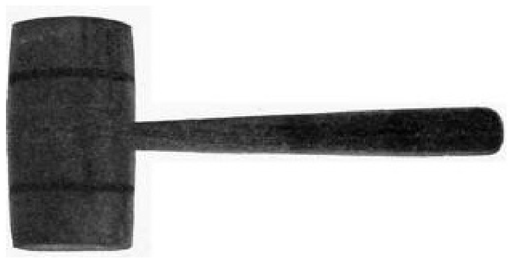
CS458 - Classification Exercise



Today, we're going to be playing with a few different classification approaches to classify newsgroup postings into one of 20 different categories.

1 Getting Everything

1. Download the 20 newsgroup dataset from:

   `http://qwone.com/~jason/20Newsgroups/`

   In particular, grab the `20news-18828.tar.gz` version. Save it somewhere where you can work on it.

2. Unpack the data

   In the shell, go into the directory where you downloaded the file and type:

   ```
   > gunzip 20news-18828.tar.gz
   > tar -xvf 20news-18828.tar
   ```

3. Download Mallet from:

   `http://mallet.cs.umass.edu/`

   (I'd suggest the .tar.gz version).

4. Compile Mallet

   In the shell, go into the directory where you downloaded the mallet file and type the following:

   ```
   > gunzip mallet-2.0.7.tar.gz
   > tar -xvf mallet-2.0.7.tar
   > cd mallet-2.0.7
   > make
   ```

2 Running Mallet

Assuming all went well, you should be in good shape now to run your first experiment with Mallet on the 20 newsgroup data.

1. Take a look at the data

   Look in the 20 newsgroup data directory. You'll see 20 subdirectories. Each of these directories corresponds to a different collection of newsgroup postings. We'll treat each of these different collections as a class.

   Look in one of the sub-directories, for example `alt.atheism`. You'll see a whole bunch of files. Each of these files is a posting, for example, `alt.atheism` contains 799 postings.

   Look at a few of these files. You'll see that each posting has who posted, the subject of the posting and then the text of the posting.

   Our goal will be given some training data (i.e. some documents from each of these 20 categories WITH known category) learn a model that given a new posting will predict which of these 20 categories that posting belongs to.

2. Preprocessing the data

   Mallet makes it pretty easy to preprocess this data and get it ready for experimentation. We'll play with pre-processing variants later, but for now, let's get a basic version of the data loaded.

   In the shell, `cd` into the main Mallet directory (i.e. `mallet-2.0.7`). To preprocess the data, type:

   ```
   > bin/mallet import-dir --input ../20news-18828/* --output ../20newsgroup.mallet
   ```

   Fill in the appropriate location for your appropriate `20news-18828` directory location and then feel free to put the `--output` file wherever you want and call it whatever you want.

3. Classify the data

   Now that we've pre-processed the data, we can train a classifier and see how it does. The default classifier is the Naive Bayes classifier, which is fine to start with.

   ```
   > bin/mallet train-classifier --input ../20newsgroup.mallet --training-portion 0.9
   ```

   The only two things we've specified is:

   - The input file: `../newsgroup.mallet`
   - The train/test split of 0.9. This tells mallet to randomly split the data into 90%/10% split. Train the classifier on the 90% portion and then evaluate (i.e. test) the algorithm on the 10% portion.

4. The results

   Mallet spits out a bunch of information regarding the results. A few useful things to note are:

   - accuracy (also referred to as the "test accuracy"): This is the proportion of the examples that were right on the test set.
   - train accuracy: This is the proportion of the examples that the classifier got right on the training data. After the classifier is trained on the training data, you can go back through and try and classify them and see how well you do.

     This may seem like a funny thing to do since you already of the labels, but it tells you a couple of important pieces of information. First, even without testing data it can give you *some* information about how well your classifier is doing. Be careful, though, since this is exactly the training data that you trained on so almost always these results are going to be inflated.

     Second, if we compare the training accuracy to the testing accuracy it can help us understand a bit about how much our classifier is biased by the training data. Larger differences between the training and testing accuracy may mean that our model is too flexible and biasing too much towards the training data.
   - confusion matrix: Mallet also prints out the confusion matrix between the 20 different classes. We saw these already when we looked at music genre classification. Confusion matrices are a useful way of understanding where the classifier is making mistakes and what classes might be related.

   Take a look at the data and see if you find any surprises.

3 Better Classification

Now that you have a basic version running, look at the documentation for Mallet online:

`http://mallet.cs.umass.edu/`

and try out some different variants to see how well you can do on a 90/10 split of the data.

A few things to consider trying are:

- Play with preprocessing parameters

  If you look at:

  `http://mallet.cs.umass.edu/import.php`

  there is a description about importing data in Mallet. Near the bottom, there are a number of preprocessing variations you can turn on/off. These can have a non-trivial impact on the performance. If you want to try some of these out you'll need to regenerate your input file and then rerun your classifier. I'd recommend generating *different* files for different preprocessing variations if you go this route.

- Different classifiers

  Mallet has a few different classifiers you can try out. Naive Bayes is fast, but it doesn't always give the best result (as we talked about last time, the Naive Bayes assumption often doesn't actually hold for text).

  Look at:

  `http://mallet.cs.umass.edu/classification.php`

  to see some of the other classifier variants. MaxEnt should work pretty well, but try others. C4.5 is a variant of decision trees.

  You can modify the algorithm with the `--trainer` flag.

4 Final Results

When you think you have a good feeling for what is working a better way of evaluating your approach is to do multiple train/test splits of the data. To do this run Mallet with the `--num-trials` flag, in particular:

```
> bin/mallet train-classifier --input ../20newsgroup.mallet --training-portion 0.9 --num-trials 10
```

(You could also consider doing 10-fold cross validation, which is similar, though not identical).