Predicting Sentiment of Fine Food Reviews on Amazon

Stephen Jaoudi - CS 63 Artificial Intelligence Final Project

Swarthmore College Computer Science Department

Overview

A supervised learning prediction model is built to classify the polarity of reviews of fine foods on Amazon.com. Feature extractors convert each review's raw text into a format appropriate for classification algorithms. Tf-idf weighting is applied to the feature vectors to give higher influence to more informative words. Several general-purpose supervised learning algorithms are applied to the data and their performances are analyzed.

Count Vectorization

Feature extractors are required in the preprocessing step for converting the collected data into a numerical format to be input to classification algorithms. For text classification, we will use feature extractors that transform a varied-length block of text into a numerical feature vector of fixed size. Each entry in the vector is the count of a particular word:

This is the first document
This is the second second document
And the third one
Is this the first document?

$$\Downarrow$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

This representation of text suffices for classification, yet the result of representing a word solely on its occurrence is that:

- Every word carries the same significance
- Words that are frequent are not necessarily more meaningful
- Uninformative and common words such as "I", "the", "is", "am" overshadow the less frequent yet perhaps more informative words

Tf-idf Weighting

Additional efforts are taken to create feature vectors that preserve the interesting and informative aspects of the original text. One measure of a word's importance is a **term-frequency inverse document frequency weight** (tf-idf):

$$\text{tf-idf}(t,d) = \text{tf}(t,d)\,\text{idf}(t,d) \tag{1}$$

- The term-frequency $\text{tf}(t,d)$ measures how frequently a term appears in a document
- The inverse document frequency $\text{idf}(t,d)$ encodes a word's significance by scaling down frequent words while scaling up less frequent words:

$$\text{idf}(t,d) = log(\tfrac{n_d}{1+\text{df}(t,d)}) \tag{2}$$

After this process, the feature vectors now give more influence to words that are more relevant and useful for this classification task.

Algorithm Performance

Algorithm	Accuracy(%)	Train (sec)
LinearSVC	90.7	5.5
Passive Aggressive	89.0	2.2
Ridge	90.8	2.3
Bernoulli Naive Bayes	84.7	0.1
Logistic Regression	87.3	49.2
Perceptron	87.7	1.9
SGD	88.9	4.3
Random Forest	84.1	1.4

Table 2: Classification Algorithm Performances

Most algorithms applied to the data found reasonable success at this classification task, although some emerged as clear advantages in overall accuracy and training time. The "LinearSVC", a support vector machine with linear kernel, and the Ridge Classifier had the highest accuracies.

Confusion Matrices

		Predicted Positive	*Predicted* Negative
LinearSVC	*Actual* Positive	0.912	0.088
	Actual Negative	0.101	0.899
Passive Aggressive	*Actual* Positive	0.897	0.103
	Actual Negative	0.115	0.885
Ridge	*Actual* Positive	0.910	0.090
	Actual Negative	0.103	0.900
Bernoulli Naive Bayes	*Actual* Positive	0.791	0.209
	Actual Negative	0.098	0.902
Logistic Regression	*Actual* Positive	0.894	0.106
	Actual Negative	0.122	0.878
Perceptron	*Actual* Positive	0.879	0.121
	Actual Negative	0.122	0.878
SGD	*Actual* Positive	0.895	0.105
	Actual Negative	0.120	0.880
Random Forest	*Actual* Positive	0.854	0.146
	Actual Negative	0.171	0.829

Table 1: The Confusion Matrix of Each Algorithm

Calculating the confusion matrices for each algorithm presents a more detailed description of their strengths and weaknesses. Each cell of a confusion matrix contains the rate of true and false positives and negatives.

A successful algorithm is one that not only has a high overall accuracy, yet also has low rates of false positives and negatives. Some algorithms had equivalent accuracies regardless of a review's positive or negative polarity, while others had higher rates of correctly classifying positive reviews in particular.

The general trend among these matrices is that the rate of true positives is higher than that of false negatives. This implies that the task of correctly classifying a review that is positive is inherently easier than correctly classifying a negative review.

Conclusion

Supervised learning algorithms from the linear classifier family are the highest performing algorithms for this classification task, with the LinearSVC and the Ridge Classifier having the overall best performance.

Linear classification algorithms also find strong results in other classification problems involving text processing. The output of the feature extraction process are very sparse, high dimensional feature vectors. Algorithms such as the Naive Bayes or the Random Forest Classifier are not as strong at generalizing high-dimensional data, and have proven to be less accurate for this task. High dimensional data are likely to be linearly separable, meaning this classification problem is best suited for algorithms like the LinearSVC and the Ridge Classifier. These algorithms are able to linearly separate high-dimensional data, while avoiding over-fitting by maximizing decision boundaries.

References

[1] Jure Leskovec and Andrej Krevl.
SNAP Datasets: Stanford large network dataset collection.

`http://snap.stanford.edu/data`, June 2014.

[2] 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.
Journal of Machine Learning Research, 12:2825–2830, 2011.

Acknowledgements

Many thanks to Bryce Wiedenbeck for his help with this project, and to the Stanford large network dataset collection (SNAP) for providing this dataset.