elastic-net logistic	`tf-idf-SVD`	$k = 50$, $\alpha = 0.05$	1.085 ± 0.020	
k-nearest neighbors	SVD	$k_{\text{SVD}} = 20$, $k_{\text{NN}} = 30$	1.211	
Naïve Bayes	None	`pseudocount` $=$ 1	0.9533 ± 0.0334	0.9464
Bagged Naïve Bayes	Stemmed	`pseudocount` $=$ 1, $p_{\text{sample}} = 1$, $p_{\text{feature}} = 0.8$	0.948 ± 0.033	0.9494
Ridge-penalized logistic	Stemmed-NB-features	`pseudocount` $=$ 1, $C = 0.3$	0.909 ± 0.033	0.9052

Table 3: List of candidate models in chronological order with performance estimated by 10-fold cross validation (± 1 standard deviation). Performance on validation set provided for models submitted to leaderboard.

As discussed before in class for the general case, we suspect that its unreasonable effectiveness in this setting despite the clearly erroneous and biased assumptions because of the relatively low variance of estimation in this high-dimensional setting.

We were happily surprised by our performance on the final test set. We were content with the performance of our model and felt that its simplicity as a generalized linear model would serve well in generalizing to the final test set. Nonetheless, as we fell in the leaderboard, our optimism waned. In hindsight, we suspect that other teams, optimizing until the very last minute and assisted by the decrease in the time-delay between submissions from 3 hours to 1.5 hours, ended up overfitting their model to the validation set.

Ultimately, there were several avenues for further investigation towards improving the model that we left unexplored. Before the deadline for the first baseline, we obtained improvements in performance with non-Naïve Bayes transformations, but we did not pursue them because of the time constraint and the later pivot to Naïve Bayes derived or inspired methods. More careful analysis could suggest a role for these feature transformations in augmenting and improving our existing models. Conversely, we could potentially extend the use of the NB-features towards making standard distances between observations more meaningful, increasing the utility of distance-based models (i.e. k-nearest neighbors, k-means, etc.). One potential application would be segmenting the data using clustering and training models on each of the clusters. Additionally, since we were given the validation features but not their labels, we could have further explored if a semi-supervised approach could have improved our estimates. Lastly, we would also be interested in removing our heavy reliance on models that by definition perform some kind of probability estimation; using Platt scaling or isotonic regression as suggested in [6] would allow us to use a wider range of algorithms without needing to modify the approach for training.

References

[1] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 616–623.

[2] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ser. ACL '12, Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 90–94. [Online]. Available: `http://dl.acm.org/citation.cfm?id=2390665.2390688` (visited on 12/12/2017).

[3] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

[4] *Stemming*, in *Wikipedia*, Page Version ID: 814128251, Dec. 7, 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Stemming&oldid=814128251` (visited on 12/12/2017).

[5] *Tf–idf*, in *Wikipedia*, Page Version ID: 813026023, Dec. 1, 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=813026023` (visited on 12/12/2017).

[6] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning*, ACM, 2005, pp. 625–632.