Supervised Multi-Class Classification of Tweets

Zahan Malkani
zahanm@stanford.edu

Evelyn Gillie
egillie@stanford.edu

14 December, 2012

Abstract

We present a study of a variety of supervised learning methods used for the problem of Twitter tweet classification. This includes Support Vector Machines (SVM), Neural Networks (NN), Naive Bayes (NB), and Random Forests. We evalutate these methods using their performance on two sources of labeled data: an 'attitudes' dataset that classifies tweets with an attitude that the tweeter might have had when composing it, and a 'topics' dataset that classifies tweets into one of a (limited-domain) topic set. We find that SVMs out-perform the others in accuracy and when used with feature selection have a reasonable runtime too.

1 Introduction

The explosive growth of data being produced in the form of short text snippets has lead to an equally voracious growth in the methods used to analyze and decode these snippets. Machine learning plays an important role in enabling such analyses, since the vastness of these datasets vexes most hand-labeling attempts.

One of the more popular mechanisms producing this data is Twitter. The medium is notorious for confounding traditional text analysis methods, since the wealth of context that older natural language processing techniques depend on is simply missing in these tweets that are artificially confined to being 140 characters in length. Accordingly, selecting informative features to extract from the little context that we are given was a priority.

We explore a variety of supervised learning techniques to use in two classification tasks: 1), labelling tweets with an attitude from the *speaker's* perspective (this is in contrast to most classification systems, that categorize tweets from the *audience's* perspective, and usually just has two classes, `positive` and `negative`), and 2) labelling tweets based upon *topics*, where we have a predefined set of topics that we are looking for.

We pursue modern approaches to classifying tweets, and explore their applications to real-world phenomena.

2 Related Work

Text classification is a well traversed area of machine learning, thanks to its potential for wide-reaching impact. [1] and [2] present good overviews of the methods used so far and their relative strengths and weaknesses. In [1], Yang and Liu go over the theory behind classifier methods such as Support Vector Machines, Neural Nets, k-Nearest Neighbor and Naive Bayes approaches. Their results do not offer much by way of quantitative outcomes of using these methods to classify short text snippets as we have. Lee makes some fascinating observations about how humans perform (long form) document classification in [2], providing a Bayesian model that fits how people seems to reason about this problem. His approach seems specific to the limited-domain topic classification problem though, that will be relevant for the second dataset in our paper.

In [3], Agarwal et. al. discuss polarity features as inputs for supervised learning methods, specifically SVMs, in order to limit the feature space so that it can meaningfully be divided up by an SVM. These features, though we hadn't realized at the time, are similar in nature to the conditional frequency features that we trained our SVMs on eventually to capture the conditional probability of features corresponding to classes.

O'Conner et. al. provide very useful Twitter-specific tokenizers and emoticon analyzing algorithms in [4], and the generalization of binary classification trees to the multi-class problem used by Lee and Oh in [7] is the same as what we use for our multitree classification algorithm.

3 Data

3.1 Data Collection

In August, we started collected Twitter data using Twitter's streaming API on an Elastic Cloud Compute instance. This collected around 55-60 tweets per second during peak hours, and 25-30 during non-peak hours.

3.1.1 Obama / Romney Dataset

We selected a sample of 800 tweets from October 28th for Romney and 1300 tweets from November 4th for Obama. We then had these tweets labeled with emotional attitudes on Amazon Mechanical Turk using Amazon certified categorizers, getting two categorizations for each tweet.

The emotional attitudes were chosen with consulation from a social data analytics startup, interested in a similar problem. Accordingly, we ended up with the set

```
{happy, mocking, funny, angry, sad,
hopeful, none / other}
```

Workers agreed on the label on only 42.3% of the samples. We manually reviewed a sample of the labels and

Emotion	Percent of Dataset
Mocking	0.45
Funny	0.28
Sad	0.08
Angry	0.03
Happy	0.01
Hope	0.01
None	0.19

Table 1: Distribution of Classes in Emotions Dataset

found that for some tweets this is understandable, as what one person may construe as mockery another can readily define as anger, yet for a few tweets we would not have agreed with either. After consulting with the course staff we ultimately decided that the data was good enough to use as an experiment and left it at that. For the rest of the computations, if there are two labels and a classifier predicts one of them, we consider it correct.

3.1.2 Topic Labeling Dataset

Our second dataset uses a sample stream from August 6th, in which the Olympics are beginning, the Mars Rover had just landed, the Apple-Samsung lawsuit was in the press, and Obama was expectedly in the news. Hence, tweets are labeled by their topic:

`{obama, olympics, mars, apple, none}`

Overall, we collected around 3 million tweets, of which we labeled 100,000 using Mechanical Turk. Of those 100,000, around 4,000 are labeled as being about any of the above topics.

the topic distribution are shown in 2

Topic	Percent of Dataset
Olympics	0.40
Apple	0.23
Mars	0.22
Obama	0.15

Table 2: Distribution of Classes in Topical Dataset

3.2 Data Processing

We take each tweet through a pipeline of steps. First, we lowercase the text to improve recall of unigram matching. Then we tokenize it with the Twitter-specific tokenizer from [4], which preseves hashtags and emoticons. Then we perform Porter stemming on the words, again to improve recall of unigram matching. Then we construct singleton and bigram features for the tweet. The other features are discussed separately.

3.2.1 Twitter Speech Features

We wanted to more carefully consider the type of language used in the these text snippets. We considered a multitude of features that we chose ourselves, including

- Emoticons / Emoji
- Punctuation (!!!!!)
- Capitalization (LMAOOOO)
- Dialogue (Retweets and @)
- Negation
- Sentence-level sentiment
- Cursing

We were able to see some small improvement to SVM accuracy for certain classes through addition of these features, as can be seen in table 3 (these tests were performed on a small, high-accuracy dataset).

Bag-of-words		+ Twitter specific	
emotion	accuracy	emotion	precision
angry	.9533	angry	.9659
mocking	1	mocking	.9699
happy	.9828	happy	.9710
sad	.9792	sad	1
hopeful	.9811	hopeful	.9709
funny	1	funny	.9879
none	.9719	none	.9713

Table 3: Adding Twitter Speech Features to SVM Input

It was disappointing to see that for most of the emotions our engineered features hurt performance, so we'd like to offer a special shout-out to the angry tweets for their user of upper-case and exclamation points.

3.2.2 Class Distribution Features

While exploring the performance of `mocking` and `angry`, we decided to analyze the normalized distribution of the sum of the weights by each token over an entire feature set, believing that this would enable us to see where some labels were nearly the winner only to just barely lose out to another label. More precisely,

Let

- $F = f_0 ... f_m$ be the vocabulary of features.
- $S = s_0 ... s_n$ be the sample input such that s_i is defined by its features $s_i = f_0 f_1 ... f_k$
- $L = l_0 ... l_n$ be the corresponding known labels be a multi-class vector.
- W be a $m \times n$ matrix such that $W_{i,j} = \sum_{s_k} I(f_i \epsilon s_k \wedge l_{k,j} = 1)$, namely W counts the co-occurences of features and labels.
- for sample s_k, v_k as $v_k = \sum_{f_i \epsilon s_k} W_{i,j}$

Upon analyzing the data we made the important realization that these vectors were strongly correlated with the labels, much more so than any element in isolation. We quickly tried these as feature vectors in our SVM framework and produced excellent results. We analyzed the literature for similar feature selection, and it seems Agarwal, et. al. introduce a similar polarity-prior term in [3]; our feature is an extension of this to the multi-class case.

Ultimately, we realized that these are precisely the conditional feature class counts that Naive Bayes exploits to good effect, but it provided a useful insight into the workings of SVMs.

4 Algorithms

4.1 Feature Selection

After stemming words and adding all words and bigrams to the feature set, our feature space is in the thousands. With methods such as SVM or random forests, we need to trim down our feature set to get reasonable running times. We tried three feature selection methods:

Our first feature selector was a basic frequency pruner: features that did not occur frequently at all were thrown out (often, these were misspellings).

Our second feature selector was a mutual information feature selector. Given a feature set F and class set C, the mutual information score is:

$$I(F;C) = \sum_{f,c} P(f,c) \log \frac{P(f,c)}{P(f)P(c)} \tag{1}$$

Which translates to:

$$\tfrac{N_{11}}{N} ln(\tfrac{NN_{11}}{N_{1.}N.1}) + \tfrac{N_{01}}{N} ln(\tfrac{NN_{01}}{N_{0.}N.1}) + \tfrac{N_{10}}{N} ln(\tfrac{NN_{10}}{N_{1.}N.0}) + \tfrac{N_{00}}{N} ln(\tfrac{NN_{00}}{N_{0.}N.0})$$

Where N is the total number of tweets, N_{11} is the number of tweets of the given class containing the given word, and N_{01} is the number of tweets of any other class containing the given word.
Our third feature selector was χ^2 feature selection:

$$\frac{(N_{11} + N_{10} + N_{01} + N_{00}) * (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) * (N_{10+11}) * (N_{10} + N_{00}) * (N_{01} + N_{00})} \tag{2}$$

Mutual Information and χ^2 feature selection is usually formulated in terms of binarry classification problems; we extend it to our multi-class problem by throwing out features whose maximum score with *any* class is below some threshold. That is, is a feature does not score highly with any class, we do not include it in our feature set.

Examples of high-scoring features for each feature selection method are shown in 5

4.2 Naive Bayes

The first learning algorithm that we tried was Naive Bayes. Despite Naive Bayes' assumption of feature independence (an assumption which doesn't hold for natural language) it appears to perform resonably well on our datasets. The algorithm works by learning the conditional probability for each particular feature, conditioned on the label of the training example being processed. It

Method	Threshold	features	runtime	accuracy
None	N/A	6000	1m30s	0.92
Frequency	2	1093	25s	0.93
MI	0.01	1852	23s	0.92
MI	0.1	1716	20s	0.93
MI	0.2	1635	20s	0.96
χ^2	0.1	5989	20s	0.87
χ^2	5.0	1814	25s	0.91
χ^2	20.0	184	10s	0.89

Table 4: Results of an SVM run using feature selection. (Using 700 examples of the topics dataset)

then makes a prediction for an unlabeled point by calculating the posterior probability for each class and predicting the maximizing assignment.

$$\underset{c}{\text{argmax}} p(C = c) \prod_{i}^{n} p(F = f_i | C = c) \tag{3}$$

The independence assupmtions made in NB (as can be seen in the second term of (3)), has the important side-effect of allowing a Naive Bayes classifier handle a large number of features with very little sacrifices in accuracy due to the curse of dimensionality that often effects other learning algorithms. This is very important for text classification, where often the feature space size dwarfs the other relevant dataset metrics.

We see this particular trait serve Naive Bayes well, since it does rather well when using the simplest feature set: bernoulli prescence vectors, leading to what is often called the multivariate Bernoulli Naive Bayes classifier. The results are summarized in table 6.

Name	size	10-fold accuracy
Romney dataset	800	54%
Obama dataset	1300	41%
Topics labeling	40,000	78%

Table 6: Accuracy of Naive Bayes for each dataset

After running Naive Bayes, we looked at the conditional probabilities of each feature, and pulled out the most informative ones for each class as shown in table 12.

Topical Dataset		Obama	
χ^2	Mutual Info	χ^2	Mutual Info
presid	googl	ridicul (sad)	vote (hopeful)
billionair	ipod	lost (afraid)	lie (angry)
money	#teamusa	nate (happy)	elect (angry)
@marscurios	us	disturb (sad)	campaign (angry)
mar	planet	86% (happy)	presid (angry)
#msl	iphon	lie (angry)	left (angry)
money	4s	newspap (afraid)	tie (sad)
iphon	@london2012	cnn (sad)	cnn (sad)
rover	basketball	die (angry)	fight (hopeful)
olymp	planet	random (afraid)	final (hopeful)

Table 5: High-scoring features using feature selection

money	obama : olympi = 193.2 : 1.0
land,on	mars : olympi = 144.4 : 1.0
rais	obama : olympi = 90.5 : 1.0
@marscurios	mars : apple = 77.9 : 1.0
app	apple : olympi = 60.0 : 1.0
what,you	mars : olympi = 53.9 : 1.0
safe	mars : olympi = 53.9 : 1.0
join	obama : olympi = 50.3 : 1.0
land	mars : olympi = 47.3 : 1.0
latest	mars : apple = 45.5 : 1.0
busi	obama : olympi = 45.0 : 1.0
youtub	apple : olympi = 44.7 : 1.0

Table 7: NB Most Informative Features

One final note is that Naive Bayes is also a very fast classifier – it handles a large feature space without a huge time cost. Our implementation of NB classified all datasets in under 5 seconds.

4.3 Neural Networks

The next machine learning algorithm we applied was an Artifician Neural Network. We used a Tahn layer, Softmax output multiclass neural network. Using three hidden layers, we were able to achieve 86% accuracy over the five-class topic dataset. ANNs outperformed Naive Bayes, but with a huge time cost – running 10-fold cross-validation in parallel over 700 examples took several minutes.

Layers	Feat Sel	Threshold	Accuracy
2	χ^2	1.0	0.79
3	None	N/A	0.86
3	χ^2	1.0	0.87
3	MI	0.1	0.87
3	MI	0.2	0.86
3	MI	0.1	0.87
4	χ^2	0.1	0.83

Table 8: Neural Net Accuracies Over the Topical Dataset

4.4 Support Vector Machines

SVMs are widely known as an excellent learning model. We used an implementation from the Milk[5] library, and customized that as detailed below.

4.4.1 Multi-Class Classification

In order to work with with multi-class classifications, we looked at two SVM models: one-vs-one and one-vs-rest.

Dataset	mode	accuracy
Attitudes	one-vs-one	0.57
Attitudes	one-vs-rest	0.59
Topical	one-vs-one	0.92
Topical	one-vs-rest	0.92

Table 9: SVM multiclass methods

The one-vs-rest method appears to have slightly higher acuracy over the attitudes dataset, so we proceed with that.

We trained over the same features as in previous models (singletons, bigrams and twitter speech features), and to preprocess the data, we used feature selection to remove linearly dependant features, and also ran stepwise discriminant analysis (SDA). SDA functions by selecting the feature that is most correlated with preditions, removing its variance from the other features and then repeating the step to add more features.

Kernel: We experimented with two kernels: radial basis function and dot product. For the radial basis function kernel, we performed a grid search over values of C from $2^{-2,-1,\ldots 3,4}$ and σ from -4 ... 4

Dataset	mode	accuracy
Attitudes	RBF	0.59
Attitudes	Dot	0.57
Topical	RBF	0.92
Topical	Dot	0.90

Table 10: A SVM Kernel Comparison

Again, the results are close, but the RBF kernel with grid search appears to outperform the dot product kernel.

Using the feature selection techniques outlined earlier and these optimal SVM settings, we were able to achieve a very high accuracy over the topical dataset. Using χ^2 feature selection, RBF kernel, and a one-vs-rest SVM classifier, we achieved 93% 10-fold cross-validation accuracy over all 4000 examples.

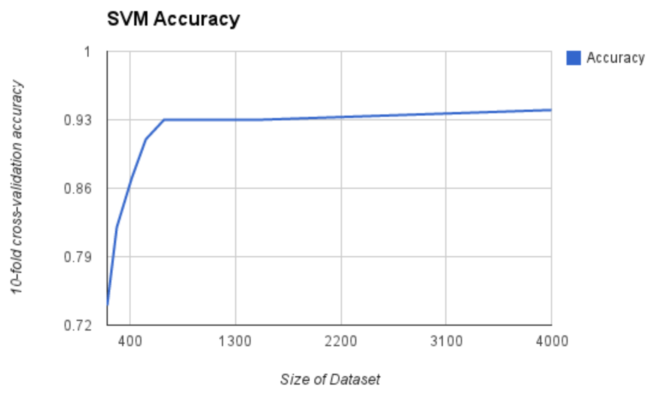


Figure 1: SVM accuracy and Dataset Size (Topic Dataset)

4.4.2 Class Distribution Features for Attitudes Dataset

As discussed in the section on Class Distribution Features, we had a breakthrough on the attitudes dataset front, when we decided to use the class conditional probabilities of each feature to construct our SVM input space. We ended up settling on using just one input per class, that was the sum of the condition probabilities of each feature for that particular class, as seen in the training data.

Since these features were presumably highly seperable with the Radial Basis Function kernel, the SVM accuracy was greatly increased, and the runtime was brought down considerably. Running 10-fold cross validation on the datasets took no longer than 5 minutes. The results for running a one-vs-rest SVM using these inputs over the various attitudes datasets are presented in table 11.

Name	size	10-fold accuracy
Romney dataset	800	92%
Obama dataset	1300	90%
Romney + Obama dataset	2100	89%

Table 11: Accuracy of SVMs with Class Distribution Features

We would expect this technique to do better with increasing dataset size, but we suspect that internal variations in the datasets themselves (the Obama dataset had more uninformative tweets like `LOOOL!!`) overpowered this overall effect.

4.5 Random Forest

We experimented with three random forest classifiers, using one-against-one, one-against-rest, ECOC ([8], and multi-tree learners. The multi-tree learner performed in the 70% range on the topical dataset, using χ^2 feature selection, and ECOC, one-against-one and one-against-rest yielded accuracies in the 90% range. We used stringent feature selection ($\chi > 1.0$) given the slowness of random forests.

Learner	Training examples	Accuracy
ECOC	700	0.92
Multi-tree	700	0.72
one-against-rest	700	0.81
one-against-one	700	0.90
one-against-one	1000	0.90
one-against-one	2000	0.90
one-against-rest	700	0.92

Table 12: Neural Net Accuracies

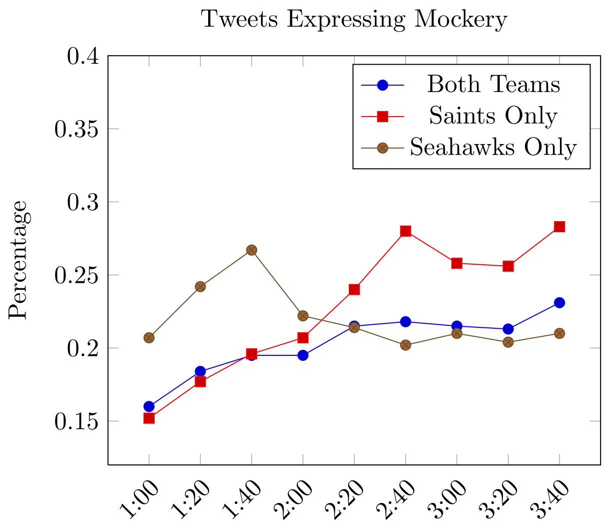
Despite its high accuracy, we decided not to try ECOC with larger datasets due to its prohibitive slowness.

5 Case Studies

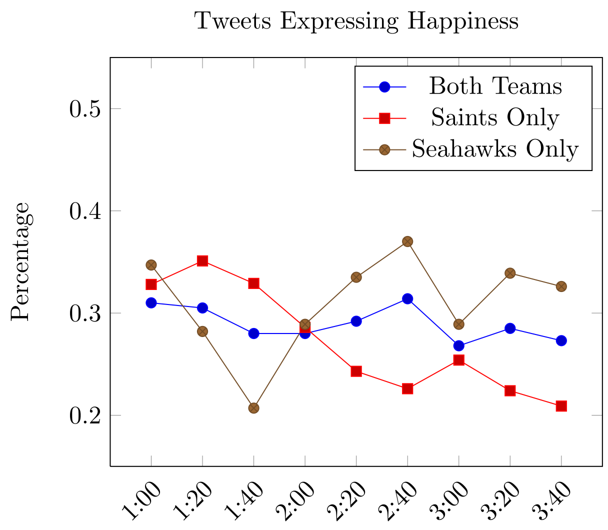
5.1 Fans Reacting to an Underdog Comeback

Now that we have a working model, we would like to evaluate it on real-world data. We collected approximately 80,000 tweets containg the keywords "Saints" or "Seahawks" between 1:00 pm and 4:00 pm on the day of a playoff game between the two teams .

In this case, the 11-5 Saints were slated to dominate the game against the 7-9 Seahawks. Surprisingly, the Seahawks won the game, causing a ripple of emotions to spread through the twitosphere. We captured that effect by analyzing the change in attitude distributions over twenty minute intervals.



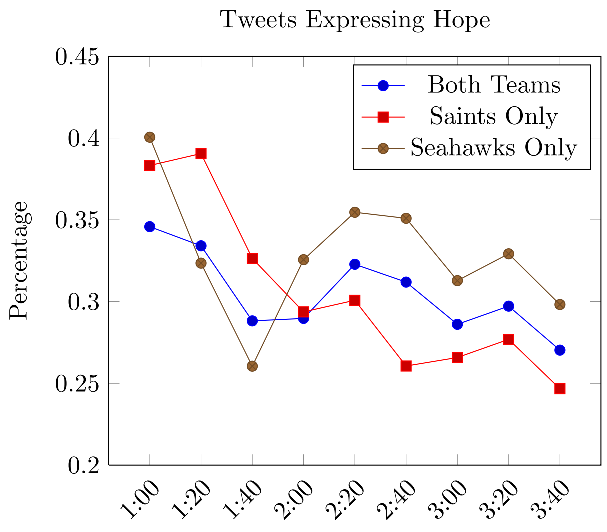
We can see where the Saints took an early lead, at the end of the first quarter they were up 10-7. Over the next



two hours, however, the Seahawks outscored the Saints 10-27. Then at the end of the game the Saints mounted a brilliant 16-10 rebound, but unfortuantely it was too little too late, and the Seahawks won the game.

Intuitively, these events align with the rise and fall of fans' emotions. We can even observe subtle differences, for example the difference at the 3:00 mark between `mocking` and `angry`, where Saints fans are angry at the Seahawks but not mocking them, while Saints fans are angry at the Saints and the Seahawks fans are mocking them.

We can see an interesting phenomenon with `hopeful`, at the beginning of the game both sides are hopeful. Initially the Saints are expected to win and the Saints to lose, so when the Saints take an early lead the Seahawks lose their hope for an upset. When the tide is turned we can see a resurgence in hope. The truly interesting part is that henceforth both teams observe patterns related to when they scored, yet the overall amount of hope decreases steadily after the first half of the game, as fans are hopeful for the teams at the beginning of the game and then later pre-occupied with other emotions.



6 Conclusions

Overall, SVM using a grid search over a Radial Basis Function Kernel outperformed Naive Bayes, neural nets, and random forests. Random forest methods were able to come close to SVM accuracy, but were considerably slower. Further, our feature selection methods were able to cut down dramatically on time without significant accuracy penalties.

Even on the 'attitudes' dataset, when used with our class distribution features, the SVM classifiers performed well, and seemed accurate enough to explore previously unseen tweets about a real-world event.

7 Acknowledgements

We would like to thank Austin Gibbons for his consultation on the project. We would like to thank Richard Socher for his help in guiding the project. We would like to thank the InfoLab for lending us their computational and data resources.

References

[1] Yiming Yang and Xin Liu. A Re-examination of Text Categorization Methods `http://www.inf.ufes.br/~claudine/courses/ct08/artigos/yang_sigir99.pdf`

[2] Michael D. Lee. Fast Text Classification Using Sequential Sampling Processes `http://lvk.cs.msu.su/~bruzz/articles/classification/Fast%20Text%20Classification%20Using%20Sequential.pdf`

[3] Agarwal et al, Sentiment Analysis of Twitter Data, 2011.

[4] Brendan O'Connor, Michel Krieger, and David Ahn. TweetMotif: Exploratory Search and Topic Summarization for Twitter. ICWSM-2010.

[5] Milk. `http://luispedro.org/software/milk`

[6] NLTK `http://nltk.org/`

[7] Jin-Seon Lee and Il-Seok Oh. Binary Classification Trees for Multi-class Classification Problems `http://www.primaresearch.org/ICDAR2003/Papers/0141_503_lee_j.pdf`

[8] `http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume2/dietterich95a.pdf`