Artificial Intelligence Practical, As part of the Intelligent Interaction module

Building a bayesian classifier, testing it and comparing it to different existing classifiers using Weka

Group 10B:

Joep Schyns (s1472860)
Willem Siers (s1424661)
Elmar Peters (s1097717)
Carmen Burghardt (s1480782)
Huub van Wieren (s1475339)

Table of Contents

Part A: Building the Bayesian Classifier

Implementation

We built a naive bayesian classifier using Java and by implementing the algorithm described in the reader. The technique used for the tokenizer is to convert all text to lowercase and then simply split all text by all characters that are not letters or apostrophes. To make the classifier learn it can load text files or accept manual additions of text through it's methods. The test used will load in a specified set of files and classify them. The final result of the test will be the fraction of correctly classified documents, where "correct" is given by that it classified the document the same as an "expected" class.

Testing the implementation

The classifier was first trained using the blogstrain test set, classifing all entries in this set in the class as specified by the testset's subfolder (m or f). Then the following results were found:

For K = 1 the accuracy on the female blogs test set was 88% and on the male test set it was 40%, so the average performance of the classifier for the two test sets was 64% for K = 1.

When K = 0.0001 it performed 80% on the female test set and 52% on the male test set, which results in a performance of 66%.

So for a low value of K it would perform better, as it at least reaches the 50% mark on the male test set accuracy.

Part B: Testing Bayes Classifier in Weka

Testing

We used cross-validation test (10 folds) to test the naive bayesian classifier of Weka.

Results

<table>
<tr><th></th><th>Blogs train</th><th>Blogs test</th><th>Spam mail</th></tr>
<tr><td>Confusion Matrix</td><td>
<table>
<tr><th>A (Female)</th><th>B (Male)</th></tr>
<tr><td>218</td><td>82</td></tr>
<tr><td>113</td><td>187</td></tr>
</table>
</td><td>
<table>
<tr><th>A (Female)</th><th>B (Male)</th></tr>
<tr><td>22</td><td>3</td></tr>
<tr><td>2</td><td>23</td></tr>
</table>
</td><td>
<table>
<tr><th>A (Ham)</th><th>B (Spam)</th></tr>
<tr><td>2393</td><td>19</td></tr>
<tr><td>102</td><td>379</td></tr>
</table>
</td></tr>
<tr><td>Summary</td><td>Accuracy 67.5%
Recall 0.674
Precision 0.677
F-Measure 0.73</td><td>Accuracy 90%
Recall 0.9
Precision 0.0.901
F-Measure 0.9</td><td>Accuracy 95.8175%
Recall 0.958
Precision 0.958
F-Measure 0.957</td></tr>
</table>

Blogs train and Blogs test

We tried altering termMinFreq (tested with 3, 5, and 15), but this did not change the accuracy of the classifier. We also tried NormalizeDocLength set to “all data” but this decreased the performance to 66.83% Lowercase tokens set to true will achieve a performance of 67.3333%, which (surprisingly) is lower than without the lowercase tokens. Changing wordsToKeep to 750 words will decrease the performance to 65% accuracy. Not surprisingly, increasing the wordsToKeep to 1250 words will improve the performance to 68.5%. Setting it much higher will not really improve the results (much), at 2000 words it will have a performance of 68.6667%.

Changing the settings of the preprocessor the same way as with the Blogstrain set will give the same differences in performances, and from the tested features only increasing the wordsToKeep to a higher value can actually increase it. With a value of 2000 it scores 94% accuracy on the test.

Spam mail

This set has a very high performance (95.8175%) on the default StringToWordVectorFilter settings and a low (squared) error. Increasing wordsToKeep to 2000 results in an accuracy of 95.8521%. On the spammail set using lowercase tokens will actually improve the

performance slightly at 95.8521%. This was the only set on which changing this feature had effect.

Part C: Logistic Regression and Decision Tree classifiers

I first applied the StringToWordVector filter to the spammail data set and then used the weka.classifiers.functions.Logistic to train and test a logistic regression classifier with Cross-validation of 10 folds. Then I did the same with the weka.classifiers.trees.J48 for training and then testing a decission tree classifier.

The results were the following:

	Accuracy	**Recall**	**Precision**	**F-measure**
Logistic Regression	89.9067 %	0.899	0.914	0.904
Decission Tree	95.2644 %	0.953	0.952	0.952

The confusion matrices after the test are as follows:

Logistic Regression

A	**B**
2199	213
79	402

Decission Tree

A	**B**
2363	49
88	393

(A = ham, B = Spam)

Overal the Decission Tree classifier performed much better as can be seen from the results above. It also seems that although the Decission Tree classifier had a better overal performance, the Logistic Regression classifier seems to find less less false positives (Spam classified as Ham) and more true positives for Spam.