University of Groningen

Artificial Intelligence 2

Lab 2

Siger Steenstra
s2408791

October 5, 2016

1 Testing Stage (3)

(**) For this assignment I have worked together with Davey Schilling.

(3.2)
We compute how many of each type of message is correctly classified, and show this with a confusion matrix. A little piece of the output of our program is listed below. We have played around with the tuning parameter ϵ, and here are some results:

```
Tuning parameter 1:
Correctly regular: 59 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 9 | 74.28571428571429%

Tuning parameter 0.5:
Correctly regular: 64 | Falsely regular: 1 | 98.46153846153847%
Correctly spam: 25 | Falsely spam: 4 | 86.20689655172413%

Tuning parameter 0.25:
Correctly regular: 66 | Falsely regular: 1 | 98.50746268656717%
Correctly spam: 25 | Falsely spam: 2 | 92.5925925925926%

Tuning parameter 0.1:
Correctly regular: 66 | Falsely regular: 1 | 98.50746268656717%
Correctly spam: 25 | Falsely spam: 2 | 92.5925925925926%

Tuning parameter 0.01:
Correctly regular: 67 | Falsely regular: 1 | 98.52941176470588%
Correctly spam: 25 | Falsely spam: 1 | 96.15384615384616%

Tuning parameter 0.001:
Correctly regular: 67 | Falsely regular: 1 | 98.52941176470588%
Correctly spam: 25 | Falsely spam: 1 | 96.15384615384616%

Tuning parameter 0.0001:
Correctly regular: 67 | Falsely regular: 1 | 98.52941176470588%
Correctly spam: 25 | Falsely spam: 1 | 96.15384615384616%
```

When the tuning parameter is set to the default of 1, there are no false negatives, so the score is 100 percent for regular. However, there are a lot of false positives. 9 of the regular messages are incorrectly classified as spam, which means only 86 percent of the regular messages was classified correctly. Another remark is that all spam messages are correctly classified as spam, but we consider it a problem that 9 messages that are regular are now classified as spam. (3.3)
When the training and testing is done on the same set, a score of 100 percent is achieved:

```
Correctly regular: 68 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 0 | 100.0%
```

This is very logical, because the probabilities for word occurence are calculated on the set. When tested on the same set, then all occurence probabilities for the words are exactly correct, as well as the number of regular and spam messages.

2 Bigrams (4)

(a)
For now we have left the ϵ parameter at 0.001, because this gave the best results in the unigram version. We have implemented a bigram version of the Naive Bayes classifier, where at least one of the words of a bigram must have length 4, and the other word has to have a length of at least 3 characters.
When we run the code for bigrams the confusion matrix looks like this:

```
Correctly regular: 63 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 5 | 83.87096774193549%
```

It is good that no spam is classified as regular, but 5 messages that are classified as spam while they are not actually spam is in my opinion a decrease in performance. I believe this has to do with the fact that bigrams have a lower frequency than unigrams, and to achieve better performance a larger training set is required.

(b)But it may also be possible to do a parameter sweep to see if the performance can get better. This means that the 100 percent score for regular messages might be lost, but if it gives less false positives that is still an increase in performance.
A frequency treshold was implemented so that we could control the amount of bigrams that are actually in the logprob vocabulary. If a bigram occurs only once in your training set, it might not be a reliable test criterion on a test set. First we test with the default setting, and increase the frequency treshold and keep the other parameters equal. Here are the results:

```
Epsilon 0.001; FT 1; Word length tresholds 4 & 3:
Correctly regular: 63 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 5 | 83.87096774193549%

Epsilon 0.001; FT 2; Word length tresholds 4 & 3:
Correctly regular: 58 | Falsely regular: 1 | 98.30508474576271%
Correctly spam: 25 | Falsely spam: 10 | 71.42857142857143%

Epsilon 0.001; FT 3; Word length tresholds 4 & 3:
```

```
Correctly regular: 58 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 10 | 72.22222222222221%

Epsilon 0.001; FT 4; Word length tresholds 4 & 3:
Correctly regular: 53 | Falsely regular: 1 | 98.14814814814815%
Correctly spam: 25 | Falsely spam: 15 | 62.5%
```

The expectation that a higher frequency treshold would increase performance is not met. This may have to do with the size of the training and test set. Next, word length parameter sweep was performed. The word length tresholds were manipulated, and for each setting different frequency tresholds were used.

```
Epsilon 0.001; FT 1; Word length tresholds 4 & 4:
Correctly regular: 64 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 4 | 86.66666666666667%

Epsilon 0.001; FT 2; Word length tresholds 4 & 4:
Correctly regular: 59 | Falsely regular: 2 | 96.72131147540983%
Correctly spam: 24 | Falsely spam: 9 | 72.72727272727273%

Epsilon 0.001; FT 3; Word length tresholds 4 & 4:
Correctly regular: 57 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 11 | 70.27027027027027%

Epsilon 0.001; FT 4; Word length tresholds 4 & 4:
Correctly regular: 53 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 15 | 63.41463414634146%


Epsilon 0.001; FT 1; Word length tresholds 5 & 4:
Correctly regular: 64 | Falsely regular: 1 | 98.46153846153847%
Correctly spam: 25 | Falsely spam: 4 | 86.20689655172413%

Epsilon 0.001; FT 2; Word length tresholds 5 & 4:
Correctly regular: 59 | Falsely regular: 2 | 96.72131147540983%
Correctly spam: 24 | Falsely spam: 9 | 72.72727272727273%

Epsilon 0.001; FT 3; Word length tresholds 5 & 4:
Correctly regular: 58 | Falsely regular: 1 | 98.30508474576271%
Correctly spam: 25 | Falsely spam: 10 | 71.42857142857143%

Epsilon 0.001; FT 4; Word length tresholds 5 & 4:
Correctly regular: 54 | Falsely regular: 1 | 98.18181818181819%
Correctly spam: 25 | Falsely spam: 14 | 64.1025641025641%


Epsilon 0.001; FT 1; Word length tresholds 6 & 4:
Correctly regular: 64 | Falsely regular: 1 | 98.46153846153847%
Correctly spam: 25 | Falsely spam: 4 | 86.20689655172413%

Epsilon 0.001; FT 2; Word length tresholds 6 & 4:
Correctly regular: 59 | Falsely regular: 2 | 96.72131147540983%
Correctly spam: 24 | Falsely spam: 9 | 72.72727272727273%

Epsilon 0.001; FT 3; Word length tresholds 6 & 4:
Correctly regular: 59 | Falsely regular: 1 | 98.33333333333333%
Correctly spam: 25 | Falsely spam: 9 | 73.52941176470588%

Epsilon 0.001; FT 4; Word length tresholds 6 & 4:
Correctly regular: 55 | Falsely regular: 1 | 98.21428571428571%
Correctly spam: 25 | Falsely spam: 13 | 65.78947368421053%


Epsilon 0.001; FT 1; Word length tresholds 7 & 4:
Correctly regular: 65 | Falsely regular: 1 | 98.48484848484848%
Correctly spam: 25 | Falsely spam: 3 | 89.28571428571429%

Epsilon 0.001; FT 2; Word length tresholds 7 & 4:
Correctly regular: 59 | Falsely regular: 2 | 96.72131147540983%
Correctly spam: 24 | Falsely spam: 9 | 72.72727272727273%

Epsilon 0.001; FT 3; Word length tresholds 7 & 4:
Correctly regular: 59 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 9 | 74.28571428571429%

Epsilon 0.001; FT 4; Word length tresholds 7 & 4:
Correctly regular: 55 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 13 | 66.66666666666666%
```

For bigger word lengths a decrease in performance is observed as the frequency treshold increases. This is in line with the earlier observation that this unmet expectation has to do with the size of the dataset. Being more strict even reduces the available data to work with. What we see is that the best results are achieved with $\epsilon = 0.001, FT = 1, WLT1 = 7, WLT2 = 4$. Now a sweep on the ϵ parameter for the bigram code has not been done yet, so this will be done next:

```
Epsilon 0.01; FT 1; Word length tresholds 7 & 4:
Correctly regular: 64 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 4 | 86.66666666666667%

Epsilon 0.01; FT 2; Word length tresholds 7 & 4:
Correctly regular: 59 | Falsely regular: 1 | 98.33333333333333%
Correctly spam: 25 | Falsely spam: 9 | 73.52941176470588%

Epsilon 0.01; FT 3; Word length tresholds 7 & 4:
Correctly regular: 58 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 10 | 72.22222222222221%

Epsilon 0.01; FT 4; Word length tresholds 7 & 4:
Correctly regular: 54 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 14 | 65.0%


Epsilon 0.0001; FT 1; Word length tresholds 7 & 4:
Correctly regular: 65 | Falsely regular: 1 | 98.48484848484848%
Correctly spam: 25 | Falsely spam: 3 | 89.28571428571429%

Epsilon 0.0001; FT 2; Word length tresholds 7 & 4:
Correctly regular: 60 | Falsely regular: 2 | 96.7741935483871%
Correctly spam: 24 | Falsely spam: 8 | 75.0%

Epsilon 0.0001; FT 3; Word length tresholds 7 & 4:
Correctly regular: 62 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 6 | 81.25%

Epsilon 0.0001; FT 4; Word length tresholds 7 & 4:
Correctly regular: 59 | Falsely regular: 0 | 100.0%
Correctly spam: 26 | Falsely spam: 9 | 74.28571428571429%


Epsilon 0.00001; FT 1; Word length tresholds 7 & 4:
Correctly regular: 65 | Falsely regular: 1 | 98.48484848484848%
Correctly spam: 25 | Falsely spam: 3 | 89.28571428571429%

Epsilon 0.00001; FT 2; Word length tresholds 7 & 4:
Correctly regular: 62 | Falsely regular: 2 | 96.875%
Correctly spam: 24 | Falsely spam: 6 | 80.0%

Epsilon 0.00001; FT 3; Word length tresholds 7 & 4:
Correctly regular: 63 | Falsely regular: 1 | 98.4375%
Correctly spam: 25 | Falsely spam: 5 | 83.33333333333334%

Epsilon 0.00001; FT 4; Word length tresholds 7 & 4:
Correctly regular: 60 | Falsely regular: 1 | 98.36065573770492%
Correctly spam: 25 | Falsely spam: 8 | 75.75757575757575%
```

An interesting observation can be made here. It seems that as ϵ approaches 0, the decrease in performance due to the frequency treshold seems to decrease. So if the frequency treshold were to be fixed at a number higher than one, it would be wise to take ϵ as close to 0 as you can. If the frequency treshold is 1, then it does not make a significant difference.

3 Final questions (5)

(1)
Let me consider the following code from `testMessage`:

```
if (sumPosteriRegular > sumPosteriSpam)
{
        return "regular!";
} else {
        return "spam!";
}
```

Well, those a Posteri numbers hold the same value for a message that is in dutch, since none of the words are recognized. This means that `sumPosteriRegular` is not greater than `sumPosteriSpam` and it is classified as spam.

(2)
In language, words often occur in patterns with other words. Like if you have a message that contains the word 'drugs.' Then you are much more likely to also find words like 'crime,' 'cocaine,' and bigrams like 'money laundering' in that same message. So in language there is always a dependence. Also bigrams are not always independent. There is always some sort of relation between and within bigrams, simply because English is an SVO language. That means that for a text with simple sentences there are a lot of bigrams that have a subject as a first token and a verb as a second token, and verb as first token and object as second token, where the first bigram directly precedes the second. Also there are lots of bigrams with a determiner as the first token and an object as the second token. You can already see that there are relations going on here, and therefore I conclude that words and bigrams are not independent. To put it in other words, the Naive Bayes assumption is violated. On a side note, it seems that the training and test data contain nonsense words, which might also be a cause for the below optimal results. However real people also use nonsense words in their e-mails, so this is a good reflection of reality.