CENG 463
Assignment 1

Cem Gündoğdu
cem.gundogdu@metu.edu.tr

December 16, 2021

Contents

1 Naive Bayes Classifier **2**
- 1.1 Introduction . . . 2
- 1.2 Metrics for different preprocessing techniques . . . 2
  - 1.2.1 Simplest version . . . 2
  - 1.2.2 Lowercase conversion . . . 2
  - 1.2.3 Punctuation removal . . . 2
  - 1.2.4 Stopword removal . . . 4
  - 1.2.5 Stemming . . . 4
  - 1.2.6 Removing short words . . . 4
- 1.3 Evaluation . . . 4
  - 1.3.1 Calculation of metrics . . . 4

1 Naive Bayes Classifier

1.1 Introduction

I used the `NaiveBayesClassifier` class from the NLTK library. I have tried various preprocessing techniques, and logged the accuracy along with the confusion matrix and metrics such as precision.

1.2 Metrics for different preprocessing techniques

In this section, I list the accuracies for different preprocessing techniques. I haven't tried all combinations of them, since the number grows exponentially. However, I added each technique incrementally, with the hope that the change caused by the added technique will be representative of its usefulness.

These trials were made on the *dev* set, since I did them during the development phase. The evaluation in subsection 1.3 will be done on the *test* set.

Complete logs from each try is available in the data directory. These include the confusion matrix, accuracy, and for each category the precision, recall and F_1 measure. I include only the last log here, to keep this report short.

Details of the calculation of precision, recall and F_1 measure are given in subsection 1.3.

A table of the accuracies for each technique is in Table 1.

Table 1: Accuracies for different preprocessing techniques.

Preprocessing type	Accuracy	
	Naive Bayes	Support Vector
Simplest version	0.6420	0.6409
Lowercase conversion	0.6592	0.6677
Punctuation removal	0.6538	0.6870
Stopword removal	0.6763	0.7074
Stemming	0.6667	0.7234
Removing short words	0.6699	0.7278

1.2.1 Simplest version

The words in title are counted twice — in this version and in all the following ones. Other than this, there is no processing in this version. The text is given to `nltk.word_tokenize()` and the resulting tokens are used to train the classifier.

Accuracy in this version is 64.2%. Confusion matrix and other metrics for each category are listed in Figure 1.

1.2.2 Lowercase conversion

Converted all words to lowercase.

1.2.3 Punctuation removal

Removed the following characters from the text:

Figure 1: Metrics for the simplest version

```
Loaded classifier from cache.
Accuracy: 0.6420150053590568
                |                          s     |
                |                          c     |
                |                          i     |
                |                          e     |
                |                          n     |
                |          p               c     |
                |          h               e     |
                |          i   r           -     |
                |      m   l   e   r   s   f     |
                |  h   y   o   l   o   c   i   s |
                |  o   s   s   i   m   i   c   p |
                |  r   t   o   g   a   e   t   o |
                |  r   e   p   i   n   n   i   r |
                |  o   r   h   o   c   c   o   t |
                |  r   y   y   n   e   e   n   s |
----------------+--------------------------------+
         horror | <44> 23   .   3  27   6   8   7 |
        mystery |   8 <88>  .   .  18   .   1   5 |
     philosophy |   .   2 <44> 20   .  47   1   . |
       religion |   .   3  13 <76>  4  12   3   4 |
        romance |   1   3   .   3 <82>  2   4  19 |
        science |   .   .   .   3   .<107>  4   1 |
science-fiction |  11   7   .   3  23  15 <52>  9 |
         sports |   .   .   .   .   8   3   .<106>|
----------------+--------------------------------+
(row = reference; col = test)



                |  Precision  |     Recall  | F1-Measure
-------------------------------------------------------------
     philosophy  |     0.7719  |     0.3860  |     0.5146
         sports  |     0.7020  |     0.9060  |     0.7910
        mystery  |     0.6984  |     0.7333  |     0.7154
       religion  |     0.7037  |     0.6609  |     0.6816
        science  |     0.5573  |     0.9304  |     0.6971
        romance  |     0.5062  |     0.7193  |     0.5942
         horror  |     0.6875  |     0.3729  |     0.4835
science-fiction  |     0.7123  |     0.4333  |     0.5389
```

```
!"#$%&'()*+,./:;<=>?@[\]^_`{|}~'
```

I did not replace them with spaces, but simply removed them. This made, for example, the text *Sophies's* to become the single word *sophies*.

1.2.4 Stopword removal

I used the corpus `nltk.corpus.stopwords.words('english')` from the NLTK library. I removed every word that occured in this list.

1.2.5 Stemming

I passed each word to the `nltk.stem.PorterStemmer()` from the NLTK library.

1.2.6 Removing short words

I thought stopword removal should have been enough, but I also tried removing words that were shorter than 3 characters. This resulted in a small increase in accuracy.

1.3 Evaluation

1.3.1 Calculation of metrics

Recall is the ratio of true positives for a class to the number of input documents of that type. To find recall, we divide each diagonal entry by the sum of corresponding row.

Precision is the ratio of true positives for a class to the number of documents that are identified to be in that class. To calculate it, we divide diagonal entries by the sum in that column.