STONY BROOK UNIVERSITY

CSE628: Homework 1

Kavita Agarwal Sonal Aggarwal

March 15, 2014

1 IMPLEMENT A LANGUAGE MODEL CLASSIFIER WITH LAPLACE SMOOTHING

We have build a language model classifier on a movie review dataset. The dataset [1] comprises of two classes namely, positive and negative each containing 1000 text files. We performed a five fold cross validation on the dataset with even distribution of the data files.

The features we used were bigrams and unigrams and then the classification accuracy was measured separately on the feature frequency and feature presence in the files. To deal with the unknowns we used Laplace Smoothing that gives the same probability value to all the unknowns.

To calculate the probability of a unigram belonging to a particular class (positive or negative) using feature presence applied with smoothing:

$$P(w_1) = \frac{Count\ of\ files\ belonging\ to\ a\ class\ containing\ the\ feature + 1}{Total\ files\ in\ the\ class + Vocabulary\ Size} \tag{1.1}$$

Similarly for feature frequency we used:

$$P(w_1) = \frac{Total\ count\ of\ the\ feature\ in\ a\ class + 1}{Total\ feature\ count\ in\ the\ class + Vocabulary\ Size} \tag{1.2}$$

To avoid underflow we use log of probabiliies which also provided the added benefit of being faster.

Table 1.1 summarizes the classification success rate for the positve and negative classification of the reviews using unigrams as features and compares the results of Unigram presence

[1] http://www.cs.cornell.edu/people/pabo/- movie-review-data

vs Unigram Count.

	Unigram Presence			Unigram Count		
	Pos	Neg	Avg	Pos	Neg	Avg
1st fold	59.0	92.0	75.5	77.5	83.0	80.25
2nd fold	73.0	89.0	81.0	79.0	83.0	81.0
3rd fold	50.5	95.5	73.0	77.5	84.5	81.0
4th fold	62.0	95.5	78.75	80.5	85.0	82.75
5th fold	60.5	93.5	77.0	74.5	84.5	79.5
Average success rate	77.05			81.3		

Table 1.1: Classification results using unigrams in %

Table 1.2 summarizes the success rate for the positve and negative classification of the reviews using bigrams.

	Bigram Presence			Bigram Count		
	Pos	Neg	Avg	Pos	Neg	Avg
1st fold	98.0	44.5	71.25	98.0	46.0	72.0
2nd fold	96.0	50.0	73.0	95.5	51.5	73.5
3rd fold	99.0	51.5	75.25	96.5	53.0	74.75
4th fold	96.5	51.5	74.0	99.0	52.0	75.5
5th fold	99.0	51.5	74.25	96.5	52.0	74.25
Average success rate	73.55			74		

Table 1.2: Classification results using bigrams in %

Conclusion: The results demonstrate the superiority of unigrams in movie review classification. One plausible explanation can be the high specifity of using bigrams which do not give good results on short documents. Also for handling unknowns theoretically unigrams should fare better. Also bigrams give better positive class classification than for negative class wheras for unigrams we get slighly better results for negative class classification for both feature frequency and feature count.

To play with the feature set we can also select a unigram if it occurs more than 3 times in a file to reduce the size of the feature set.

2 Building a Perceptron Classifier

The second problem was to build a Perceptron Classifier on the same movie review data set. Our approach towards it is listed below:

- **Weight vector learning:-** Initialized the weight vector to zero. The weight vector is learned from feature vector of the training data set. The correction rate is the difference of expected class and the dot product of feature and weight vector. This correction rate is multiplied with the feature vector and added to the weight vector

$$Score = WeightVector * FeatureVector \tag{2.1}$$

$$Correction = ExpectedClass - \frac{Score}{Total\ no\ of\ words\ in\ feature\ vector} \tag{2.2}$$

$$WeightVector = WeightVector + Correction \tag{2.3}$$

- **Text Categorization:-** The learned weight vector is used to classify the new test data. The dot product of Test data feature set and weight vector gives us the classification. If the classification result is positive the test data set belongs to 'pos' class else 'neg' class. This result is gathered for five fold cross validation and average is computed

We used the feature presence for this classification. The feature set used were unigrams and bigrams.

	Unigram Presence		
	Pos	Neg	Avg
1st fold	81.0	89.0	85.0
2nd fold	86.0	86.0	86.0
3rd fold	83.0	89.0	86.0
4th fold	87.0	90.5	88.75
5th fold	82.5	89.5	86.0
Average success rate	86.35		

Table 2.1: Perceptron classification results using unigram presence in %

	Bigram Presence		
	Pos	Neg	Avg
1st fold	83.5	78.0	80.75
2nd fold	86.5	73.5	80.0
3rd fold	83.0	77.0	80.0
4th fold	90.0	70.0	80.0
5th fold	89.5	80.5	85.0
Average success rate	81.15		

Table 2.2: Perceptron classification results using bigram presence in %

Conclusion: We used Unigram Presence and bigram presence features. We ran this classifier for different learning rates 1 and 0.7 and iterations of 1,2 and 10 but did not observe any significant difference in the performance. On the whole unigram surpasses bigram with average performance of 86.35 and 81.15 respectively. Here again unigrams were giving better performance as compared to bigrams.

3 CLASSIFICATION USING A PRE-EXISTING SVM CLASSIFIER

This involved classifying the data set using a SVM classifier which provided a good tool to compare with the results of our language model classifier. We used the LibSVM [2] tool, its Java interface and command line utility for classification.

The approach we followed was:

- We converted the training data and test data into LibSVM format required by the SVM classifier by using Unigram presence and Unigram count as feature vectors
- Two approaches were followed to run this LIBSVM classifie
- LIBSVM utility through command prompt:-
  - We ran the train_data_file generated above on the svn-train.exe which generates a model file
  - The test_file was classified using svm-predict.exe and the model generated above
- LibSVM Java based interface
  - The test_data and training_data files as per the required format were run on this

The results for unigram presence are summarized in 3.1.

	Classification Accuracy
`1st fold`	86.0
`2nd fold`	85.5
`3rd fold`	83.5
`4th fold`	84.75
`5th fold`	83.5
`Average success rate`	**84.65**

Table 3.1: SVM classification results using unigram presence in %

The results for unigram frequency are summarized in 3.2.

[2] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.

	Classification Accuracy
1st fold	84.5
2nd fold	79.25
3rd fold	82.25
4th fold	84.5
5th fold	85.75
Average success rate	**83.25**

Table 3.2: SVM classification results using unigram count in %

CONCLUSION:

- The classifier was run for different cost factors 20, 30, 50,100 and 110 but the performance over different runs remained almost same over the two approaches.
- LibSVM utility through command prompt:-
  - Unigram Presence gave an average performance of 67%
  - Unigram Count gave an average performance of 60%
  - One reason which we think attributes to low performance in this case is the windows DLL required by the LibSVM command line utility which might be influencing the classifier because of some inner architecture based dependencies of Windows.
- LIBSVM Java based interface :-
  - When the same files were run in java (which does not require any platform related DLLs) the performance shoots up to 86%
- We could not set the regularization parameters for LibSVM classifier.

Error Analysis:

We found the common files which were wrongly classified by the perceptron classifier and the language model classifer. Then we saw the corresponding training data set for both of them and came to the conclusion that the files which belong to positive class and are wrongly classified as negative class have words in training data set which belong to negative class and vice versa.

Filename	Actual Class	Classification
\cv034_29647.txt	positive	negative
\cv040_8276.txt	positive	negative
\cv060_11754.txt	negative	positive
\cv104_19176.txt	negative	positive

Table 3.3: Wrong classification of files

Inference:

Given the success rates of the the classifiers used by us, we obtained the best classification accuracy by the **Perceptron Classifier** using unigram presence.

The reason for this might be the alternative class learning approach we followed while learning the weight vector instead of learning one entire class at a time.