Multiclass Emotion Analysis of Social Media Posts

Alec Glassford and Berk Çoker
Stanford University
gla@stanford.edu, bcoker@stanford.edu

Abstract

We performed 15-class emotion classification on tweet-length social media posts. Our models were able to predict emotions with far greater accuracy than random chance. Among the models we built, the Linear SVM performed the best with 28.43% accuracy. Despite these results, however, distinguishing between similar emotions remains a challenge. The high variance in human text is hard to mitigate with model optimizations; and potential improvements with more data introduce scalability problems.

1 Introduction

Social media provides a corpus of text that is rich with emotion. For our project, we were curious to see how successfully we could implement multi-class sentiment analysis on human text. Specifically, we wanted to classify social media posts into 15 fine-grained buckets, as seen below, building a classifier that would output the mood that best describes the author's mindset in writing the text:

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

For this task, we built and optimized the following models: Multinomial Naive Bayes, Softmax Regression, Linear SVM, ν-SVM with RBF Kernel and Convolutional Neural Network. Our selection of models was guided by the literature review.

The ability to detect fine-grained emotion in social media posts has many potential applications; for instance, it could be used to build and maintain an emotional profile for a user to track psychological health and general well-being, or even detect suicide risk. In fact, for our CS221 project, we explored using the same data to build such emotional profiles with hidden Markov models, with the classifications from this project helping define the emission probabilities.

Moreover, our fine-grained emotional analysis could be applied to other types of short form text, like Amazon or Yelp reviews, which could provide specific feedback and insights to merchants. Finally, from a purely psychological standpoint, it is interesting to see if there is underlying structure to text that distinguishes similar emotions like 'lonely' and 'depressed.'

2 Related Work

Our review of prior work on supervised semantic analysis resulted in two important deductions: First, we concluded that the complexity of models and methods for this task saw a dramatic change over the past two decades. We observed that while the earlier works used linear models like Linear Regression and SVM's, the recent ones use non-linear vector space models like neural networks. In contrast, the earlier works seem to have incorporated more sophisticated preprocessing techniques and intricate systems, unlike the recent ones, which favor generalizable features like word vectors, and all-in-one model systems for scalable and flexible applications. These improved models also allowed for research on more challenging tasks, which brings us to the second deduction: the definition of the task has also partially changed over the past years. There was a shift in interest from the binary categorization of opinionated text to the multiclass one, which, for instance, brought Stanfords new Sentiment Treebank into existence. The cutting-edge research is tending toward classification with finer grained scales. As is the case, we see our project as a natural continuation of these prior works.

In their paper, Hatzivassiloglou and McKeown[2] focus on the task of predicting semantic orientation of adjectives. They approach the problem by focusing on conjunctions. They first train a log-linear regression model to predict if conjoined adjectives have the same polarity, and then using a clustering algorithm, they categorize all the adjective instances into two separate clusters. The polarity is determined by the frequencies in each cluster, the higher frequency being the right polarity. For adjectives that occur modest number of times in the corpus, they report 90% accuracy.

Dave, Lawrence and Pennock[1], on the other hand, focus on the task of extracting opinions. They use numerous pre-processing techniques like thresholding and laplacian smoothing on counts. A unigram Naive Bayes model, and an SVM is used to make predictions. They also post-process the results by reweighting the predictions. They report that SVM's do not necessarily outperform NB and other models. Some issues they raise are ambivalence of human text and sparsity of features.

Another paper on sentiment analysis is by Pang and Lee[5], who focus on the task of multiclass sentiment analysis. Unlike earlier research, this paper considers finer-grained scales: they try to predict the numerical ratings in IMDb movie reviews. For classification, they train one-vs-all SVM's, which they report as performing slightly better than NB and MaxEnt models[6]. They also report, however, that "Koppel and Schler (2005) found that applying linear regression to classify documents...provided greater accuracy than OVA SVMs and other algorithms.[5]

Socher[7] takes on a more complicated formulation of sentiment analysis in his paper. He trains a Recursive Neural Tensor Network that is based on the compositionality of phrases, with each word having a polarity from 1 to 5 that contributes to the overall score. Trained on Stanfords new treebank, the RNTN model outperforms all previous methods, bringing the accuracy on movie reviews from 80% up to 85.4% on binary classification.

Lastly, Yoon Kim[3], in his paper uses pre-trained word vectors on Convolutional Neural Networks to do sentence-level sentiment classification. He shows that a simple CNN with little hyper-parameter tuning achieves better results on multiple benchmarks.

3 Dataset and Features

We acquired labeled data from Experience Project, a social networking website where users can, among other things, share posts with a mood tag that describes how they are feeling at the time:

> Text: Feeling better then yesterday but am lost
> Mood: lonely
>
> Text: Job interview today!! :D
> Mood: excited

Kanjoya, the company that runs Experience Project, provided us with 15,000 examples per emotion, for each of the 15 aforementioned emotions. We removed duplicate statuses and statuses with less than 10 characters to reduce noise. We were left with around between 12,000 and 13,000 examples per emotion, which we split between a training set (63%), a hold out cross validation set (27%), and a final test set (10%).

Then, we preprocessed this data in various ways throughout our experiments, but generally we treated the status as bags-of-words, representing each one as a vector of term frequencies, whose length was the vocabulary of our training set. We made sure to include punctuation as tokens, since it is often significant to the meaning of social media posts. On the other hand, we did not distinguish upper and lowercase text, to better capture document similarities (could be done otherwise with more data). Using the TweetTokenizer from the NLTK module took care of certain normalizations that helped our models.

In some cases, we used term frequency-inverse document frequency (tf-idf) to transform these vectors. That is, we multiplied the term-frequency weights by smoothed inverse document frequency: $1 + \log \frac{m}{n_t}$ where m is the total of documents we trained on and n_t is the number of documents which term t appeared in. This had the effect of reducing the weight of common stop words. The scikit-learn implementation of tf-idf that we used also normalized the lengths of our feature vectors to 1. Idf was learned from the training set but applied to all vectors when tested.

4 Methods

4.1 Multinomial Naive Bayes

This model makes the relaxed assumption that features, that is the weighted term frequencies, are conditionally independent given the mood tags. Then for an example with features $x_1 \dots x_n$, its label y can be estimated with

$$\arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

where in our case, the priors $p(y)$ are the same for each mood since our data set is evenly split among emotions, and $p(x_i|y)$ is estimated by $\phi_{i|y} = \frac{\text{\# times token i occurs in statuses labeled y}+1}{\text{total \# tokens in statuses labeled y}+n}$ These ϕ are gathered from our training set (the frequencies being weighted by tf-idf when we used it), and the $+1$ and $+n$ are Laplace Smoothing.

4.2 Softmax Regression

This model predicts the conditional probability of each label y given x, using the following formulation: $p(y = i|x; \theta) = \phi_i = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$. As each ϕ_i is a probability, the sum of all ϕ_i's equals to 1. The model is trained by maximizing the following log-likelihood, which is equivalent to finding θ that gives the maximum number of correct predictions:

$$l(\theta) = \sum_{i=1}^{m} \log \prod_{l=1}^{k} \Big(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \Big)^{1\{y^{(i)}=l\}}$$

4.3 Support Vector Machines

And SVM finds a separating hyperplane with maximal margin, and so is trained by solving the following optimization problem (or its dual problem):

$$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C \sum_{i=1} \xi_i \quad \text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \; \xi_i \geq 0$$

where w and b are weight and intercept vectors that are later used for prediction. We trained SVMs in a one-vs.-rest scheme, that is, we trained a classifier for each mood, where $x^{(i)}$ are our feature vectors, and $y^{(i)}$ is 1 or -1 depending on whether that status is labeled with the desired mood. At prediction time, all 15 classifiers are run, and then mood with the greatest margin is predicted.

4.4 ν-SVM and RBF Kernel

The ν-SVM formulation is a re-parametrization of the C-SVM:

$$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + \frac{1}{\nu l} \sum_{i=1} \xi_i - \rho \qquad \text{s.t. } (w \cdot \Phi(x^{(i)})) \geq \rho - \xi_i, \; \xi_i \geq 0$$

As described by Schlköpf et al., the parameter ν has a natural interpretation[1]: it is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the

[1]Schölkopf, Bernhard, et al. ”Estimating the support of a high-dimensional distribution.” Neural computation 13.7 (2001): 1443-1471. APA

total number of training examples. Thus on a large data set, higher ν values make the NuSVC model on sk-learn scalable (as opposed to SVC).

In our implementation of ν-SVM, we use the RBF kernel. Radial Basis Function (RBF) is defined as $k(x, x') = \exp(-\gamma|x - x'|^2)$, and is a generalization of the Gaussian kernel. Thus, we formulate the dual problem as:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \quad \text{s.t. } 0 \leq \alpha_i \leq \frac{1}{\nu l}, \ \sum_i \alpha_i = 1$$

4.5 Convolutional Neural Network

Our implementation follows the model architecture formulated by Yoon Kim[3]. In our model, each sentence is represented as a matrix, formed by the concatenation of word vectors $x_i \in R^k$ (row vectors), as shown above in the figure. To find the hidden layers, we apply the convolution operation on the matrices, which takes windows of h words, filters them with weight $w_h \in R^{hk}$, and then applies a non-linear transformation, such as ReLu or tanh. Thus:

$$c_i = f(w \cdot x_{i:i+h1} + b)$$

A *feature map* c is then defined as $c = [c_1, ..., c_i, ..., c_{n-h+1}]$. Over the feature map, we apply a max-overtime pooling operation[2] and take the maximum value $\hat{c} = \max\{c\}$ as the feature corresponding to the window size h. Using the features generated with different window sizes, we apply a final softmax layer to make the prediction. For the back-propagation, we retrain our word vectors, and apply dropout to prevent over-fitting.

5 Experiments, Results, Discussion

5.1 Experiments

The order in which we trained the models follow the outline of Section 4. Thus the first model we trained and optimized for our task was Multinomial Naive Bayes. Surprisingly, the model gave very good results, even as a baseline. Whereas a random classifier would have performed with 6.6% accuracy on 15 classes, our Naive Bayes model was performing at 28.01% accuracy. These results were obtained with tf-idf reweighting of the vocab length document vectors. Removing English stop words decreased the accuracy by 2%. This might be indicative of how even though semantically trivial, some stop words have emotional implications.

After Multinomial Naive Bayes, we trained and optimized Softmax Regression models. While optimizing our Softmax Regression model, we conducted several checks on our pre-processing techniques. One of the conclusions we drew at this point was that stemming was actually decreasing the accuracy of the models. After cross-checking this with Naive Bayes, we decided to not stem our words. Another realization we came to was that while training logistic regression on bigrams with count vectors was over-fitting, by normalizing the vectors, tf-idf was performing very well. We saw a slight improvement when we added trigrams to our tf-idf vectorizer.

The next model we trained was a linear SVM. We trained this model with both tf-idf vectors and count vectors. Here's the learning curve of the SVM:

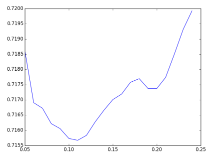


Figure 1: Learning Curve

[2]R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12:24932537

To reduce the number of features to prevent over-fitting, we tried applying PCA on our document vectors. This was not only not helpful, but also time costly. Following that, we tried training a kernel SVM with RBF, however the size of our data made it impossible for a regular computer to train the model in reasonable time. Thus we trained a ν-SVM instead with the kernel trick, sacrificing on erroFinally we trained a CNN model, hoping that word vectors pre-trained on Twitter posts[3] would help better classify the documents, since they capture semantic information. The CNN model did not outperform the SVM's, but it could be promising with more data (Increasing the number of examples as we trained proved this).

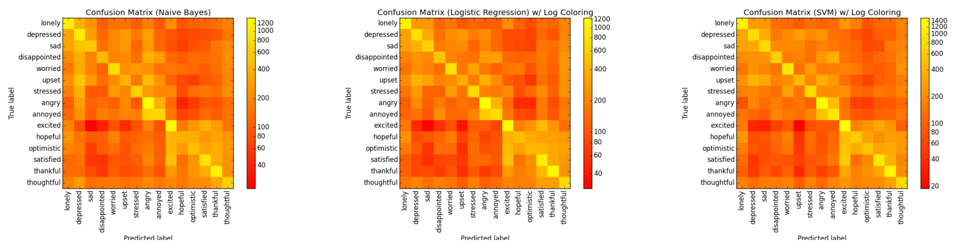
Some other models we tried that are not included explicitly in this writeup are Supervised LDA's[4], RandomForest and AdaBoost.

5.2 Results

In reporting our results, we used the accuracy metric, as we had equal number from each emotion, making precision and recall scores less relative. Another metric we thought of was giving partial credit when the models were picking up on similar emotions. Even though this approach made sense, it did not give meaningful variations, so we dropped this evaluation metric. Below we report the accuracies obtained from each model.

Model	**Accuracy**
Multinomial Naive Bayes	0.2801
Softmax Regression (one-vs.-rest)	0.2797
SVM (linear kernel, squared hinge loss, C=0.11)	0.2843
NuSVM (rbf kernel, nu=0.50)	0.2213
CNN (rbf kernel, w-dim=0.50)	0.2710

5.3 Discussion



(a) Naive Bayes (b) Softmax Regression (c) SVM

Figure 2: Normalized Confusion Matrices

One of the results we got was that SVM's were worse in separating positive from negative emotions, as evident by the confusion matrix above. This was backed up with numbers we got after splitting our mood set into two. Naive Bayes also seems to under-perform for the emotion 'upset' compared to other models. A common issue is separating similar emotions like 'annoying' and 'angry'. As this is the case for all classifiers, the data might be noisy. After hand-tagging some posts, we were able to confirm this. In fact after tagging 101 posts, we were able to get 32% accuracy, which means that our models are performing close to human judgment.

6 Conclusion and Future Work

One of the issues that we did not consider was the imbalance of emotion in space. Because of our data, we had to split emotions evenly. Also with more data, we know that we can get better results using CNN's.

[3]http://nlp.stanford.edu/projects/glove/

7 References and Acknowledgments

Acknowledgments

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

References

[1] Dave, Kushal, Steve Lawrence, and David M. Pennock. ”Mining the peanut gallery: Opinion extraction and semantic classification of product reviews.” Proceedings of the 12th international conference on World Wide Web. ACM, 2003.

[2] Hatzivassiloglou, Vasileios, and Kathleen R. McKeown. ”Predicting the semantic orientation of adjectives.” Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics. Association for Computational Linguistics, 1997.

[3] Kim, Yoon. ”Convolutional neural networks for sentence classification.” arXiv preprint arXiv:1408.5882 (2014).

[4] Mcauliffe, Jon D., and David M. Blei. ”Supervised topic models.” Advances in neural information processing systems. 2008.

[5] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. ”Thumbs up?: sentiment classification using machine learning techniques.” Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002.

[6] Pang, Bo, and Lillian Lee. ”Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.” Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005.

[7] Socher, Richard, et al. ”Recursive deep models for semantic compositionality over a sentiment treebank.” Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.