Ben Schwemlein
Final Project Documentation
Naive Bayes Text Classifier
CSC 578
Fall 2008

a). Installation and Run Instructions

1. Compiling
   1. I used Java version 1.6.0_01 to compile and run my project. I used Windows XP, but any OS with this JVM/SDK installed should work.
   2. Copy all files and directories, including .java, and .txt from my code directory onto your computer.
   3. To compile type **javac *.java** in the same directory as the code. This should compile and create a .class file for all the .java files.
   4. My project has 4 java/class files, which are ...
      1. BayesText: The main file that contains most of the Algorithm.
      2. Int: A simple container for int values with an increment() method for HashMap use.
      3. TargetValue: Represents each target value and has a reference to each of it's example files.
      4. Word: Represents a unique word and stores the per documents word count and conditional probabilities

2. Running
   The input is through the terminal with either one or two input parameters (ignoring the two heap size options). The first parameter is always the directory containing the example documents. The second parameter is the directory containing the validation documents. Type **java -Xmx1024m -Xms1024m BayesText <example directory name> <validation directory name>** to run.

   If the second parameter is omitted then BayesText will use 1/3 of the example documents, located in the first parameter's directory, to validate. This reduces the number of example documents down to 2/3.

   The example directory folder should have the same structure as Tom Mitchell's 20_Newsgroups. Namely **../example_directory/target_value_name/file_name** In this case, typing **java -Xmx1024m -Xms1024m BayesText <example_directory>** would run BayesText on the directory. My program uses the names of the **target_value_name** directories for target values. If a separate validation set is used then it's directory, specified by the 2nd parameter, does not necessarily need to have the same structure. However, if it does not, then my program can't determine the classification accuracy on the validation set, which is indicated with a n/a in the outputfile.

   Please note that since my program stores all unique vocabulary words in a HashMap, your JVM default heap size will likely run out of space for large example sets such as Tom Mitchell's 20 newgroups. To remedy this you must temporarily increase your JVM Heap size by including the parameters -Xmx1024m -Xms1024m, which increases the heap size to 1 gigabyte.

   Most output is sent to the terminal, which includes runtime updates and the total classification accuracy. Validation file names and their classification are outputted to **bayestext_out_xxx.txt**, where xxx is the current data and time.

Examples

- ***Runs BayesText on Tom Mitchell's newsgroup examples, using 1/3 of the 20000 files as the validation set. THIS MAY TAKE A MINUTE OR TWO TO RUN***

```
java -Xmx1024m -Xms1024m BayesText 20_newsgroups
```

Heres the Terminal output.

```
Example Set Size = 13332
Validation Set Size = 6665

Defining target values...

        target = alt.atheism
        target = comp.graphics
        target = comp.os.ms-windows.misc
        target = comp.sys.ibm.pc.hardware
        target = comp.sys.mac.hardware
        target = comp.windows.x
        target = misc.forsale
        target = rec.autos
        target = rec.motorcycles
        target = rec.sport.baseball
        target = rec.sport.hockey
        target = sci.crypt
        target = sci.electronics
        target = sci.med
        target = sci.space
        target = soc.religion.christian
        target = talk.politics.guns
        target = talk.politics.mideast
        target = talk.politics.misc
        target = talk.religion.misc

Building vocabulary...

        666 of 13332 documents read.    15157 unique words found
        1332 of 13332 documents read.    26959 unique words found
        1998 of 13332 documents read.    49657 unique words found
        2664 of 13332 documents read.    55648 unique words found
        3330 of 13332 documents read.    60616 unique words found
        3996 of 13332 documents read.    73318 unique words found
        4662 of 13332 documents read.    79460 unique words found
        5328 of 13332 documents read.    85345 unique words found
        5994 of 13332 documents read.    90558 unique words found
        6660 of 13332 documents read.    95724 unique words found
        7326 of 13332 documents read.    102245 unique words found
        7992 of 13332 documents read.    108642 unique words found
        8658 of 13332 documents read.    113519 unique words found
        9324 of 13332 documents read.    121211 unique words found
        9990 of 13332 documents read.    126989 unique words found
        10656 of 13332 documents read.    131911 unique words found
        11322 of 13332 documents read.    136983 unique words found
        11988 of 13332 documents read.    143998 unique words found
        12654 of 13332 documents read.    148623 unique words found
        13320 of 13332 documents read.    151717 unique words found

Removing 100 most frequent words and words that occur less than 3 times...

Final vocabulary size = 48038


Calculating  p(v_j) and p(w_k|v_j) probablity terms...


Classifying validation set...

        333 of 6665 documents classified.
        666 of 6665 documents classified.
        999 of 6665 documents classified.
        1332 of 6665 documents classified.
        1665 of 6665 documents classified.
        1998 of 6665 documents classified.
        2331 of 6665 documents classified.
        2664 of 6665 documents classified.
        2997 of 6665 documents classified.
        3330 of 6665 documents classified.
        3663 of 6665 documents classified.
        3996 of 6665 documents classified.
        4329 of 6665 documents classified.
        4662 of 6665 documents classified.
        4995 of 6665 documents classified.
        5328 of 6665 documents classified.
        5661 of 6665 documents classified.
        5994 of 6665 documents classified.
        6327 of 6665 documents classified.
        6660 of 6665 documents classified.

Number correctly classified = 5968
Accuracy = 89.5%
```

- **Runs BayesText using all 20000 of Tom Mitchell's newsgroup files for examples and then uses a a few bible excerpts for validation. Note that classification accuracy can't be determined since the bible excerpts have different target values. However, the classifications are intuitively correct (see output file below).** ***THIS MAY TAKE A MINUTE OR TWO TO RUN***

```
java -Xmx1024m -Xms1024m BayesText 20_newsgroups bible2
```

Here's the file output. I think it's pretty interesting.

```
Document_Path         Classification         Correct

bible2\exodus\1.txt   soc.religion.christian  n/a
bible2\exodus\2.txt   soc.religion.christian  n/a
bible2\exodus\3.txt   soc.religion.christian  n/a
bible2\exodus\4.txt   soc.religion.christian  n/a
bible2\exodus\5.txt   soc.religion.christian  n/a
bible2\exodus\6.txt   soc.religion.christian  n/a
bible2\exodus\7.txt   soc.religion.christian  n/a
bible2\exodus\8.txt   soc.religion.christian  n/a
bible2\exodus\9.txt   comp.os.ms-windows.misc         n/a
bible2\kings\1.txt    comp.os.ms-windows.misc         n/a
bible2\kings\2.txt    comp.os.ms-windows.misc         n/a
bible2\kings\3.txt    talk.religion.misc      n/a
bible2\kings\4.txt    talk.religion.misc      n/a
bible2\kings\5.txt    talk.religion.misc      n/a
bible2\kings\6.txt    talk.politics.mideast   n/a
bible2\kings\7.txt    comp.os.ms-windows.misc         n/a
bible2\kings\8.txt    comp.os.ms-windows.misc         n/a
bible2\kings\9.txt    talk.religion.misc      n/a
bible2\peter\1.txt    soc.religion.christian  n/a
bible2\peter\2.txt    soc.religion.christian  n/a
bible2\peter\3.txt    soc.religion.christian  n/a
bible2\peter\4.txt    soc.religion.christian  n/a
bible2\peter\5.txt    soc.religion.christian  n/a
bible2\peter\6.txt    soc.religion.christian  n/a
bible2\peter\7.txt    soc.religion.christian  n/a
bible2\peter\8.txt    soc.religion.christian  n/a
bible2\peter\9.txt    soc.religion.christian  n/a
```

- ***Runs BayesText on Depaul CDM course description examples, using 1/3 of the files as the validation set. Target values are the various CDM departments such as CSC, SE and IS***

```
.          java -Xmx1024m -Xms1024m BayesText course_descriptions
```

Here's the Terminal output.

```
Example Set Size = 451
Validation Set Size = 225

Defining target values...

        target = ANI
        target = CNS
        target = CSC
        target = DC
        target = DS
        target = ECT
        target = GAM
        target = GPH
        target = HCI
        target = IM
        target = IS
        target = IT
        target = ITS
        target = MM
        target = PM
        target = SE
        target = TDC
        target = VFX
        target = XANI
        target = XCNS
        target = XCSC
        target = XDS
        target = XECT
        target = XHCI
        target = XIS
        target = XPM
        target = XSE
        target = XTDC

Building vocabulary...

        22 of 451 documents read.    362 unique words found
        44 of 451 documents read.    792 unique words found
        66 of 451 documents read.    1088 unique words found
        88 of 451 documents read.    1197 unique words found
        110 of 451 documents read.   1378 unique words found
        132 of 451 documents read.   1546 unique words found
        154 of 451 documents read.   1813 unique words found
        176 of 451 documents read.   1935 unique words found
        198 of 451 documents read.   2038 unique words found
        220 of 451 documents read.   2216 unique words found
        242 of 451 documents read.   2302 unique words found
        264 of 451 documents read.   2425 unique words found
        286 of 451 documents read.   2519 unique words found
        308 of 451 documents read.   2588 unique words found
        330 of 451 documents read.   2734 unique words found
        352 of 451 documents read.   2823 unique words found
        374 of 451 documents read.   3080 unique words found
        396 of 451 documents read.   3133 unique words found
        418 of 451 documents read.   3229 unique words found
        440 of 451 documents read.   3315 unique words found

Removing 100 most frequent words and words that occur less than 3 times...

Final vocabulary size = 1186
```

```
Calculating  p(v_j) and p(w_k|v_j) probability terms...


Classifying validation set...

        22 of 225 documents classified.
        33 of 225 documents classified.
        44 of 225 documents classified.
        55 of 225 documents classified.
        66 of 225 documents classified.
        77 of 225 documents classified.
        88 of 225 documents classified.
        99 of 225 documents classified.
        110 of 225 documents classified.
        121 of 225 documents classified.
        132 of 225 documents classified.
        143 of 225 documents classified.
        154 of 225 documents classified.
        165 of 225 documents classified.
        176 of 225 documents classified.
        187 of 225 documents classified.
        198 of 225 documents classified.
        209 of 225 documents classified.
        220 of 225 documents classified.

Number correctly classified = 141
Accuracy = 62.6%
```

- **Runs BayesTest on a very small test set. I Included a spreadsheet which duplicates the calculations.**

```
java -Xmx1024m -Xms1024m BayesText test
```

Here's the terminal output.

```
Example Set Size = 4
Validation Set Size = 2

Defining target values...

        target = ben
        target = clare

Building vocabulary...

        1 of 4 documents read.    5 unique words found
        2 of 4 documents read.    8 unique words found
        3 of 4 documents read.    12 unique words found
        4 of 4 documents read.    13 unique words found

Final vocabulary size = 13


Calculating  p(v_j) and p(w_k|v_j) probablity terms...


Classifying validation set...

        1 of 2 documents classified.
        2 of 2 documents classified.

Number correctly classified = 1
Accuracy = 50.0%
```

Here's the output file.

```
Document_Path       Classification          Correct
```

```
test\ben\ben_two.txt   clare        F
test\clare\clare_two.txt            clare        T
```

b). Description of the System

My naive Bayes classifier mostly follows Tom Mitchell's algorithm with the exceptions that I count per target word counts while Vocabulary is built and used the product of the inverse of conditional probabilities' logarithms to avoid floating point underflow. The former was for performance efficiency.

Most of the algorithm is located in BayesText.java, which makes heavy use of HashMaps to store values and to do counting.   **Map<String, Word> vocabulary** stores every unique vocabulary word using the Word type which in turn stores the number of times the word is found in each target and the corresponding conditional probability p(w_x|v_j). **Map<String, TargetValue> docs** stores the target values, determined by the directory names, and stores a list of document files located in the directory. My program populates the Maps and then calls various methods on the Word and TargetValue types to calculate the p(v) and P(w_k|v_j) values. **Word.calculateP(int n, int vocabSize, String target)** calculates the P(w_k|v_j) value, while **Word.getP(String targetKey)** is used to get it later. **TargetValue.setPV(int numExamples)** is used to calcualte P(v_j).

My program determines words by using space delimiters, then deleting any non alphabetic characters such as punctuation and turns all character into lower case. Also per Tom Mitchell's description I eliminated words that occur less than 3 times and the 100 most frequent words. The result is a vocabulary of 48,000 words on the 20 newgroups data set; This 10,000 more than the 38,500 words that Mitchell indicated. However, my accuracy was still 89.5%

Classification accuracy is done by taking the number of validation set documents whose target value, given by their parent directory, match the class determined by argmax and then dividing by size of the validation set.

Debugging classification accuracy was a headache because of my ignoring the effect of floating point underflow when multiplying many small conditional probabilities. This caused some non-zero values in argmax to incorrectly be zero, which caused misclassification. My solution was to instead to multiply the inverse of the logarithms of the conditional probabilities, which still maintains their relative order of magnitude.

c) Ideas for enhancement

One obvious area for improvement is performance tuning. This isn't a particularly important issue with small data sets, but would appear with scaling to large ones such as the 20_newsgroups. I'm sure there are areas where I unnecessarily instantiated objects that otherwise could have been reused, which would hinder VM memory and garbage collection. Also, although String parsing using tokens is easy to implement, there are probably faster ways to parse text by using the native char type and buffers to read and parse through files more efficiently. If I wanted to scale to example sets much larger than the 20 newsgroups then I would probably need to store the vocabulary and corresponding probabilities in a database, tuple-space, distributed JVM or some other established way to scale large volumes of data.

From a software architecture standpoint, it might be good to use a text stream instead of the particular file structure as input in the algorithm so it robustly handle more situations such as web services, ect. Also it would be better to separate/modularize the classification function, so it can be used independent of training.

Algorithmically speaking, this separation is important since one of primary benefits of naive Bayes classification is that you don't need to keep the training examples around to classify; just the probabilities and target values.

Another other area for enhancement is to more precisely define what a word is. Tom Mitchell's book wasn't specific on what he meant by a word. However, since my vocabulary size is different, I assume that our definitions our different. Although, our accuracy was the same for the 20_newgroups example set, I am not confident enough that my code is production ready.

One last potential issue, occurred when at the last minute I tested my code on my Linux Laptop. When I run BayesText with just one parameter, splitting the documents into 2/3 example set and 1/3 validations set, I noticed different accuracies and vocabulary sizes. (A file is thrown into the validation set if it's count is modulus 3) However, when I run BayesText with a separate validation set (using 2 parameters) the results were identical to my windows pc. This makes me think that Linux may return a different file order to the JVM than windows, which would therefor cause my program to have different example and validation sets with Linux. The output files do show a different order. I did notice that large sets like the 20 newsgroups had a small difference, which would occur with larger samples. Smaller sets like course descriptions had a larger difference; 62.6.% - 56.4% = 6.2%. Regardless of the cause, I should really choose a consistent and OS transparent way to split into the example and validation sets