CS224N Final Project
Sentiment Analysis of Tweets: Baselines and Neural Network Models

Kai Sheng Tai
(advised by Richard Socher)
(Dated: December 6, 2013)

Social media sites such as Twitter are a rich source of text examples expressing positive and negative sentiment. In this project, we investigate the classification accuracy achieved by a range of learning algorithms on a Twitter sentiment dataset. We evaluate the performance of a neural network classifier over bag-of-words features in relation to simpler linear classifiers.

This is a joint project with CS229.

I. INTRODUCTION

The goal of sentiment analysis is to classify text samples according to their overall positivity or negativity. We refer to the positivity or negativity of a text sample as its *polarity*. In this project, we investigate three-class sentiment classification of Twitter data where the labels are "positive", "negative", and "neutral".

We explore a number of questions in relation to the sentiment analysis problem. First, we examine dataset preprocessing specific to the natural language domain of tweets. We then evaluate a number of baseline linear models for sentiment analysis. Finally, we attempt to improve on the performance of our baseline models using neural networks initialized with linear model weights. All the algorithms we consider in this project are supervised methods over unigram and bigram features.

II. DATASET COLLECTION

We evaluate the performance of our classifiers on a test set derived from two hand-labelled Twitter sentiment datasets: (i) the SemEval 2013 "B" task dataset (7,458 examples) [10] , and (ii) the Sanders Analytics dataset of product reviews on Twitter (3,220 examples) [11]. From the combination of these two datasets, we hold out a test set of 2,136 tweets. The remaining tweets from the combined dataset is used as a training set. We refer to this as the "base" training set.

To study the effect of additional training data on performance, we create a second, "augmented" training set by combining the base training set with labelled data from several additional sources:

- The Stanford Sentiment Treebank data (239,232 examples): a sentiment dataset consisting of snippets from movie reviews [12]
- Tweets from news sources (21,479 examples) [13]
- Tweets from keyword search (52,738 examples) [14]

In total, the augmented training set contains 332,669 examples. The first three datasets contain labelled positive, neutral and negative examples. We assign all exam-

Honey Badger and Jordan Jefferson both got arrested for pot possession for the 3rd time. He's out for good #fuckkkk #herbaddiction

Yay !!!!RT kellymonaco1: Excited to interview the new cast of DWTS tonight for E News! They have no idea what their in for ;)

I went to the Marijuana March, but I do not remember what happened there.

FIG. 1: Sample tweets from base training set.

Protesters, anti-riot police in Egypt clash near a Cairo university: http://t.co/ohvtgqIJrE -NS

BREAKING: US National Hurricane Center says Tropical Storm Raymond has formed in Pacific south of Mexico.

Oklahoma City Thunder will rebuild school basketball courts in Moore, Okla., devastated by tornado: http://t.co/cngFapdYtz -SS

FIG. 2: Sample tweets from news sources (labelled neutral).

ples drawn from news sources a neutral label. The tweets retrieved from search queries with positive polarity are labelled "positive", and the tweets retrieved from search queries with negative polarity are labelled "negative".

III. PREPROCESSING

We first investigate the effect of various preprocessing on classification accuracy. The baseline preprocessor strips all nonalphanumeric characters from the text. We test the following additional preprocessing steps in a cumulative fashion: (i) preserving punctuation symbols, deduplicating and adding whitespace around punctuation, (ii) preserving emoticons and adding whitespace around punctuation, (iii) replacing user references with a generic token (@XYZ $\rightarrow$ $\langle$USER$\rangle$), (iv) replacing URLs with a generic token, (v) replacing numbers with

RT @deaddollsclub: Things look seriously delicious down here. #kimchi #amazing @GalbiBros http://t.co/FcGQZiAExf

Tickets go on sale in 10 mins!!! HARDWELL #fingerscrossed #awesome #neededneedneed http://t.co/jKzTRyeUry

im about to smash my phone up @MattyOTBC style #Angry #needupgrade #pieceofshit

FIG. 3: Sample tweets from keyword search (labelled based on polarity of search term).

is the one hour and thirty-three minutes spent watching this waste of time .

going to face frightening late fees

zero thrills , too many flashbacks and a choppy ending make for a bad film

FIG. 4: Sample snippets from sentiment treebank.

a generic token, (vi) using stemmed tokens.

For each additional preprocessing step, a logistic regression classifier (with regularization parameter $C = 0.1$) is trained on the bigram features derived from the preprocessed text. The preprocessing steps that yielded improvements on test accuracy are listed in Table I. User reference replacement and stemming yielded lower test set accuracies. The improvement realized by the emoticon handling step is reflective of the importance of emoticons as lexical features in this domain. In the subsequent analysis, we use the full set of preprocessing options that yielded improvements in accuracy.

IV. MODEL EVALUATION

We use two performance metrics for model evaluation: (i) the percentage of correctly-labelled examples, and (ii) the average of the F1-scores for the positive and negative classes, $F = (F_{\text{pos}} + F_{\text{neg}})/2$, where as usual F_c is the harmonic mean of the precision and recall for class c. Note that even though this F1-score is only an average of the F1-scores for the positive and negative classes, an incorrect labelling of a neutral example will still hurt the precision of the positive or negative class, and would therefore impact the overall F1-score of the classifier.

	train	test	change
base	82.6	70.8	0.0
+punct/special	82.7	71.6	+0.8
+emoticons	82.9	72.2	+0.6
+URLs	83.0	72.5	+0.3
+numbers	82.9	72.6	+0.1

TABLE I: Effect of preprocessing on classifier accuracy

V. BASELINE MODELS

A. Linear models

We set performance baselines using the following algorithms: logistic regression (LogReg), Multinomial Naive Bayes (MNB), Support Vector Machine (SVM) with the linear kernel, and the Naive Bayes SVM (SVM) [1]. For these algorithms, we let $C \in \mathbb{R}$ denote the inverse L2 regularization strength.

We now give a brief description of the Naive Bayes SVM algorithm.

B. Naive Bayes SVM (NBSVM)

The Naive Bayes SVM is a simple but surprisingly effective method for text classification, achieving state-of-the-art performance for a two-class sentiment classification task on a large dataset of 50k IMDB movie reviews [2] using only bigram features [1]. In [1], a version of the NBSVM for two-class classification is described. Here we give a natural generalization of the method to the multiclass case ($\geq$ 3 classes).

Let $\mathbf{f}^{(i)} \in \{0,1\}^{|V|}$ be the binary feature occurence vector for the ith example, where V is the set of features and $\mathbf{f}^{(i)}_j = 1$ iff feature V_j occurs in the example. For each class c, define the count vector $\mathbf{p}_c$ as $\mathbf{p}_c = \alpha + \sum_{i:y^{(i)}=c} \mathbf{f}^{(i)}$, where α is the Laplace smoothing parameter. For the MNB and NBSVM algorithms, we take $\alpha = 1$. For each class c, define the log-likelihood ratio vector $\mathbf{r}_c$ as:

$$\mathbf{r}_c = \log\left(\frac{\mathbf{p}_c/\|\mathbf{p}_c\|_1}{1 - \mathbf{p}_c/\|\mathbf{p}_c\|_1}\right). \tag{1}$$

For each class c, train a one-vs-rest linear SVM $s_c(x) = \mathbf{w}_c^T\mathbf{x} + b_c$ on the set of feature vectors $\{\mathbf{f}^{(i)} \circ \mathbf{r}_c\}$, where $\mathbf{w}_c$ is the weight vector for class c, b_c is the bias term, and $\circ$ denotes the elementwise product. Following [1], we interpolate between MNB and SVM for each class:

$$\mathbf{w'}_c = (1-\beta)\bar{w}_c + \beta\mathbf{w}_c, \tag{2}$$
$$b'_c = \beta b_c \tag{3}$$

where $\bar{w}_c = \|\mathbf{w}_c\|_1/|V|$ is the mean magnitude of $\mathbf{w}_c$, and $\beta \in [0,1]$. Let $s'_c(x) = \mathbf{w'}_c^T x + b'_c$. Given example x, the NBSVM classifier returns the prediction $\hat{y} = \arg\max_c\{s'_c(x)\}$. We also consider the possibility of

	base		augmented	
Method	Accuracy	F1	Accuracy	F1
LogReg	**<u>71.11</u>**	57.57	73.13	**<u>63.32</u>**
MNB	68.12	47.72	66.20	58.06
SVM	70.65	54.55	**73.17**	61.25
NBSVM-0	66.76	59.07	63.20	54.69
NBSVM	66.95	**59.57**	58.90	54.84
NN-LogReg	70.27	57.51	72.14	**62.58**
NN-MNB	70.74	56.82	**<u>73.36</u>**	62.47
NN-SVM	**70.83**	**<u>60.02</u>**	72.05	62.22
NN-NBSVM	70.13	59.00	69.99	58.47

TABLE II: Results on Twitter sentiment test set. The top result from each column is **<u>underlined</u>**, and the second-highest result is in **bold**. All classifiers use bigram features. **LogReg**: $C = 1$. **SVM**: $C = 0.01$. **NBSVM**: $C = 0.01, \beta = 0.25$. **NBSVM-0**: NBSVM without intercept fitting. **NN**: neural networks initialized with linear model weights, using $g = 50$ feature groups per class. **NN-LogReg**: $C = 0.1$ for LogReg classifier. **NN-SVM**: $C = 0.1$ for *SVM* classifier. **NN-NBSVM**: $C = 1.0$, $\beta = 0.25$.

omitting the bias term by training SVMs that do not fit an intercept; for this version of the NBSVM, omit the terms b_c and b'_c in the above description.

In our experiments, we use $\beta = 0.25$.

VI. NEURAL NETWORK MODELS WITH LINEAR MODEL INITIALIZATION

A. Description

Neural networks initialized using weights derived from linear models have been shown to exhibit good performance on a variety of classification tasks, often improving on the linear model on which it is based [3]. Here we give a description of this class of neural network models which, in effect, act as meta-algorithms over their base linear models.

1. *Structure*

Let n be the number of classes and let $g \in [1, |V|]$ be a parameter that denotes the number of *feature groups per class*. The number of hidden units is given by the product gn, and the number of output units is n. The output of the network for an input $\mathbf{x}$ is:

$$f(\mathbf{x}) = W_o \tanh(W_h \mathbf{x} + \mathbf{b}_h) + \mathbf{b_o}, \tag{4}$$

where W_o is a $n \times gn$ matrix, W_h is a $gn \times |V|$ matrix, $\mathbf{b_o}$ is a vector of dimension n and $\mathbf{b}_h$ is a vector of dimension gc.

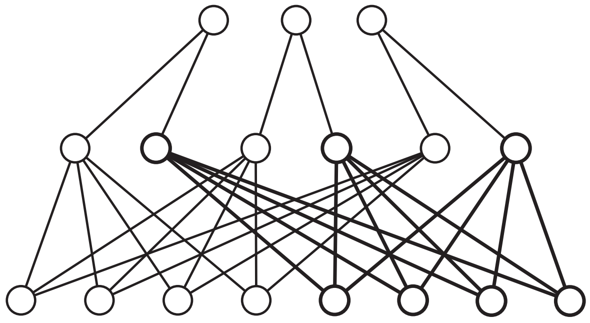

FIG. 5: Example neural network structure for 3 classes, 8 features and 2 feature groups per class. Bold nodes and connections correspond to one feature group. Connections with weights that are initialized to zero are omitted from the illustration but are not *constrained* to remain at zero.

2. *Initialization with linear model weights*

Let $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ be a set of weight vectors derived from a linear model. Let $\{\pi^{(1)}, \ldots, \pi^{(g)}\}$ be a random partition of the set of feature indices $\{1, 2, \ldots, |V|\}$ into g equally-sized groups.

The initialization scheme is illustrated in Fig. 5. For each class c, associate g hidden units $\{h_1^{(c)}, \ldots, h_g^{(c)}\}$. For all $i \in [1, g]$ and $j \in [1, |\pi^{(i)}|]$, the weight of the connection between $h_i^{(c)}$ and the input unit $x_{\pi_j^{(i)}}$ is given by $(\mathbf{w}_c)_{\pi_j^{(i)}}$. In other words, connect each input unit to each of the n hidden units corresponding to its assigned feature group using the weight assigned by the linear model. The remaining input-hidden weights are initialized to zero.

The connection between each hidden unit and its corresponding output unit is initialized with weight $1/g$. The hidden biases b_h are initialized with the the intercept terms fit by the linear model divided by g (if the linear model has no intercept terms, then this is initialized to zero), and the output biases b_o are initialized to zero.

For logistic regression, SVM and NBSVM, the weights are initialized using the raw linear model weights. For MNB, we stack the weight vectors $\mathbf{r}_c$, then subtract the mean and normalize by dividing by the component with the largest absolute value.

In our experiments, we use $g = 50$.

B. Training

1. *Gradient descent methods*

The neural network is implemented using the Theano library in Python [4]. Neural networks trained on the base dataset are optimized using minibatch SGD (batch size 50) with AdaGrad [5] and an initial learning rate of 0.001. For networks trained on the much larger augmented dataset of $\sim$ 300k examples, we use minibatch SGD with a fixed learning rate of 0.01 in the interest of computational speed.

We use the multiclass hinge loss as the cost function:

$$J(W_h, W_o, \mathbf{b}_h, \mathbf{b}_o) = \frac{1}{m}\sum_{i=1}^{m} \max_{c}(1\{y^{(i)} \neq c\} - \quad (5)$$
$$s_{y^{(i)}}(x^{(i)}) + s_c(x^{(i)})). \quad (6)$$

Backpropagation training is performed for 10 epochs, and the highest test accuracy over these epochs is reported.

2. Feature dropout

Random feature "dropout" [6] during training has yielded significant improvements in neural network performance in several classification tasks by preventing overfitting on groups of features. Though dropout training yields improvements for some text classification tasks [3], for this sentiment classification problem we did not observe any gains from input or hidden layer dropout during training. The following results are derived from networks trained without random feature dropout.

We conjecture that feature dropout does not help much in this task since the feature representations of tweets are already extremely sparse in the feature space induced by the very large augmented training set, so that overfitting due to feature co-dependence is already unlikely.

VII. DISCUSSION

A. Limitations of bigram features

Bigram features fail to capture the sentiment expressed by semantic information in the text beyond individual-word or individual-bigram polarities. For example, both the logistic regression and NN-MNB models predict that the following example is "negative" (the correct label is "positive"):

> Idc if tomorrow decides to be a shitty day I'm gonna make it the best goddamn day ever. #determined

The word "best" slightly biases this example towards "positive" (both models assign "positive" a higher score than "neutral"), but this is presumably outweighed by the negative polarity of "shitty" and "goddamn". The key to this example is to recognize that the structure "[I don't care] if ..." negates the negativity of "shitty day", but neither model is able to take this information into account.

B. Ambiguity of Aspect-Specific Examples

Our classifiers also have difficulty with examples that express conflicting aspect-specific sentiment. For example, the following example (again "positive" mislabelled as "negative") expresses conflicting sentiment regarding PCs and Macs:

> So, I am using my work PC (NEVER EVER) to get a feel for it; it has the worst speakers ever!! apple you have spoiled me!! #imamac

Both classifiers assign extremely high scores to the "negative" class for this example and fail to recognize the positivity expressed with respect to Apple products. This is probably due to the ambiguity of "spoiled" and the failure to understand the out-of-vocabulary token "imamac" (a complex task that requires both morphological parsing and understanding of the expression "I'm a Mac").

Another misclassified example with conflicting aspect-specific sentiment is the following:

> Max might have to get put down tomorrow </3 absolutely heart breaking if I have to see my puppy go. Love you Maxy x pic.twitter.com/lyHVHL1F

Here the confusion is due to the word "love", which has a positive polarity that should be negated by the context.

C. Logistic Regression is a Solid Performer

With proper tuning of the regularization parameter, logistic regression offers a combination of speed and good classification accuracy that is difficult to improve upon.

D. Out-of-Domain Data Yields Improvements

Recall that we augmented our base training set of tweets with $\sim$ 200k example from the Stanford Sentiment Treebank dataset. Even though this dataset was derived from a corpus of movie reviews, its inclusion in the training set still yielded improvements in classification accuracy on the test set. The improvement is significant: a logistic regression classifier trained on the augmented training set with treebank data excluded achieves a test accuracy of 70.83%, while the test accuracy achieved after training on the entire augmented training set is 73.13% — an improvement of 2.3%. Training on the treebank data alone gives a test accuracy of 62.22%, so it is indeed the combination of the datasets that yields the observed improvement.

VIII. CONCLUSIONS AND FURTHER WORK

We find that a complex neural network classifier achieves only modest improvements over logistic regression and a linear SVM. With each tweet represented as an extremely sparse feature vector, it appears to be the case that neural network models are not well-suited to classifying short text samples using bag-of-words features.

A clear avenue for further investigation is to compare the methods discussed in this project with the parse-tree-based models described in [7], [8], and [9]. These models should be better able to leverage semantic information in classifying difficult cases such as those described in Sec. VII A.

[1] S. Wang and C. D. Manning, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (Association for Computational Linguistics, 2012), pp. 90–94.

[2] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (Association for Computational Linguistics, 2011), pp. 142–150.

[3] R. Socher, *Naive neural networks for very fast and accurate text classification*, Private communication (2013).

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, in *Proceedings of the Python for Scientific Computing Conference (SciPy)* (2010), vol. 4.

[5] J. Duchi, E. Hazan, and Y. Singer, The Journal of Machine Learning Research **999999**, 2121 (2011).

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv preprint arXiv:1207.0580 (2012).

[7] T. Nakagawa, K. Inui, and S. Kurohashi, in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (Association for Computational Linguistics, 2010), pp. 786–794.

[8] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2011), pp. 151–161.

[9] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts (EMNLP, 2013).

[10] http://www.cs.york.ac.uk/semeval-2013/task2/

[11] http://www.sananalytics.com/lab/twitter-sentiment/

[12] http://nlp.stanford.edu/sentiment/treebank.html

[13] From the following sources: AP, BBCTech, BBCWorld, ReutersBiz, ReutersLegal, ReutersScience, ReutersSports, ftfinancenews, nytimes, nytimestech.

[14] From search queries for pre-selected keywords with positive and negative polarity.