Classification of Review Sentiments

Coauthor
Jaeyoon Jung
jyjung@princeton.edu

Coauthor
Jelani Denis
jdenis@princeton.edu

Abstract

In this paper, we perform an survey of numerous well known classifiers on the task of sentiment analysis regarding a relatively small data set of product reviews. The data set contains 3,000 reviews each with a label of either 1 or 0 for positive or negative sentiment respectively. Classifiers are trained on bag-of-words representations of these reviews, along with a variety of feature extraction and selection techniques. We find that overall, the Multinomial Naive Bayes classifier with Unigram and Bigram features and the SVM classifier with Unigram, Bigram, TFIDF, and Review Length features have the highest classification accuracy.

1 Introduction

Online user reviews are an important feedback mechanism for many Internet-based companies like Netflix or Amazon. Often, these companies give users the ability to provide a numeric rating for a particular service or product. However, this quantitative information can be enhanced by qualitative user input such as unstructured textual responses that accompany these ratings. Sentiment analysis aims at uncovering latent features within these responses that enable machines to classify them as negative, positive, or some mixture of both at scale.

In this work, we evaluate six different classifiers for the task of classifying arbitrary reviews into positive or negative sentiment category. We use the popular machine learning library Sci-Kit Learn to convert samples into bag-of-words representations. We use custom functions to extract new features, and an extremely modularized code base to test different Sci-Kit Learn classifiers with different combinations of feature extraction, selection, and hype rparameters. We perform an exhaustive comparison of classifiers in the Results and Discussion sections.

2 Spotlight Classifier SVM

Support Vector Machines are classifiers which output a linear hyperplane separating two classes of data, such that the hyperplane maximizes the margin between both classes at the support vectors (hence the name). SVM?s naturally handle binary classification, therefore, but can also be extended to do regression and multi-class classification [1].

SVM?s are a type of sparse kernel machine, which takes advantages of easier classification by kernelizing features into a higher dimensional space. Let us consider the L2 regularized loss function as determined by tuning parameters w and l [1].

$$L2(w,l) = \sum_{i}^{N} L(y_i, \hat{y_i}) + l * \|w\|^2 \, Y_i = w^T x_i + w_0 \qquad (1)$$

Where L is quadratic loss the problem is known as ridge regression, and where it is log-loss it is known as logistic regression. The goal is to minimize this loss function. For ridge regression, the solution can be reduced to a combination of inner products $x^T x$? which can be swapped out for calls to a kernel function (if we recall, mercer kernels are simply defined as the inner product between two p-vectors).

Replacing L2 with a special alternative, the hinge loss, guarantees a sparse solution in the kernelized feature space, which is exactly the motivation behind the construction of an SVM. Hinge loss can be defined as [1] :

$$L_{hinge}(y, \eta) = max(0, 1 - y\eta) \tag{2}$$

Where η represents the degree of ?confidence? we have in choosing label y = 1. The total hinge loss is not differentiable, but by using slack terms we can create an equivalent loss representation over N constraints (N is the number of samples) [1]. The objective is as follows:

$$Min\frac{1}{2}\|w\|^2 + C\sum_{1}^{N}\epsilon_i \text{ s.t. } \epsilon_i \geq 0, y_i(x_i^T w + w_0 \geq 1 - \epsilon_i, i = 1 : N) \tag{3}$$

The objective can be solved into something with the form:

$$\hat{w} = \sum_i \alpha_i x_i, \alpha_i = \lambda_i y_i \tag{4}$$

The x_i for which the alphas are positive are known as support vectors. Prediction is based on the sign of the classification result.

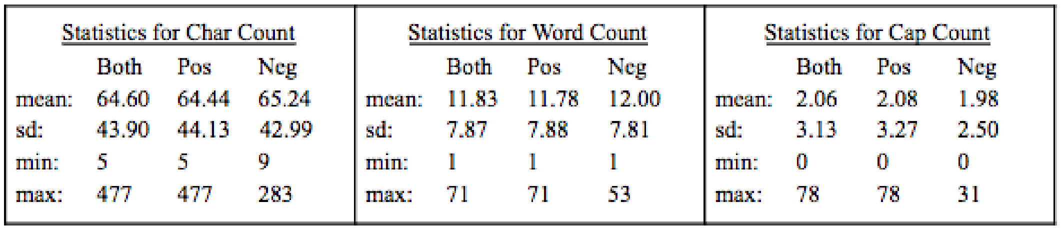
$$\hat{y}(x) = sgn(\hat{w_0} + \hat{w}^T x) \tag{5}$$

The above equations are not generated out of nowhere. They have real graphical interpretations, where $\frac{1}{\|w\|}$ is the perpendicular distance between a data point and some high dimensional hyperplane margin (scaled by a constant) [1]. Maximizing the minimum of these distances is equivalent to minimizing $\|w\|^2$, with the added constraint that labels are correctly applied to points $y_i(w^T x_i + w_0 \geq 1, i \in 1 : N)$. Where we introduce a relaxation on the degree to which points can be mislabelled, the equation introductions slack variables which get us to our original equation derived from hinge loss before [1].

One final note on hyperparameters for an SVM. There are two: C and the kernel function. C is a regularization parameter that controls the number of mistakes we accept from our classifier. The larger C is, the less we tolerate mistakes, but also the harder it is to fit non-linearly separable data. There is a trade off therefore to tuning C [1]. The ?kernel function? is simply whatever kernel we use to transform the features. These hyperparameters are usually chosen via grid search or cross-validation, and it turns out that the choice of C has quite a large effect on the performance of the kernel .

3 Description of Data

The dataset provided by the course staff for this project was a collection a 3,000 labelled product reviews. The length of the train set, as partitioned by the course staff, was 2,400 samples, and 600 for the test samples. The average length of all reviews was about 64 characters or 12 words. The statistics for sample length and capitalization count are provided in the table below for all reviews and positive/negative separately.



Statistics for Char Count

	Both	Pos	Neg
mean:	64.60	64.44	65.24
sd:	43.90	44.13	42.99
min:	5	5	9
max:	477	477	283

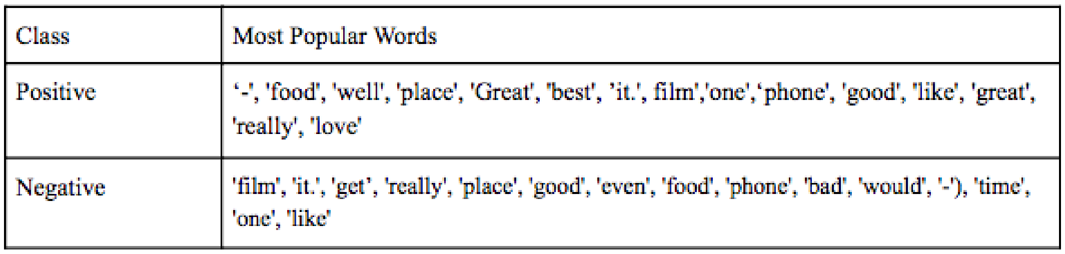
Statistics for Word Count

	Both	Pos	Neg
mean:	11.83	11.78	12.00
sd:	7.87	7.88	7.81
min:	1	1	1
max:	71	71	53

Statistics for Cap Count

	Both	Pos	Neg
mean:	2.06	2.08	1.98
sd:	3.13	3.27	2.50
min:	0	0	0
max:	78	78	31

The 15 most popular words associated with positive reviews (excluding stop words) and their counts are as follows:



Class	Most Popular Words
Positive	'-', 'food', 'well', 'place', 'Great', 'best', 'it.', film','one','phone', 'good', 'like', 'great', 'really', 'love'
Negative	'film', 'it.', 'get', 'really', 'place', 'good', 'even', 'food', 'phone', 'bad', 'would', '-'), 'time', 'one', 'like'

4 Data Preprocessing

We used CountVectorizer in the scikit-learn library to streamline the preprocessing step. We switched all upper case letter to lower case and removed punctuation and special characters. Stop-words were also removed, based on the stopword list from the nltk module. Then, we lemmatized and extracted a stem word from each token, using nltk's WordNet Lemmatizer and Porter Stemmer. Then, the CountVectorizer method output a sparse matrix in which rows represent samples (reviews), columns represent bags of word (each word, in this case), and values indicate frequency of each word. For Bernoulli Nave Bayes classifier, all non-zero values were recorded as 1 instead. At the end of the preprocessing step, we had a 2400 x 3479 spare feature matrix.

4.1 Additional Features

1. Bigram: the order of word occurrence could be informative. Bigram is helpful in recreating this without making features too specific to training data.
2. TF-IDF (term frequency-inverse document frequency): using TF-IDF values decreases weight on terms that appear very often across the entire corpus. Terms with high document frequency are not very informative and function as corpus-specific stop words.
3. Character length & word count: users with strong sentiment may write more than others.
4. Upper case count: users with strong sentiment tend to write reviews in upper case to make them dramatic.

4.2 Feature Selection

1. SVD (Singular Value Decomposition): We decomposed the sparse feature matrix into linear combination of independent components. The decomposed matrix is sorted by eigenvalue, which represents variance explained by each eigenvector (component). By taking first 100 columns, we reduced dimensionality of data to remove some noise. Because this transformation generates negative values, we could not use it for Nave Bayes classifiers.
2. Chi-Square Feature Selection: We used chi-square feature selection as an alternative of SVD. We computed chi-square score of independence between features and classes (positive vs negative) to find 100 best features that are dependent on classes.

4.3 Classifiers

The following classifiers were tested with different combinations of above features and selection methods. Unless specified, parameters are default parameters in Scikit-Leanrn module, and random state was set to 0.

1. Bernoulli Nave Bayes
2. Multinomial Nave Bayes
3. Decision Tree
4. Random Forest: using 100 estimators
5. K Nearest Neighbors: using 50 nearest neighbors
6. Support Vector Machine.

All classifiers were binary classifiers with positive as 1 and negative as 0 label. We implemented them through the Scikit-Learn module. For SVM, parameters were chosen using grid search over C values 0.001, 0.01, 0.1, 1, 10, 100, and 1000 and linear, poly, rbf, and sigmoid kernels. SVM with best performance turned out to have C = 1 and linear kernel.

4.4 Model Evaluation

For each method, we trained our model with the given training data and cross-validated on test dataset. We used accuracy and F1 scores to compare performance of different classifiers. Then, for more rigorous evaluation of models, we performed 5-fold cross validation on 3 classifiers with the best performance. We used the same folds for 3 classifiers, and used average accuracy and F1 scores to do final evaluation.

5 Results

With Bigram & TFIDF & Length Features

Classifier	Accuracy	F1	Time
Multinom. NB	0.813	0.804	1.264
RF	0.795	0.780	3.243
SVM	**0.842**	**0.835**	**118.194**

With Bigram

Classifier	Accuracy	F1	Time
Bernoulli NB	0.817	0.804	1.126
Multinom. NB	**0.818**	**0.814**	**1.241**
DT	0.775	0.761	1.643
RF	0.792	0.760	3.749
SVM	0.817	0.799	89.040

Cross Validation

Classifier	Features	Accuracy	F1
Multinom. NB	-unigram -bigram -TFI-DF	0.819	0.823
SVM	-unigram -bigram -TFI-DF	0.818	0.819

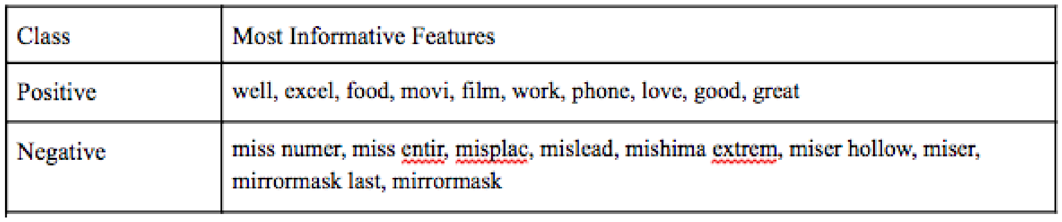
More combinations of classifier, feature extraction, and selection were run, but due to space constraints we only show the the runs that generated overall top two classification accuracies. We also show the results for a five-fold cross validation for our top two classifiers, Multinomial Naive Bayes and SVM. Other combinations ran include Bigram & TFIDF, Bigram & Length Features, Bigram & TFIDF & SVD, Bigram & TFIDF & Chi2, Bigram & TFIDF & Length Features & Chi2. Results can be seen in our accompanying jupyter notebook.

6 Discussion

We discovered that each model reacts very differently to additional features and dimensionality reduction. For example, the model that performed the best on the given test dataset was SVM (C = 1, linear kernel) fit on TF-IDF of unigrams and bigrams and features of character, word, and uppercase letter count. However, Multinomial Nave Bayes, the second best model, performs better when given just the term frequency of unigrams and bigrams. Overall, SVM and Multinomial Nave Bayes performed better than other models (we stopped adding features to noticeably bad models, such as KNN and Decision Tree, and Nave Bayes models were restricted to data with positive values), and according to the results of K-Fold cross validation, the two models are very close in accuracy and F1 score. However, SVM has time complexity greater than quadratic, and the result above also shows that fit time grows exponentially as features grow. Hence, Multinomial Nave Bayes is more appropriate for scaling on a much larger dataset.

Surprisingly, all models performed worse with feature selection- both SVD and Chi Square. One possible explanation for this is that the training data is fairly small: only 2400 reviews. Patterns of term frequency are less apparent in a small dataset, and thus both SVD and Chi Square feature selection fail to extract terms with high relevance.

Most Informative Features



Class	Most Informative Features
Positive	well, excel, food, movi, film, work, phone, love, good, great
Negative	miss numer, miss entir, misplac, mislead, mishima extrem, miser hollow, miser, mirrormask last, mirrormask

We also computed the most informative features for the Multinomial Naive Bayes classifier to shed some light on what type of features helped it to work versus which types did not. It seems that basic root words which are popular across all positive samples seem to be the most informative features for Multinomial Naive Bayes classification of positive samples. However, that is not the case for the negative class. Indeed, the most informative features associated with the negative class are much harder to make sense of at face value. Firstly, they seem to be morphological negations, not root words. Perhaps they are derived from the stemming and lemmatizing steps that occur during preprocessing of the data. In any case, the poor overlap that they have with the 'most popular words' group for the negative class suggests that grammatical analysis of negative samples is more important for their classification than the simple presence of 'negative' buzz words. This makes intuitive sense. When people are happy with something, they tend to express it short and sweet, but if they are dissatisfied, they will explain precisely what they found to be dissatisfactory (requires higher levels of grammatical and semantic parsing).

7 Conclusion

Overall, the Multinomial Naive Bayes classifier with unigram and bigram features and the SVM classifier with Bigram, TFIDF, and Review Length features have the highest classification accuracy. When we run these classifiers on a 1.5 million sample data set from Twitter, however, only the MultinomialNB fit and classified the data decently. The SVM could not finish the task within our time constraints. SVM?s reliance on grid search for hyperparametertuning and gradient descent for hyperplane fitting cause it to scale poorly with the size of the underlying data.

8 References

[1] Murphy, K. Machine Learning: A Probabilistic Perspective. MIT, in press. (MLAPP)

[2] Farooq, Umar & Mansoor, Hasan & Nongaillard, Antoine & Ouzrout, Yacine & Qadir, Muhammad Abdul. (2016). Negation Handling in Sentiment Analysis at Sentence Level. Journal of Computers. 12. 470-478.

[3] Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2 (ACL '12), Vol. 2. Association for Computational Linguistics, Stroudsburg, PA, USA, 90-94.

Acknowledgments

All coding was done in Python Jupyter Notebook.

All classification was executed using canonical packages from Sci-Kit Learn.

Our preprocessing code was adapted from the provided preprocessSentences.py file.