Comparing different classifiers for sentiment analysis

Final Report

Ratish Garg(rgarg7), Bhopesh Bassi(bbassi2)

May 11, 2017

Contents

1 Overview

The advent of social networks, review platforms, and online market places have made us more reliant on reviews than ever before. These days we use platforms like Yelp before getting food, Amazon before buying a product and IMDB when deciding which movie to watch. While the numerical ratings help us get a overview, it is the text associated with the review that reveals the key characteristics about the product. Sentimental analysis can help in categorizing and organizing these reviews to learn more about the product. For example, we can examine positive reviews to understand key features of the product whereas negative reviews can help in identifying faults.

In this project, we used state of the art NLP techniques and classifiers to classify IMDB movie reviews[1] to positive or negative. We then compared performance of different classifiers. Our main aim of this project was to get started on NLP problems and get to apply the concepts and algorithms we learnt in class.

2 Features

We used uni-grams + bi-grams + tri-grams as features to represent the reviews making total of 4302432 words in vocabulary. The intuition is that the unigrams alone can fail in capturing negations, conjunctions and other complicated semantics of our language. For example, negative phrases such as 'not good' could be interpreted as positive upon seeing the uni-gram: 'good'.

2.1 Feature Representation

The vector model to represent a document as a bag-of-features gives equal weights to all the features. Some of these features may be present in a lot of the documents thus making them useless. Therefore, to overcome this we used TF-IDF bag-of-feature representation for a review as it penalizes words that appear too often in the collection. We also removed stop words from vocabulary.

2.2 Feature Selection

We want to do feature selection for two reasons:

1. Feature space is huge, so algorithms like SVM and Decision Trees take too much time to run on our laptops.
2. Some of the algorithms like Decision Trees and Naive Bayes are more sensitive to irrelevant features than others like Perceptron or SVM.

We used chi-square for feature selection. Number of features was tuned along with other parameters for the algorithm. More details in section.

3 Classifiers and experimental evaluation

We used following classification algorithms to classify reviews into positive and negative and compare their performance.

1. Perceptron.
2. SVM.
3. Logistic Regression.
4. Naive Bayes with multinomial probability distribution.
5. Decision Tree.

3.1 Methodology

We use Python and Scikit Learn library.

3.1.1 Parameter tuning

We used 5-fold cross validation for parameter tuning. Number of features was to be tuned for all the algorithms. Various values that were tried were all the features and top 100, 500, 1000, 10000 and 100000 features. Other than this all algorithms had different set of parameters to be tuned listed in table 1. The results of the tuning are shown in table 2.

Classifier	**Parameters to be tuned and values tried**
Perceptron	Learning rate = (1.5, 1, 0.25, 0.03, 0.005, 0.001), Regularization = (L1, L2), epochs = (5, 10, 15, 20)
SVM	Kernel = (linear, rbf)
Logistic Regression	Regularization = (L1, L2), C = (1, 0.1, 0.01, 100, 1000, 10000, 100000)
Multinomial Naive Bayes	-
Decision Tree	Criteria = (gini, entropy), Max Depth = (500, 1000, 10000, 100000, No max depth set)

Table 1: Various classifiers and parameters to be tuned.

Classifier	**Tuned feature size**	**Other tuned params**	**5-fold avg accuracy**
Perceptron	100000	Learning rate = 0.005, Regularization = L2, epochs = 10	0.897
SVM	100000	Kernel = linear	0.893
Logistic Regression	100000	Regularization = L1, C = 100000	0.913
Multinomial Naive Bayes	10000	-	0.836
Decision Tree	500	Criteria = gini, Max Depth = 500	0.676

Table 2: Various classifiers and tuned parameters.

3.1.2 Training and testing

After tuning parameters, we trained our classifiers on 25000 training examples and then tested on 25000 test examples. Results are shown in table 3.

Classifier	**Feature vocabulary size**	**Precision**	**Recall**	**F1**	**Accuracy**
Naive Bayes	10000	0.86	0.86	0.86	0.859
SVM	100000	0.88	0.88	0.88	0.883
Logistic regression	100000	0.84	0.84	0.84	0.837
Decision Tree	500	0.73	0.73	0.73	0.73
Perceptron	100000	0.86	0.86	0.86	0.855

Table 3: **Results from classifiers on testing data set.** For each classifier, we report feature size, precision, recall, F1-scores, test accuracy for sentiment classification task.

4 Results interpretation.

1. Sentiment analysis problem is mostly linearly separable in space of n-grams bag of word representation. That is why all four linear classifiers gave accuracy of around 85%.
2. Because the problem is well solved by linear classifier, using RBF kernel with SVM didn't help. Infact, we observed longer training time and low test accuracy with RBF.
3. Decision trees are sensitive to lots of attributes and also to maximum depth. We see that they gave over 70% accuracy with just 500 features and with maximum depth of only 500. On the other hand, using more features and deeper trees performed worse.
4. Naive Bayes used 10 times less features than other linear classifiers and gave almost same accuracy as other linear classifiers. This verified our discussion in class that even though naive bayes assumptions don't hold in practice, it is still a very useful classifier.

5 Future work.

1. Explore neural networks for the problem and understand if increase in accuracy is worth the complexity.
2. We decided on number of features before hand and tried various different number of features to find the best one. Future work would be to explore techniques which can add/remove features as new data/information comes in.

6 Side Note

Third team member Abid Khan (aakhan3) has dropped out of this team.

References

[1] Source of data. `http://ai.stanford.edu/~amaas/data/sentiment/`.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[3] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[4] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070, 2002.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. *Scikit-learn: Machine Learning in Python*, volume 12. 2011.