Dan Haskin
Project 3 Report
4-28-2013

I cannot tell you how grateful I am that you are reading this report.

NOTE: At the time of writing, the code used in this project can also be checked out from https://github.com/djhaskin987/CS453-project3.git for convenience.
Simply run
git clone https://github.com/djhaskin987/CS453-project3.git
And follow the instructions in the file `README.md` .

Introduction

This report details the Implementation Choices, Outcome, and general performance of the Muti-Nomial Bayes Classifier as it has been implemented by myself.

Implementation Choices

Tokenization

I used a porter stemmer in my tokenizer, and made the tokens case-insensitive by changing all the characters in each token to lower-case. Tokens were split by spaces, with any punctuation adjacent to the spaces removed. For example, the line

```
“I can't keep holding on to your hand,” she said.
```

Would be tokenized as

```
[”keep”,”hold”,”hand”,”said”]
```

by my tokenizer.

Choice of Data Structures

After the documents were tokenized, they were put into data structures which presented the document in its Multi-Nomial representation. The document collection (the structure “DC” as described in the specifications) was a Map, whose keys were the doc ID, and whose value was data about that documentation. That data was a pair, whose first element was the document of the class. The second element of the pair was a map from words to the number of times each word appeared in that document. Making the structure this way was straitforward and seemed to agree with the specifiations' request to represent the documents as they appeared on the slides.

Optimizations

I employed two main optimizations in creating my implementation of the MNB:

1. `MNB_probability`: in `computeWordProbability()`, I didn't compute
$$P(w|c)$$
for *every* word; rather, just for every word which appeared in the class `c`. Then, in `getWordProbability()`, if a word didn't appear in `c`, the value

$$\frac{1.0}{(|c|+|V|)}$$

was returned. This is correct behavior, and it allowed me to save space and lots of time during `computeWordProbability()`.

2. `MNB_classification`: I didn't compute the value

$$P(w|c)^{tf_{wd}}$$

for *every* word, just for those terms for which the word *w* was in *d*, since if *w* wasn't in *d* then

$$P(w|c)^{tf_{wd}} = P(w|c)^0$$

which would be 1, and so it wouldn't affect the computation.

With the above optimizations in addition to stemming and lower-casing the tokens, I was able to get around 1 second training *and* testing time average, even when using the entire vocabulary as features.

Experimental Results on MNB

Documents from the 20NG collection were classified into the following categories:

- comp.graphics
- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- rec.autos
- rec.motorcycles
- sci.crypt
- sci.electronics
- soc.religion.christian
- talk.politics.guns
- talk.politics.mideast

Output of the Program

```
Tokenizing corpus found under ../data/20NG...



Reporting results for using ***ALL FEATURES***
Run #1 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:02.049

Testing with all features...
```

```
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:02.248
Accuracy: 49.3%
Run #2 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:04.507

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:01.544
Accuracy: 48.26%
Run #3 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:01.540

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:01.480
Accuracy: 49.3%
Run #4 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:01.707

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:01.362
Accuracy: 48.36%
Run #5 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:01.490

Testing with all features...
Determining Accuracy...
```

```
Done measuring Accuracy.
Test time: 0:00:01.614
Accuracy: 47.53%
=====================================
Average training time: 0:00:02.258
Average test time: 0:00:01.649
Average accuracy: 48.55%



Reporting results for ***6200 FEATURES***
Run #1 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.140

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.122
Accuracy: 69.34%
Run #2 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.152

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.207
Accuracy: 70.43%
Run #3 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.142

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.112
Accuracy: 66.74%
Run #4 ==============================
Initializing...
```

```
Done initializing.
Training...
Done training.
Training time: 0:00:00.142

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.101
Accuracy: 69.39%
Run #5 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.138

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.113
Accuracy: 70.54%
=====================================
Average training time: 0:00:00.142
Average test time: 0:00:00.130
Average accuracy: 69.29%



Reporting results for ***12400 FEATURES***
Run #1 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.263

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.253
Accuracy: 78.5%
Run #2 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.263
```

```
Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.188
Accuracy: 77.98%
Run #3 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.283

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.208
Accuracy: 76.94%
Run #4 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.283

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.216
Accuracy: 77.3%
Run #5 ==============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.284

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.207
Accuracy: 78.24%
=====================================
Average training time: 0:00:00.275
Average test time: 0:00:00.214
Average accuracy: 77.79%



Reporting results for ***18600 FEATURES***
```

```
Run #1 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.423

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.374
Accuracy: 85.16%
Run #2 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.411

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.296
Accuracy: 83.39%
Run #3 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.428

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.338
Accuracy: 84.44%
Run #4 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.400

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.310
Accuracy: 83.65%
Run #5 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.416

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.496
Accuracy: 81.94%
=====================================
Average training time: 0:00:00.415
Average test time: 0:00:00.362
Average accuracy: 83.72%



Reporting results for ***24800 FEATURES***
Run #1 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.572

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.400
Accuracy: 85.01%
Run #2 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.580

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.408
Accuracy: 84.23%
Run #3 ===============================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.644
```

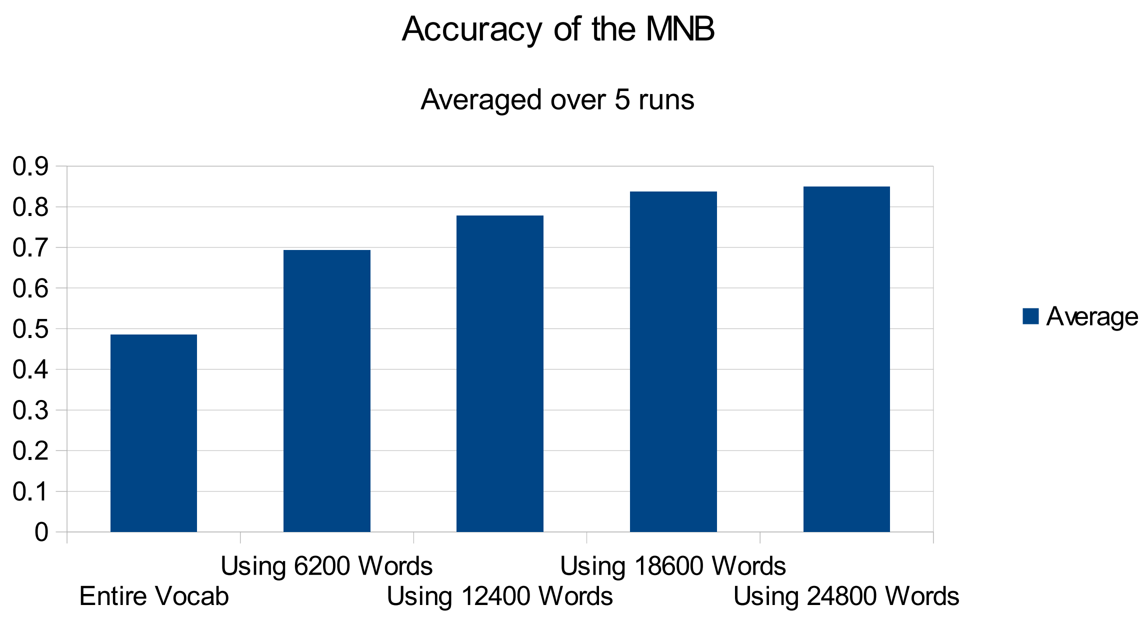
```
Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.402
Accuracy: 84.8%
Run #4 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.575

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.457
Accuracy: 86.21%
Run #5 ================================
Initializing...
Done initializing.
Training...
Done training.
Training time: 0:00:00.541

Testing with all features...
Determining Accuracy...
Done measuring Accuracy.
Test time: 0:00:00.391
Accuracy: 84.59%
=====================================
Average training time: 0:00:00.582
Average test time: 0:00:00.411
Average accuracy: 84.97%
```

Accuracy of the MNB

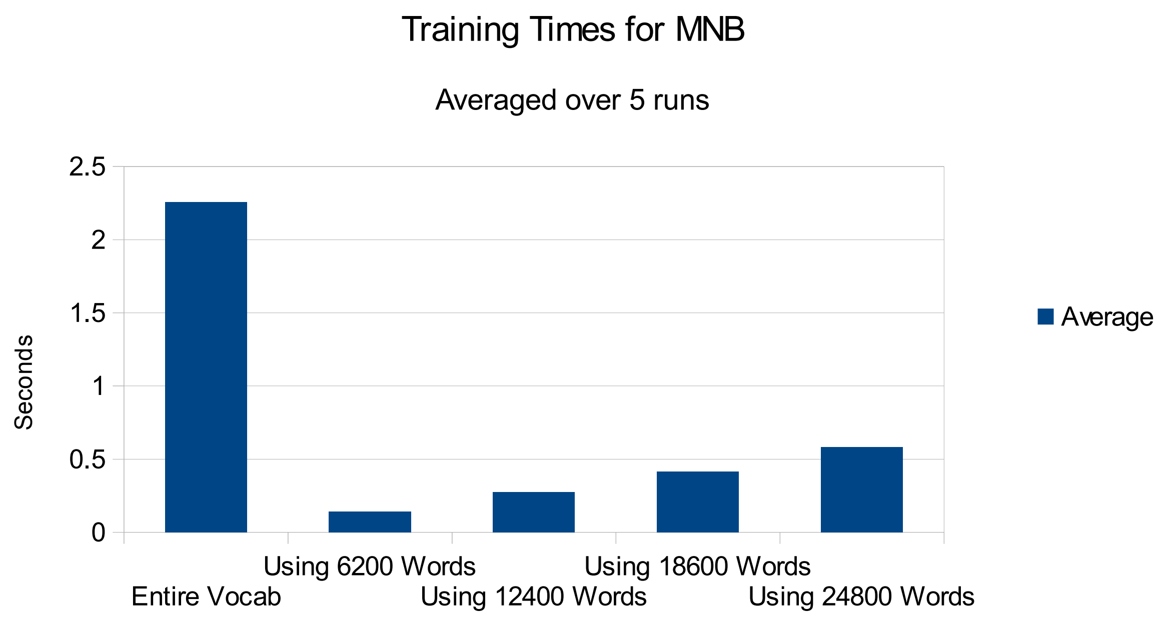
Diagram: Accuracy with Feature Selection vs. No Feature Selection

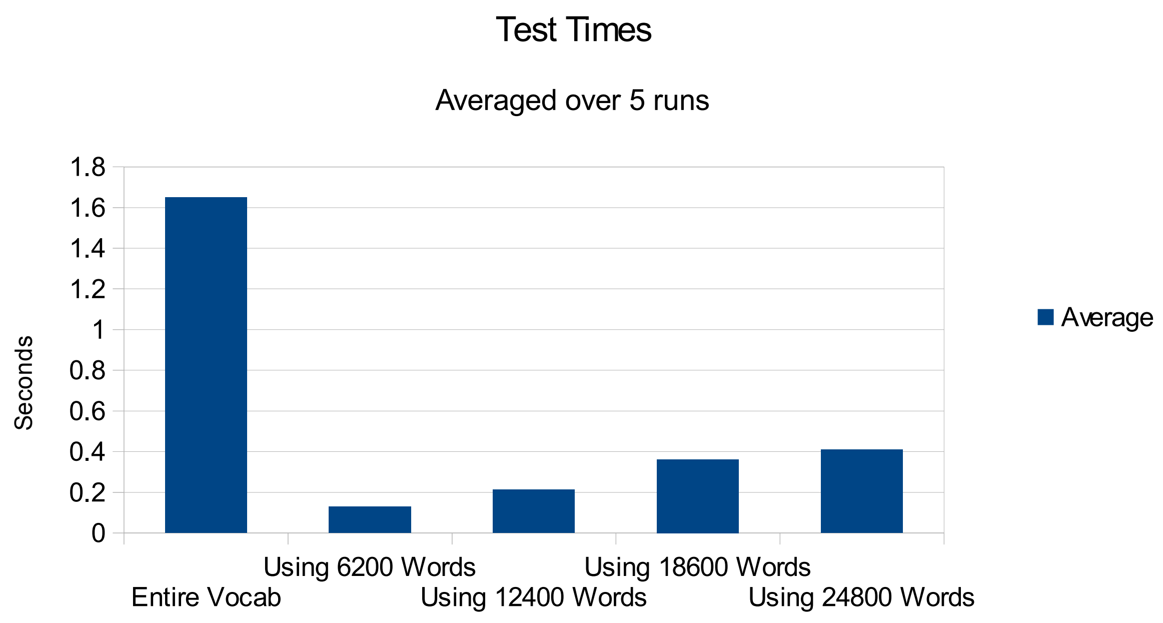


Discussion: Accuracy with Feature Selection vs. No Feature Selection

Perhaps the most intriguing part of the results of the evaluation of my MNB implementation is that, when feature selection was used, my accuracy actually drastically *improved*, instead of worsening. Using all the words, my MNB classifier was wrong about half the time. Using fewer, accuracy grew to about 85%. Reflecting, this is perhaps because my tokenizer considers things as "words" such as "-", which is a word according to my tokenizer. Such "noisy" words are eliminated by feature selection.

Diagram: Training and Test Time with Feature Selection vs. No Feature Selection





Discussion: Training and Test Time with Feature Selection vs. No Feature Selection

Although the training time was much shorter using feature selection, running the feature selection algorithm itself took lots of time to run when that didn't need to happen for the 'all features' run (depicted above). That said, clearly using feature selection drastically improves actual training and test time.

Conclusion

Looking at these results, I would say that the best configuration for an MNB as used in this project would be when it is using 18600 words in its feature selection. That seems to get nearly as accurate results as when there are 24800 words, but while the accuracy doesn't change much, the time spent running the algorithm still grows. So, then, using 18600 words seems to be a good fit for the purpose of classifying this particular corpus.