CS506 Kaggle Amazon Challenge
Chenyang Yu

Introduction

Users comment on businesses, products and services through reviews consisting of free-form text and a numeric star rating, from 1 to 5. User reviews become a crucial predictor for the ratings for web services like Amazon.

In this project, I used the Amazon review data for text analytics to predict the score. We are given the set S = {(r1, s1), ...,(rN , sN)} for a product P, where ri is the i'th user's text review of P and Si is the i'th user's numeric rating for P, the goal is to learn the best mapping from a word vector r to a numeric rating Si.

This project combined two feature extraction methods with each of four supervised learning algorithms. This report describes preprocessing procedures, feature extraction methods, the supervised learning algorithms, and performance evaluation Metrics.

Preprocessing

First, I extracted the reviews, score and Id from a range of variables and create a new dataframe. Then, I want to calculate term frequencies after tokenizing the reviews. The first tokenizing operations is to tokenize letters, ignoring special symbols and numbers through Snowball stemmer. Snowball stemmer returns the root of a word. Stop words are also removed in the vectorization step. Term frequencies counts the term. I've stored the tokenized letters and term frequencies into sklearn.externals.joblib for easier access.

To computer clusters, I've implemented K Means and LDA. K means method put reviews into one category, but LDA allows reviews to be included in multiple categories. I choose 5 categories and compare the top 10 stem words for those categories.

Feature Extraction

Because the dimension of the dataset is too large, I perform a Singular Value Decomposition to reduce the dimensionality of the matrix and to extract features.

Latent Semantic Indexing (LSI) and Singular Value Decomposition (SVD)

In Latent Semantic Indexing, I first construct a word-review matrix M, of size m × t, using the unigrams model, and then do Singular Value Decomposition of M.

Due to our large dataset, so it's more efficient to use TruncatedSVD method in the scikit-learn package instead of svds from scipy.sparse.linalg package . I've also performed randomized_svd method. It has a different result is from TruncatedSVD. I've decided to use the result from TruncatedSVD to transform the dataset into a representation using the top 30 components, thus preserving variance in the data. This has a similar effect to Principal Component Analysis where I represent the original data using an orthogonal set of axes rotated and aligned to the variance in the dataset.

SVD method preserves a lot of variance but is still too large to visualize. I was intended to implement t-SNE to reduce the dimension of our dataset for visualization. However, due to high dimensionality of the dataset, it takes too long to run.

Supervised Learning

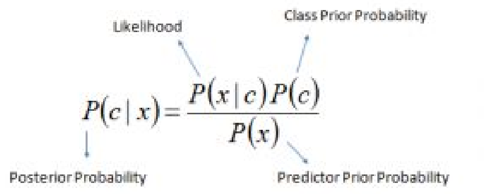
To train our prediction models, I use four supervised learning algorithms.

Logistic Regression

The conditional probability function P(s|r) is modelled, where r is a feature vector for review r and s belongs to the set of class labels {1, 2, 3, 4, 5}. Then, given a new feature vector r ∗ for a new review r ∗ , this probability function is computed for all values of s, and the s value corresponding to the highest probability is output as the final score label for this review.

Naive Bayes Classification

Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c) in the equation below:


Likelihood
Class Prior Probability
$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Support Vector Machines (SVM)

A Support Vector Machine is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. In this case, I use linear SVMs for multiclass classification. It's best used to separate classes. For each feature extraction method, I do internal 10-fold cross validation.

Performance Metrics & Implementation

I use 70% of the train dataset for training, and 30% for testing. I perform 10-fold cross validation on testing dataset and compute Mean Squared Error and Root Mean Squared Error (RMSE).

The programming language used is Python, and extensive use is made of its libraries numpy, scipy and scikit-learn.

Results and Analysis

RMSE was used to select the optimal model using the smallest value. By comparing RMSE score for four learning algorithms, I chose SVM model. Although logistic regression has the lowest score, we should not use RMSE as a metrics for logistic regression.

Sources

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72