Multiclass Emotion Analysis of Social Media Posts

Alec Glassford and Berk Çoker
Stanford University

Abstract

We performed 15-class sentiment classification on tweet-length social media posts. Preprocessing text and training models such as Multinomial Naive Bayes, Linear SVM and Logistic Regression, we were able to predict emotion with far greater accuracy than random chance. However, distinguishing between similar emotions (e.g. 'annoyed' and 'angry') still remains a challenge. Moreover, we suffered from high variance due to limited data, which was mitigated with only meager success by regularization and feature selection. Techniques like neural networks and word vectors remain promising.

Introduction

Sentiment analysis of text has been explored in many ways. Research on this task include simple classification by polarity (positive or negative, or along a spectrum like a 5-star scale), detecting political sentiment, and classifying emotions along a happy sad spectrum. For our project, we were curious to see how successfully we could implement multiclass sentiment analysis of text. Specifically, we wanted to classify social media posts into 15 fine-grained emotional buckets:

['lonely', 'depressed', 'sad', 'disappointed', 'worried', 'upset', 'stressed', 'angry', 'annoyed','excited', 'hopeful', 'optimistic', 'satisfied', 'thankful', 'thoughtful']

Social media is an emotion-rich realm, and this type of classification could be useful for a range of applications from advertising to suicide-risk detection.

Data

We acquired 225,000 emotion-labeled short text posts, evenly split among the 15 emotions above, from the social media website "Experience Project." Here are some examples of our raw data:

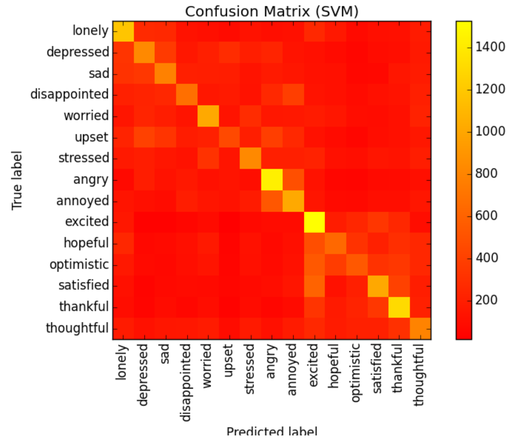
Text: 'Job interview today!! :D'
Emotion: excited

Text: 'Feeling better then yesterday but am lost'
Emotion: lonely

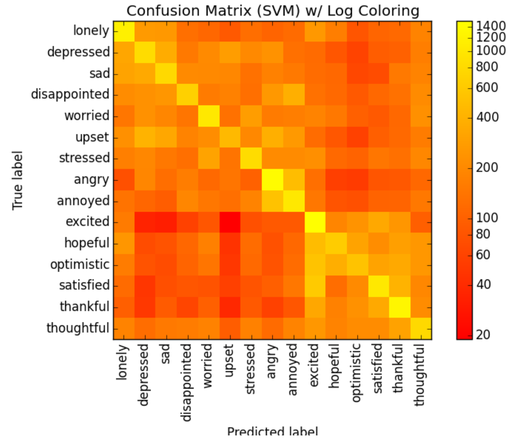
Preprocessing

- Filter out duplicate statuses, statuses with less than 10 characters
- Tokenize with NLTK's TweetTokenizer: Keep punctuation (significant to meaning of social media posts).
- Build vocabulary from training set and treat each document as a bag-of-words, counting unigrams and bigrams.
- Weigh terms according to tf-idf and normalize feature vectors.

Performance of SVM



Same as left, with logarithmic color scale



Estimation

We implemented several different types of estimators by utilizing the Scikit-learn Python library:

- Multinomial Naive Bayes
- One vs. rest Support Vector Machine with Linear Kernel (experimented with L1 vs L2 penalties, different regularization parameters, hinge vs squared hinge loss)
- Logistic regression (one vs. rest and multinomial)

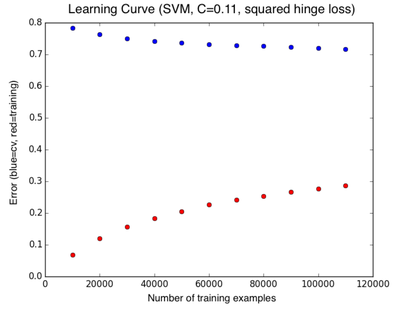
By testing these on a hold-out cross-validation set, we found their performance to be comparable, with some SVM formulations having slightly better accuracy.

Some of the estimators we tried

Estimator	Cross Validation Accuracy
Multinomial Naïve Bayes	0.2801
Naïve Bayes with English stop words removed	0.2679
Multinomial Naïve Bayes w/ trigrams added	0.2816
Logistic Regression (multinomial)	0.2795
Logistic Regression (one vs rest)	0.2797
SVM (linear kernel, hinge loss, C=1)	0.257
SVM w/ features reduced to 100,000	0.2454
SVM (linear kernel, squared hinge loss, C=1)	0.2597
SVM (linear kenrel, squared hinge loss, C=0.11)	0.2843

Over-fitting

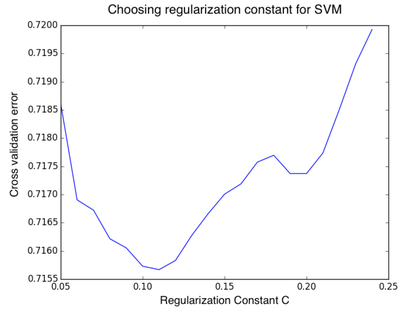
Unfortunately, our high dimensional feature vectors were prone to over-fitting, especially for SVM implementations which could achieve accuracy of around 80-90% on the training set, depending on hyper-parameters. Ultimately, acquiring more data could have improved our results. This learning curve for an SVM gives a sense of our high variance:



Regularization

We attempted to address over-fitting with several techniques

- Feature Selection: Removing stop words, only choosing words with the top SVM weights. These were largely ineffective (see above). Tf-idf already did a lot of this work.
- Regularization: For example, adjusting C for SVM with squared hinge loss, to some minor success.



Results

While there is much room for improvement, we can see a clear diagonal in the confusion matrices to the left. Retraining our cross validation-tuned SVM on the training set + cross validation set combined and then testing a completely withheld test set led to prediction with accuracy 0.2879. This far surpasses the expected accuracy of 0.0667for 15-class classification by random chance.

Future Work

We have two tasks remaining on our roadmap. Training our supervised LDA model, and our Convolutional Neural Network model. We currently can't present our results, as we still have to optimize our code to run faster. With the former, we intend to understand the similarities between the mood categories. Unlike the former models we have implemented, SLDA is generative. The latter model we've implemented makes use of word vectors and learns to classify sentences. As such is the case, CNN has the potential to capture more intricate semantic similarities/differences.

Acknowledgements

We thank Kanjoya Inc. for providing us with the sample of moods data and for being encouraging about our research.

And of course we'd like to thank the CS 229 teaching team, especially Professor Ng and our project mentor Sam Corbett-Davies.

References

[1] Jon D. McAuliffe David M. Blei.
Supervised topic models.
2007.

[2] Yoon Kim.
Convolutional neural networks for sentence classification.
2014.

[3] Andrew L. Maas Christopher Potts Moritz Sudhof, Andrés Gómez Emilsson.
Sentiment expression conditioned by affective transitions and social forces.
2014.

Contact Information

- Email: gla@stanford.edu
- Email: bcoker@stanford.edu