This directory provides an implementation of the Naive Bayes learning algorithm similar to that described in Table 6.2 of the textbook. It also provides a dataset containing 20,000 newsgroup messages drawn from the 20 newsgroups described in Table 6.3. This code and data are only supported under the Unix and Linux operating systems. For example, one of the `unixXXX.andrew.cmu.edu` servers will do fine.

There are four files with extension `tar.gz` under this directory. Copy each to your home directory and unpack with the following commands:

```
cp /afs/cs/project/theo-20/www/mlc/hw3/text/XXX.tar.gz ./
gtar zxvf XXX.tar.gz
```

Now you should find four subdirectories under your home directory: `<./20_newsgroups>`, `<./mini_newsgroups>`, `<./bow-20020213>` and `<./example>`.

Newsgroup Data

`<./20_newsgroups>` contains 20 subdirectories with names described in Table 6.3 of the text book. Each of these subdirectories contains 1000 newsgroup articles drawn from the corresponding newsgroup, forming a dataset of 20,000 documents. Each document is a Unix ASCII file with header information (e.g. newsgroup name, subject, date, etc).

`<./mini_newsgroups>` is a subset of the Newsgroup data, which contains 100 randomly selected messages from each newsgroup. This is a useful dataset for learning to use the provided code.

Naive Bayes Code

You will use the program Rainbow written by Andrew McCallum. Rainbow provides a suite of learning methods for text classification, including the Naive Bayes method described in section 6.10 of the Machine Learning textbook. This code may be used as both a building block for creating other programs, or as a stand-alone learning/classification system.

Rainbow is based on Bow/Libbow, a library of C code useful for writing statistical text analysis, language modeling and information retrieval programs. The current distribution includes the library, as well as front-ends for document classification (rainbow), document retrieval (arrow) and document clustering (crossbow).

`<./bow-20020213>` contains the most recent Bow/Libbow source code. You need to compile the package first by issuing the following commands:

```
cd ./bow-20020213
./configure
gmake
```

When the compilation is done, you should see the executable program `rainbow` under the source code directory. Copy it to the directory where training/test is carried out, say your home directory.

```
cp rainbow ../
```

For your convenience, the distribution has provided a Perl script `rainbow-stats` that helps you look at the classification result. Copy it to the directory where training/test is carried out, say your home directory.

```
cp rainbow-stats.pl ../rainbow-stats
cd ../
chmod a+x rainbow-stats
```

A list of options and their functionalities for Rainbow is available with the following command:

```
./rainbow --help | more
```

The example below will step you through the process of training and testing a classifier. However, for a more complete tutorial on using Rainbow, please consult its online documentation at

`http://www.cs.cmu.edu/~mccallum/bow/rainbow/`

Example

`<./example>` is a self-contained example of using Rainbow to implement the Naive Bayes text classifier. You will find an executable file `rainbow` under this directory which is precompiled on the Andrew-side Unix machine. Replace it with the program you compiled from the source code if necessary.

Rainbow assumes each training or test document is a Unix ASCII file. To provide training data, you must create a directory containing no files, then create one subdirectory per target value and place the files in these subdirectories. An example is the subdirectory `<./example/mini_newsgroups>`. Here you will find two subdirectories named `alt.atheism` and `misc.forsale`. Each of these subdirectories contains 100 files, drawn from the corresponding newsgroup.

To get started, switch to the `<./example>` directory, and type the following command on a single line to define the alias `naivebayes`:

```
alias naivebayes './rainbow -m naivebayes --no-stoplist --skip-header
                        --prune-vocab-by-occur-count=1 -d ./learned-model'
```

This tells Rainbow to use naivebayes as its learning method, to consider every word found in every document, to skip header information in documents, and to store its learned model in the subdirectory `<./learned-model>` of the example directory.

Train a Naive Bayes classifier to classify newsgroup articles as `alt.atheism` versus `misc.forsale` by typing

```
naivebayes --index mini_newsgroups/*
```

This should produce output something like:

```
Created directory `./learned-model'.
Class `alt.atheism'
  Counting words... files : unique-words ::
Class `misc.forsale'
  Counting words... files : unique-words ::
Class `alt.atheism'
  Gathering stats... files : unique-words ::    100 :     6330
Class `misc.forsale'
  Gathering stats... files : unique-words ::    100 :     6330
```

Notice Rainbow should now have created a subdirectory for you named `<./learned-model>` in which you will find a number of files.

After the training has finished, examine the words with highest information gain for discriminating the two target classes by typing

```
./rainbow -d ./learned-model --print-word-infogain=10
```

This should produce output something like:

```
Loading data files...
Placed remaining 200 documents in the train set:
Calculating info gain... words ::        30
 0.32457 writes
 0.23588 that
 0.21752 article
 0.14941 sale
 0.13943 not
 0.11935 say
 0.11286 condition
 0.11021 what
 0.10589 of
 0.10338 it
```

You may ask Rainbow to classify new documents using the model we just trained.

```
./rainbow -d ./learned-model --query="mini_newsgroups/misc.forsale/75987"
```

The output produced by Rainbow for this query is a list of the classes, each followed by a number. This number is the probability $P(class|document)$ calculated using the Naive Bayes formula. Note in this example, the probabilities output for the two classes are 1 and 0 respectively.

```
misc.forsale 1
alt.atheism 0
```

You can command Rainbow to classify all the data in the training set by using the command

```
./rainbow -d ./learned-model --test-files mini_newsgroups/
```

Each line of output lists a file name, its true classification, and the posterior probabilities assigned to each class by the learned Naive Bayes model. You can put the output into a file and look at a summary of the results using the provided Pearl script.

```
./rainbow -d ./learned-model --test-files mini_newsgroups/ > f
./rainbow-stats < f
```

You can tell Rainbow to test the classification performance on a hold-out set, i.e. a portion of the data that will not be used for training the classifier. This should give much more trustworthy results than testing on the training data itself. For example,

```
./rainbow -d ./learned-model --test-set=.4 --test=3 > h
./rainbow-stats < h
```

will output the results of three trials, each with a randomized test-train split in which 60 percent of the documents are used for training, and 40 percent for testing.