Search Engine Technology, Fall 2013, Homework 2
Report on 'Sentiment Analysis of Movie Reviews'

Rupayan Basu (rb3034)

Kaggle ID: Rupayan

1. Overview

   The overall approach used here is to treat words in the reviews as features for training a Naive Bayes Classifier. During training, conditional probabilities of the form p(word|label) are computed for every word seen in the training corpus. Laplace smoothing is used for unseen words. Prior probabilities for each label, p(label) are also computed.

   During testing, for each review in the test set, a score is computed for the positive/negative labels using Bayes rules and the naive independence assumption. The label with the higher score is selected for that review.

2. Design

   The implementation consists of 3 files:

   2.1 NBC.py - contains the class NaiveBayesClassifier which encapsulates all the attributes and methods required to preprocess training data, extract and select features, train, and test the classification model.

   Some of the important attributes and methods are described below:

   Attributes required for chi square:
   DocsCount - total document count in training dataset
   labelCount - stores document counts for each label
   jointCounts - stores document counts for each token/label pair

   Attributes required for NBC:
   labelPrior - stores prior probabilities for each label
   totalJointCount - stores total counts for each token/label pair
   condProb - stores conditional probabilities for every token/label
   sumTokenCounts - stores the sum of all cond probs for each label

   Methods:
   2.1.1. preprocess - performs lowercasing, and based on boolean switches - stemming, lemmatization, and removal of stop words and

small words.

2.1.2. preprocessTrain - collects counts required for feature selection and training the classifier.

2.1.3. chiSquareFeatureSelection - calculates chi square score for every word in the vocabulary and removes those words for which the independence assumption cannot be rejected with some confidence.

2.1.4. TrainNBClassifier - calculates all the prior and conditional probabilities necessary for the Naive Bayes Classifier.
It also produces *NaiveBayesProbs.dat* which is a human-readable representation of the trained model containing all the conditional probabilities.

2.1.5. classify - given a review, calculates a score for every label and returns the label with the higest score.

2.1.6. testNBClassifier - iterates over all test reviews and writes results to the prediction file.

2.2. train.py - takes training file and model file as command line arguments and produces a trained and serialized classification model.

2.3. test.py - takes the model file, test set and prediction file as command line arguments and writes <id,label> pairs for each test review in the prediction file.

3. Implementation details

Language used: Python 2.7.3
External libraries: NLTK (for preprocessing and tokenization only)
Modules: math, pickle

3.1. Preprocessing

All words are lower cased. Stemming, Lemmatization and removal of stopwords can be turned on or off by passing True/False to preprocessTrain() method.

The NLTK wordnet corpus is used for lemmatization and the stopword corpus is used for identifying stopwords in English.

3.2. Feature Extraction

Every review in the training set is tokenized to produce tokens that are then used as features for training the classification model.

Tokenization is used instead of splitting on white spaces because in the latter case, sometimes words get paired with punctuation marks.
Example: ‘Hi, my name is Rupayan.’

Splitting on white space produces:
[‘**Hi,**’, ‘my’,’name’,’is’,’**Rupayan.**’]

3.3. Feature Selection

The chi square method is used to select significant features. The chi square score is computed for every token in the vocabulary and those that are less than or equal to a threshold value are discarded, since the independence assumption for these tokens cannot be rejected with confidence.
Example:
If the chi square score for a word <= 6.63
then we cannot say with 99% confidence that this word is not independent of the class, and hence it is rejected.

3.4. Classifier Implementation

The classifier used is the Naive Bayes Classifier.
--Advantages:
It is simple to implement and is known to produce relatively accurate results for sentiment analysis.
--Disadvantages:
It ignores context and places all the emphasis on individual words.
Example:
“It could have been a great movie.”
This has a negative connotation, but the presence of the word “great” will give it a higher probability of being classified as a positive review by Naive Bayes.

During training, the TrainNBClassifier() calculates all prior and conditional probabilities, which were used by classify() during testing to calculate a score for each review and each label. The label with the higher score was returned for that review.
The output file NaiveBayesProbs.dat contains a list of all the conditional probabilities generated by the model.
After training, the model is pickled and later used by test.py

4. References

‘An Introduction to Information Retrieval’ (Manning, Raghavan, Schutze)