Predicting Rating with Sentiment Analysis

Jason Jong

December 16, 2011

Abstract

From recent trends, many online reviews include a numerical or star rating that quantifies the satisfaction of the reviewer's experience. However, an objective mapping of this quantitative rating to the reviewer's textual description does not yet exist. In this paper, we explore models ranging from support vector machines to learning word vectors that capture the sentiment information of individual words in relation to ratings of an entire document. We use Yelp reviews for restaurants near particular universities to predict corresponding star ratings per review.

1. Introduction

More than ever before, people's judgments of what to do, or what to eat, are governed by the opinions of other people. The internet has become the ultimate trove of the opinions of many, many people. Today, sites like Yelp have become a vast database for places and restaurants that include reviews and opinions written by everyday people. This crowdsourcing method of extracting satisfaction has currently been a successful model for providing accurate predictions of one's experience for a certain service.

However, Yelp's qualitative reviews suffer from the fact that their quantitative star ratings rarely provide the most objective or the fairest rating. In fact, most of the reviews on Yelp are skewed towards higher ratings. And most of the stars are virtually meaningless given that most range from 3.5 to 4.5 stars, with very few below or above. Distinguishing these restaurants, and giving them a proper ranking system, becomes a major challenge.

We seek to turn words and reviews into quantitative measurements. By extracting satisfaction from this feature, a text review will hold more quantitative value than a star rating. Once summed together, a restaurant's list of text reviews will give a rich quantitative measurement of the service's satisfaction rating.

2. Yelp Reviews Dataset

We use Yelp's recently released academic dataset, which provides over one hundred fifty thousand reviews and their corresponding ratings for restaurants centered near many different universities. The data also includes other information collected by Yelp, including "useful," "funny," and "cool" ratings that is specific to each review.

For the sake of training, we did not consider correspondences between the

users or the restaurants with reviews because we wanted to provide a purely objective analysis of semantic information. We treated each review as its separate document and assumed each document corresponded to a single review and its single star rating.

Unfortunately, the star ratings were just as vague as the star ratings on Yelp's reviews. Each rating was rounded to a whole star, so the set of ratings only included 1, 2, 3, 4, and 5.

The average rating was about 3.6. Objectively, we decided to place all ratings above this value into the pool of "positive" sentiment responses, and all the ratings below this value into the "negative" sentiment response.

3. Comparison of Models

To capture the sentiments of our reviews, we will model our data after three different learning algorithms. First, we implement a Naïve-Bayes Classifier, a model that analyzes the Bayesian probability of each word occurring within each model. Next, we implement a support vector machine, a model well known in the realm of textual analysis. Our last algorithm, from the paper Learning Word Vectors for Sentiment Analysis, relies first on an unsupervised learning algorithm to capture word semantics [1]. It then uses supervised learning to capture word sentiments.

3.1 Naïve-Bayes Classifier

We implement the multinomial event model with Laplace smoothing. This relatively simple implementation from the well-known Naïve-Bayes models provides a solid beginning for our semantic analysis.

First, we split the entire pool of documents into their respective ranking so the occurrence of certain words for a certain star rating goes into the same bin. After training every single word and its likely occurrence in either a negative or positive rating, we can test the resulting words of an anonymous review for its likely star rating.

This implementation is relatively basic compared to other implementations because it only measures the likely occurrence of certain words within a certain classification. For instance, it completely disregards grammar and word order, and it does not give a proper measure of the likelihood of two word similarities.

3.2 Support Vector Machines

Next, we implement a support vector machine (SVM) that uses a linear kernel. There is considerable belief that support vector machines provide one of the best models for predicting textual information.

For instance, SVM's provide strong responses to high-dimensional input spaces, which is the case with text analysis. Also, SVM's deal well with the fact that document vectors are sparse [2]

3.3 Learning Word Vectors for Sentiment Analysis: Theoretical Background (Maas et al., 2011)

Lastly, we will implement a learning algorithm that is considerably more advanced that our other implementations. This involves two processes: capturing semantic

similarities and modeling after word sentiment.

3.3.1 Capturing Word Semantics

We will first capture semantic similarities between words [1]. This is done by relying on a continuous mixture distribution over words indexes by a multi-dimensional random variable theta that is specific to each document. We assume that the probability distribution will be conditionally independent with respect to theta. Thus, the probability of the document is:

$$p(d) = \int p(d, \theta) d\theta = \int p(\theta) \prod_{i=1}^{N} p(w_i|\theta) d\theta$$

To model each word with respect to the document variable theta, we will use a softmax formula:

$$p(w|\theta; R, b) = \frac{\exp(\theta^T \phi_w + b_w)}{\sum_{w' \in V} \exp(\theta^T \phi_{w'} + b_{w'})}$$

Simplifying, and taking the log of the following, we will get:

$$\max_{R,b} p(D; R, b) = \prod_{d_k \in D} \int p(\theta) \prod_{i=1}^{N_k} p(w_i|\theta; R, b) d\theta$$

3.3.2 Capturing Word Sentiments

To capture the polarity of sentiment values, another term in our objective function will capture the sentiments of individual words within a document. Each word vector receives a sentiment labeling with the following equation:

$$p(s = 1|w; R, \psi) = \sigma(\psi^T \phi_w + b_c)$$

With this equation, you can capture the sentiment of an entire document:

$$\max_{R,\psi,b_c} \sum_{k=1}^{|D|} \sum_{i=1}^{N_k} \log p(s_k|w_i; R, \psi, b_c)$$

The sentiment analysis part of this model is a supervised learning algorithm that relies on the star ratings of past reviews to capture the word vectors representations.

Finally, adding the two objective functions together gives a whole objective function for maximization.

4. Implementation

4.1 Details and Design Decisions

We objectively chose to level each rating at the 3.5 level divide, such that star ratings above this level would be marked as "1," and star ratings below this level would be marked as "0." With our choice, we could demarcate at a level that closely marked the median of the data. However, in the end, there were still slightly more high-rated stars than low-rated stars, but this would not skew the data given our chosen weightings.

We were not able to directly predict star ratings because most current models rely on the choice between "true" and "false" values. We generate our error responses based on this denomination.

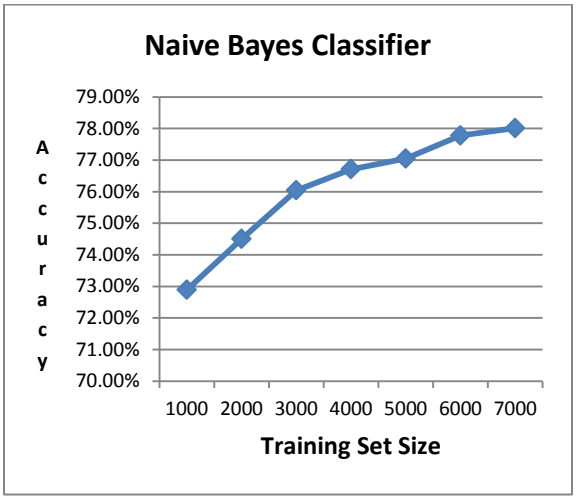
Following our training of seven different training set sizes, we tested our data on the same data set size of twenty thousand unique reviews.

4.2 Naïve Bayes Classifier

Despite an absence of distinguished semantics, Naïve Bayes ended up performing the best of the three implementations. This may be the cause of several reasons. For instance, Naïve Bayes may perform better with smaller datasets, but deprecate in performance as data values grow.

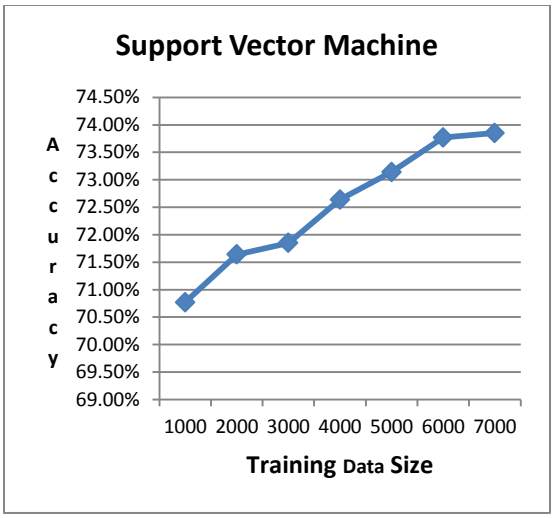
From the graph below, Naïve Bayes makes a significant jump towards the beginning of the data set, but begins to level off, almost proportional to the amount of data available.

However, Naïve Bayes still proves to be a significant classifier despite ignoring relative semantic and sentiment features. It has proven in many cases to provide a substantial model for probabilistic text analysis.



4.3 Support Vector Machine

To train our support vector machine model, we used NTU's Liblinear SVM library, which uses a linear kernel to build the model. Surprisingly, the model does not perform better than Naïve Bayes Classifiers, but this may be because of the smaller dataset we had to deal with.

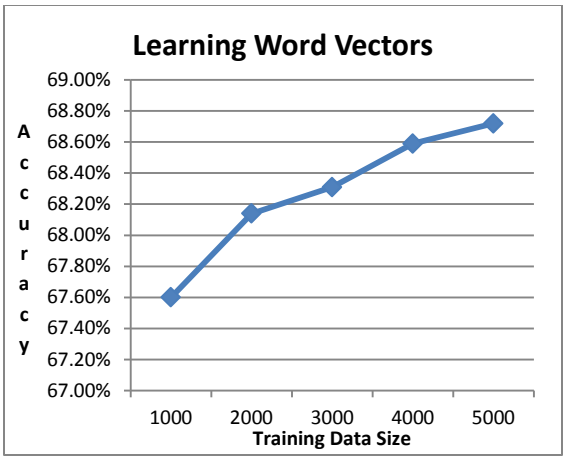


4.4 Word Vector Learning with Sentiment Analysis

Lastly, we implement the most significant model of our project that involved sentiment analysis.
In order to implement this model, we relied on the CVX Matlab Library for Convex Programming for the continuous optimization of this learning algorithm.

To optimize, we use coordinate ascent because the word representation variables (R, b, and psi) were non-convex with respect to the theta values of each document.

The program worked effectively, but provided very, very slow responses. Either way, the results did not prove to be better than SVM or Naïve Bayes.



4.5 Overall Results

The results proved to be much better than expected. For opinionated texts, there is usually a 70% agreement between human raters. Thus, our models, which have accuracies around and above 70%, provide a very strong model for sentiment analysis.
Still, the future of opinion mining will vary based specifically on the growth of sentiment information from the last model because, only then will machine learning have a stronger objective grasp of the sentiments of a document.

5. Future Work

In our experiment, we mapped the star ratings down to simplified 1 and 0 values, to signify a sharp polarity between positive and negative reviews. Initially, we had hoped to work towards a model that allowed us to make an entirely quantitative star rating measure of a review. We can design our model to better capture this information. We can also include more data by factoring in the three ratings per review provided by Yelp.

6. Reference

[1] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng, and C. Potts. 2011. Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

[2] T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the European Conference of Machine Learning (ECML).*

[3] R-E Fan, K –W. Chang, C.-j. Hsieh, X.-R. Wang, and C.J. Lin. 2008. Liblinear: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 1871-1874.

[4] R. Socher, A. Maas, and C. Manning. 2011. Spectral Chinese Restaurant Processes: Nonparametric Clustering Based on Similarities. *AISTATS 2011*.