Learning from Data - Week 1

Assignment 1

Mart Busger op Vollenbroek
S2174634

Script explanation

The script used in this assignment (and report) is LFDassigment1_S2174634.py. The script has two functions: read_corpus() and identity(). Read_corpus(), like its name suggests, reads in the corpus used for the assignment, identity() is dummy function. Based on the first argument given, either a binary classification or a multinomial classification is used. The binary classification consists of pos (for positive) or neg (for negative), the multinomial classification consists of books, camera, dvd, health, music, software. The function returns the document and its corresponding label. After reading in the corpus, the data is splitted at 75% which means the classifier will train on 75% of the data and test on the remaining 25%. Then after initializing the classifier, it is trained on the train data and predicts the classes in the test data. Finally the scripts outputs the precision, recall and f-score results from the classifier, the overall accuracy, the posterior probabilities and the label counts used for the prior probabilities.

Report questions

1. Why should one not look at the test set?

One should not look at the test set because the data in the test is used for testing your system on **unseen** data. This is because a system is developed to also be able to handle data outside of the corpus used for training and testing. Also, if you would look at the testdata, you would be able to modify your program so that it scores better on the test data which would give a distorted version of reality.

2. What happens with cross-validation?

Using cross-validation, the data in the corpus is splitted a certain amount of times so that the classifier is trained and tested on all of the data. After running the classifier the amount of times specified before (using a 10-fold cross-validation the classifier is run 10 times), an average can be calculated which gives a more real estimation on how your system is performing.

3. What baselines could you use for the binary sentiment classification and for the six- class topic classification? What would be their performance?

For the binary classification in positive and negative you could use a 50% baseline, because by not looking at the data there is a fifty-fifty percent change that a document is positive or negative. After looking at the counts for the documents however, it is seen that the prior probability for positive tweets is higher (6000 / (6000 + 5914) = 50,4%) which implies that the baseline for the binary classification should be higher (e.g. 50,4%).

For the multinomial classification there are six possible classifications using the same method as above, you could assume that the baseline is 1/6 = 16,7%. After again looking at the prior probabilities however, this baseline is slightly increased to 16,8% (2000 / (2000 + 2000 + 2000 + 2000 + 1999 + 1915).

If these baselines would be used, both systems would score very high with accuracy scores of 82,8% for the binary classification and 89,3% for the multinomial classification.

4. What features are used in the classification task you ran?

The features used in the classification are all the words in the de documents. This is also called the Bag of Words method.

System output

```
             precision    recall  f1-score   support

        neg       0.84      0.82      0.83      1540
        pos       0.81      0.84      0.82      1439

avg / total       0.83      0.83      0.83      2979

Overal accuracy: 0.828130245049
Posterior probabilities:
[[ 0.52808227  0.47191773]
 [ 0.28056395  0.71943605]
 [ 0.73489084  0.26510916]
 ...,
 [ 0.32242054  0.67757946]
 [ 0.58021586  0.41978414]
 [ 0.78035644  0.21964356]]
Counter({'pos': 6000, 'neg': 5914})

             precision    recall  f1-score   support

      books       0.91      0.91      0.91       490
     camera       0.76      0.98      0.86       474
        dvd       0.91      0.89      0.90       511
     health       0.97      0.73      0.83       530
      music       0.94      0.93      0.94       525
   software       0.90      0.93      0.91       449

avg / total       0.90      0.89      0.89      2979

Overal accuracy: 0.892917086271
Posterior probabilities:
[[ 0.14680012  0.1152495   0.08634145  0.47392747  0.04433552
0.13334593]
 [ 0.15424524  0.01382021  0.72419274  0.01343912  0.07744528
0.01685741]
 [ 0.00690135  0.03109302  0.00817811  0.00983671  0.00332952
0.94066129]
 ...,
 [ 0.01650236  0.84688663  0.01536382  0.04611711  0.01075017
0.06437991]
 [ 0.12283968  0.12994681  0.10203327  0.51339297  0.07919294
0.05259433]
 [ 0.05237549  0.41904252  0.09962694  0.19470154  0.06844157
0.16581195]]
Counter({'books': 2000, 'health': 2000, 'music': 2000, 'dvd': 2000,
'camera': 1999, 'software': 1915})
```