Text Classification Tools Version 0.11

Paul Wolfgang
Department of Computer and Information Science
Temple University
Philadelphia PA
USA

Introduction

This document is preliminary documentation on text classification tools which were developed to support the Pennsylvania Policy Database Project. Five classification algorithms are provided: State Vector Machine, Naïve Bayes, Maximum Entropy, Character N-Gram, and Token N-Gram. The programs provided are built on tools developed by others as documented in the references. The goal of these programs is to provide a relatively simple to use command line interface and to provide for input/output to a database. These programs are designed to be run under the Microsoft Windows operating system.

Prerequisites

To access data from a database, the database must be registered as an ODBC datasource, or use a MySQL database.

Installation

Un-zip the `TextTools_v0.11.zip` file to a working directory.

Running the tools

The tools are designed to be run from the command line. To get to a command line window either double click on the appropriate icon on the desktop, select `Command Prompt` from the `START` menu, or enter `cmd` into the `Run` dialog box.

Training tools

The training tools take labeled input and generate a model for the specified classifier.

TrainSVM

The TrainSVM tool will train the SVM classifier. To run this tool issue the command:

`TrainSVM`*parameters*

Where *parameters* is a set of command parameters defined as follows:

`-Xmx`*nnnn*`m`

This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

--input_file *File containing the training data*
--datasource *The datasource name – see discussion of datasource below*
--table_name *Table containing the training data*
--id_column *Column containing the ID*
--text_column *Column(s) containing the text*
--code_column *Column containing the code*
--model *Write the model files to this directory*
--feature_dir *Write the training feature files to this directory*
Default is SVM_Training_Features
--use_even [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--use_odd [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--compute_major [TRUE|FALSE]
If true, the major code is computed from the minor code
Default is false
--remove_stopwords [TRUE|FALSE | *language*]
If true, remove common "stop words" from the text using stop words defined by Chris Buckley and Gerard Salton. If a language is specified, a language specific list of stop words is used. These stop words are provided by Porter.
Default is true
--do_stemming [TRUE|FALSE | *language*]
If true, pass all words through the Porter stemmer. If a language is specified pass all words through a language-specific stemmer. The language specific stemmers are also defined by Porter. The one for English is an improvement over Porter's original.
Default is true

Training input may be either from a file as specified by the --input_file parameter or from a database as specified by the --datasource, --table_name, --id_column, --text_column, and --code_column parameters. Multiple text columns may be specified by using an expression enclosed in quotes. If a column name contains spaces it must be enclosed in brackets and quotes.

If the --input_file option is selected the input file is assumed to consist of individual training records, one record per line. The classification code is separated from the text by the | character. For example:

```
1|An Act making appropriations to the Treasury Department out of various
funds to pay replacement checks issued in lieu of outstanding checks when
presented and to adjust errors.
2|"An Act amending the act of October 27, 1955 (P. L. 744, No. 222),entitled,
as amended, ""Pennsylvania Human Relations Act,"" making it an unlawful
practice for a review organization to discriminate against a physician on the
basis of race."
3|"An Act amending the act of March 10, 1949 (P. L. 30, No. 14), entitled
""Public School Code of 1949,"" further providing for the Alcohol and
Chemical Abuse Program."
⋮
```

```
24|"An Act amending the act of July 12, 1972 (P. L. 781, No. 185),entitled,
as amended, ""Local Government Unit Debt Act,"" further providing for the
incurring of debt for certain assessment revisions."
```

(Note that each entry is on one line in the input file.)

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter is `--input_file` or `--datasource`, `--table_name`, `--id_column`, `--text_column`, and `--code_column`.

Examples

To train from the file `training_file.txt` and write the model to `Model1`, use the following:

```
TrainSVM --input_file training_file.txt --model Model1
```

To train from the database registered as datasource `PAPolicy19992000` using the table `Newspaper_Data_1999_2000`, where the ID is in column `ID`, the text in column `Abstract`, and the code in column `Code`, and write the model to `Model2`, use the following:

```
TrainSVM --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --model Model2
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

To train from the database registered as datasource `NYTSaample` using the even samples from the table `Issues`, where the ID is in column `ID`, the text in columns `Title` and `Text`, and the code in column `2-digit policy code` and write the model to `Model2`, use the following:

```
TrainSVM --datasource NYTSample --table_name Newspaper_Data_1999_2000
--id_column ID --text_column "Title & ' ' & Text" --code_column "[2-
digit policy code]" --use_even --model Model3
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

TrainMALLET

The TrainMALLET tool will train either the Naïve Bayes classifier or the Maximum Entropy classifier. To run this tool issue the command:

`TrainMALLET` *parameters*

Where *parameters* is a set of command parameters defined as follows:

`-Xmx`*nnnn*`m`

This is an optional parameter, but if specified it must be first. The value *nnnn* is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value

is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

`--input_file` *File containing the training data*
`--datasource` *The datasource name – see discussion of datasource below*
`--table_name` *Table containing the training data*
`--id_column` *Column containing the ID*
`--text_column` *Column(s) containing the text*
`--code_column` *Column containing the code*
`--model` *Write the model to this file*
`--use_even [TRUE|FALSE]`
If true, use even numbered samples for training
Default is `false`
`--use_odd [TRUE|FALSE]`
If true, use even numbered samples for training
Default is `false`
`--compute_major [TRUE|FALSE]`
If true, the major code is computed from the minor code
Default is `false`
`--preserve_case [TRUE|FALSE]`
If true, do not force all strings to lowercase
Default is `false`
`--remove_stopwords [TRUE|FALSE |` *language*`]`
If true, remove common "stop words" from the text using stop words defined by Chris Buckley and Gerard Salton. If a language is specified, a language specific list of stop words is used. These stop words are provided by Porter.
Default is `true`
`--do_stemming [TRUE|FALSE |` *language*`]`
If true, pass all words through the Porter stemmer. If a language is specified pass all words through a language-specific stemmer. The language specific stemmers are also defined by Porter. The one for English is an improvement over Porter's original.
Default is `true`
`--trainer` *classifier*
The name of the Java class that is the classifier
The default is `NaiveBayes`
To use the Maximum Entropy classifier specify `MaxEnt`

Training input may be either from a file as specified by the --input_file parameter or from a database as specified by the `--datasource`, `--table_name`, `--id_column`, `--text_column`, and `--code_column` parameters. Multiple text columns may be specified by using an expression enclosed in quotes. If a column name contains spaces it must be enclosed in brackets and quotes.

MALLET supports several classifiers, but only the Naïve Bayes and Maximum Entropy have been tested with this program.

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter is `--input_file` or `--datasource`, `--table_name`, `--id_column`, `--text_column`, and `--code_column`.

Examples

To train from the file `training_file.txt` and write the model to `Model1`, use the following:

```
TrainMALLET --input_file training_file.txt --model Model1
```

To train from the database registered as datasource `PAPolicy19992000` using the table `Newspaper_Data_1999_2000`, where the ID is in column `ID`, the text in column `Abstract`, and the code in column `Code`, write the model to `Model2`, and use the Maximum Entrophy classifier use the following:

```
TrainMALLET --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --model Model2 --trainer MaxEnt
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

Comment on the `--trainer` parameter: The MALLET package contains several training/classifying algorithms. Only the NaiveBayes, MaxEnt, and AdaBoostM2 algorithms have been tested. The ADAM2Boost algorithm is special in that it theoretically improves the performance of another algorithm. The syntax for its use is special:

```
--trainer "new AdaBoostM2Trainer(new NaiveBayesTrainer())"
```

This specifies that the AdaBoostM2 algorithm is to be applied to the Naïve Bayes algorithm to create an improved classification algorithm. Preliminary tests show that the performance is about the same as the Naïve Bayes algorithm alone, but different in that different samples are correctly classified.

TrainLingPipe

The TrainLingPipe tool will train either the Character N-Gram classifier or the Token N-Gram classifier. To run this tool issue the command:

`TrainLingPipe` *parameters*

Where *parameters* is a set of command parameters defined as follows:

`-Xmx`*nnnn*`m`

This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

`--input_file` *File containing the training data*
`--datasource` *The datasource name – see discussion of datasource below*
`--table_name` *Table containing the training data*
`--id_column` *Column containing the ID*
`--text_column` *Column(s) containing the text*
`--code_column` *Column containing the code*
`--model` *Write the model to this file*
`--use_even [TRUE|FALSE]`
If true, use even numbered samples for training
Default is `false`

--use_odd [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--compute_major [TRUE|FALSE]
If true, the major code is computed from the minor code
Default is false
--language_model
The language model used for classification
-- Options are N-Gram, Porter Stemmer, and WordsOnly
If N-Gram is specified a character-based N-Gram model is used
If Porter Stemmer is specified a token-based N-Gram model is used where the tokens are words that have been converted to their stem
If WordsOnly is specified a token-based N-Gram model is used where the tokens are words converted to lower-case.
Default is N-Gram
--gram_size The number of characters (tokens) in an N-gram
Default is 6
--categories Classification Categories
-- Default is 1 2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 19 20 21 24 99

Training input may be either from a file as specified by the --input_file parameter or from a database as specified by the --datasource, --table_name, --id_column, --text_column, and --code_column parameters. Multiple text columns may be specified by using an expression enclosed in quotes. If a column name contains spaces it must be enclosed in brackets and quotes.

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter is --input_file or --datasource, --table_name, --id_column, --text_column, and --code_column.

Examples

To train from the file training_file.txt and write the model to Model1, use the following:

```
TrainLingPipe --input_file training_file.txt --model Model1
```

To train from the database registered as datasource PAPolicy19992000 using the table Newspaper_Data_1999_2000, where the ID is in column ID, the text in column Abstract, and the code in column Code, write the model to Model2, and use a token-based language model with stemming, and a n-gram size of 3, use the following:

```
TrainLingPipe --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --model Model2 --language_model Porter Stemmer --
gram_size 3
```

To train from the database registered as datasource NYT1000Sample using the table Issues, where the ID is in column ID, the text in column Text, and the code in column 2-digit_topic, write the model to Model3, using the even samples, using a specified set of categories other than the default:

```
TrainLingPipe --datasource NYT1000Sample --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Text --
```

```
code_column "[2-digit_topic]" --model Model3 --use_even --categories 1
2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 19 20 21 23 24 26 27 29 30 31 99
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

Classification tools

The training tools take unlabled input and determine the classification based upon the supplied model.

ClassifySVM

The ClassifySVM tool will use an SVM classifier. To run this tool issue the command:

ClassifySVM*parameters*

Where *parameters* is a set of command parameters defined as follows:

-Xmx*nnnn*m

This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

--datasource *The datasource name – see discussion of datasource below*
--table_name *Table containing the training data*
--id_column *Column containing the ID*
--text_column *Column(s) containing the text*
--code_column *Column containing the code*
--output_code_col *Column where the computed code is written*
--model *Directory containing the model files*
--feature_dir *Write the feature files to this directory*
Default is SVM_Classification_Features
--result_dir *Write intermediate result files to this directory*
Default is SVM_Classification_Results
--use_even [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--use_odd [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--compute_major [TRUE|FALSE]
If true, the major code is computed from the minor code
Default is false
--remove_stopwords [TRUE|FALSE]
If true, remove common "stop words" from the text
Default is true
--do_stemming [TRUE|FALSE]
If true, pass all words through the Porter stemmer
Default is true

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter are `--datasource`, `--table_name`, `--id_column`, `--text_column`, `--code_column`.

Examples

To classify from the database registered as datasource `PAPolicy19992000` using the table `Newspaper_Data_1999_2000`, where the ID is in column `ID`, the text in column `Abstract`, and the code in column `Code`, use the `Model2`, and write the results to `SVM_Computer_Code` use the following:

```
ClassifySVM --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --output_code_col SVM_Computer_Code --model Model2
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

To classify from the database registered as datasource `NYTSaample` using the odd samples from the table `Issues`, where the ID is in column `ID`, the text in columns `Title` and `Text`, and the code in column `2-digit policy code`, write the results to `SVM_Computer_code` and write the model to `Model3`, use the following:

```
ClassifySVM --datasource NYTSample --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column "Title & ' ' &
Text" --code_column "[2-digit policy code]" --use_odd --
output_code_col SVM_Computer_Code --model Model3
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

Note: while concatenation of table columns is possible as described above, experience has shown that the performance is very slow.

ClassifyMALLET

The ClassifyMallet tool will classify against a previously generated model (either the Naïve Bayes classifier or the Maximum Entropy classifier). To run this tool issue the command:

`ClassifyMALLET` *parameters*

Where *parameters* is a set of command parameters defined as follows:

`-Xmx`*nnnn*`m`

This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

`--model` *The model file containing the trained classifier*
`--input_file` *File containing the training data*

--datasource *The datasource name – see discussion of datasource below*
--table_name *Table containing the training data*
--id_column *Column containing the ID*
--text_column *Column(s) containing the text*
--code_column *Column containing the code*
--use_even [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--use_odd [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--compute_major [TRUE|FALSE]
If true, the major code is computed from the minor code
Default is false

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter are --datasource, --table_name, --id_column, --text_column, --code_column.

Examples

To classify from the database registered as datasource PAPolicy19992000 using the table Newspaper_Data_1999_2000, where the ID is in column ID, the text in column Abstract, and the code in column Code, use the Model 2, and write the results to SVM_Computer_Code use the following:

```
ClassifyMALLET --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --output_code_col Bayes_Computer_Code --model Model 2
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

To classify from the database registered as datasource NYTSaample using the odd samples from the table Issues, where the ID is in column ID, the text in columns Title and Text, and the code in column 2-digit policy code, write the results to MaxEnt_Computer_code and write the model to Model 3, use the following:

```
ClassifyMALLET --datasource NYTSample --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column "Title & ' ' &
Text" --code_column "[2-digit policy code]" --use_odd --
output_code_col SVM_Computer_Code --model Model 3
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

Note: while concatenation of table columns is possible as described above, experience has shown that the performance is very slow.

ClassifyLingPipe

The ClassifyLingPipe tool will classify against a previously generated model (either the character-based N-Gram or token-based N-Gram). To run this tool issue the command:

ClassifyLingPipe *parameters*

Where *parameters* is a set of command parameters defined as follows:

-Xmx*nnnn*m

This is an optional parameter, but if specified it must be first. The value *nnnn* is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

--model *The model file containing the trained classifier*
--input_file *File containing the training data*
--datasource *The datasource name – see discussion of datasource below*
--table_name *Table containing the training data*
--id_column *Column containing the ID*
--text_column *Column(s) containing the text*
--code_column *Column containing the code*
--use_even [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--use_odd [TRUE|FALSE]
If true, use even numbered samples for training
Default is false
--compute_major [TRUE|FALSE]
If true, the major code is computed from the minor code
Default is false

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter are --datasource, --table_name, --id_column, --text_column, --code_column.

Examples

To classify from the database registered as datasource PAPolicy19992000 using the table Newspaper_Data_1999_2000, where the ID is in column ID, the text in column Abstract, and the code in column Code, use the Model 2, and write the results to NGram6_Computer_Code use the following:

```
ClassifyLingPipe --datasource PAPolicy19992000 --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column Abstract --
code_column Code --output_code_col NGram6_Computer_Code --model Model2
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

To classify from the database registered as datasource NYTSaample using the odd samples from the table Issues, where the ID is in column ID, the text in columns Title and Text, and the code in column 2-digit policy code, write the results to NGram6_Computer_code and write the model to Model 3, use the following:

```
ClassifyLingPipe --datasource NYTSample --table_name
Newspaper_Data_1999_2000 --id_column ID --text_column "Title & ' ' &
```

```
Text" --code_column "[2-digit policy code]" --use_odd --
output_code_col NGram6_Computer_Code --model Model3
```

Note the above should be typed as a single line. The text will wrap when the end of the console window is reached. Only enter a "return" (the enter key) after all input is typed.

Note: while concatenation of table columns is possible as described above, experience has shown that the performance is very slow.

Other Tools

FindClusters

The FindClusters program scans a classified table and determines whether entries with (nearly) identical text have been classified inconsistently. To determine closeness each entry is converted to its attribute vector using the same algorithm as TrainSVM. Each pair of vectors is then compared by computing the cosine of the angle between them. (This is done by computing the dot produce and dividing by the product of the magnitudes.) Pairs are considered similar if this value is greater than a specified threshold. The default threshold is 0.7 representing a 45° angle. Output is an HTML file containing a table that groups the clusters. Each line in the cluster, other than the first, is displayed showing the differences between it and the first line.

FindClusters *parameters*

Where *parameters* is a set of command parameters defined as follows:

-Xmx*nnnn*m

This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.

--datasource *The datasource name – see discussion of datasource below*
--table_name *Table containing the data*
--id_column *Column containing the ID*
--text_column *Column(s) containing the text*
--code_column *Column containing the code*
--cluster_column *Column containing the cluster flag.*
--output *Name of the output file*

Command options may be issued in any order. Parameters for which a default value is specified may be omitted. The only required parameter are --datasource, --table_name, --id_column, --text_column, --code_column.

Datasources

The datasource parameter must either specify an ODBC registered datasource or it must be the name of a text file that specifies a MySQL database. Note that the Access database may only be registered as an ODBC datasource using a 32-bit OS or in the 32-bit mode in a 64-bit OS.

For a MySQL database a text file must be created in the following format:

```
jdbc.driver:    com.mysql.jdbc.Driver
jdbc.url:       jdbc:mysql://host/database
jdbc.username:  username
jdbc.password:  password
```

Where host is the *host* for the MySQL database (e.g `localhost` if the MySQL database is running on the same computer as the TextTools or the name of the host computer)

References

The SVM classifier is based upon the Perl Script `run_svm.pl` http://www.purpuras.net/pac/run-svm-text.html. And implement the algorithm described in Purpura, S., Hillard D. "Automated Classification of Congressional Legislation." Proceedings of the Seventh International Conference on Digital Government Research. San Diego, CA.

The SVM calculations use SVM_Light http://svmlight.joachims.org.

The Naïve Bayes and Maximum Entropy classifiers are provided by the MALLET package see: McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

Contents of the TextTools_v0.10 directory

`Text Classification Tools v0.11.pdf`	This document as a .pdf file
`Text Classification Tools v0.11.docx`	This document as a Word 2007 document
`ClassifyFromDBUsingSVM`	Source for the SVM classification program. This is a NetBeans project directory
`ClassifyLingPipe.cpp`	Source Code for the LingPipe classification driver program
`ClassifyLingPipe.exe`	Executable for the LingPipe classification driver program
`ClassifyLingPipe.o`	Compiled binary for the LingPipe classification driver program
`ClassifyMallet.cpp`	Source Code for the Mallet classification driver program
`ClassifyMallet.exe`	Executable for the Mallet classification driver program
`ClassifyMallet.o`	Compiled binary for the Mallet classification driver program
`ClassifySVM.cpp`	Source for the SVM Classify driver program
`ClassifySVM.exe`	Executable for the SVM Classify driver program
`ClassifySVM.o`	Compiled binary for the SVM Classify driver program
`ClassifyUsingLingPipe`	Source for the LingPipe classification program. This is a NetBeans project directory
`ClassifyUsingMallet`	Source for the Mallet classification program. This is a NetBeans project directory

FileSort	Source code for the FileSort program. This is a NetBeans project directory.
FindClusters.cpp	Source for the FindClusters driver program
FindClusters.exe	Executable for the FindClusters driver program
FindClusters.o	Compiled binary of the FindClusters driver program
java	A copy of the Java run-time system.
java64	A copy of the 64-bit Java run-time system.
java-diff	Library containing methods to compute the differences between strings.
libstemmer	Library of stemming and stop-word removal methods for different languages.
lingpipe	The source and binary of LingPipe
LingPipeCommon	Source for the classes common to the LingPipe training and classification programs classification program. This is a NetBeans project directory
Mallet-0.4	The source and binary of Mallet with corrections to the ADABoost and ADABoostTrainer classes
Mallet-0.4.1	Netbeans project for Mallet – it references the Mallet-0.4 directory for the source and library files
MyMalletCommon	Source for the classes common to the Mallet training and classification programs classification program. This is a NetBeans project directory
MyUtil	Source for the classes common to the SVM and Mallet training and classification programs classification program. This is a NetBeans project directory
RunProgram.cpp	Common function used by the driver programs
RunProgram.o	Compiled binary for RunProgram
svm_classify.exe	Program to classify using SVM classifiers (from svm_light)
svm_learn.exe	Program to build SVM classifiers (from svm_light)
svm_light	Source for svm_light
TrainLingPipe.cpp	Source code for the LingPipe trainer driver program
TrainLingPipe.exe	Executable for the LingPipe trainer driver program
TrainLingPipe.o	Compiled binary for the LingPipe trainer driver program
TrainMallet.cpp	Source Code for the Mallet trainer driver program
TrainMallet.exe	Executable for the Mallet trainer driver program
TrainMallet.o	Compiled binary for the Mallet trainer driver program
TrainSVM.cpp	Source code for the SVM trainer driver program.
TrainSVM.exe	Executable for the SVM trainer driver program
TrainSVM.o	Compiled binary for the SVM trainer driver program
TrainUsingLingPipe	Source for the LingPipe training program. This is a NetBeans project directory
TrainUsingMallet	Source for the Mallet training program. This is a NetBeans project directory
TrainUsingSVM	Source for the SVM training program. This is a NetBeans project directory

Summary of Parameters

To the extent possible the parameters are common across the programs, with individual parameters where required. These are summarized in the following table:

Notes	Parameter	TRAIN			CLASSIFY			Other
		SVM	MALLET	LINGPIPE	SVM	MALLET	LINGPIPE	FindClusters
4	-Xmx*nnnn*m	X	X	X	X	X	X	X
1	--input_file	X	X	X				
1, 5	--datasource	X	X	X	X	X	X	X
1	--table_name	X	X	X	X	X	X	X
1	--id_column	X	X	X	X	X	X	X
1	--text_column	X	X	X	X	X	X	X
1	--code_column	X	X	X	X	X	X	X
2	--compute_major	X	X	X	X	X	X	
2	--use_even	X	X	X	X	X	X	
2	--use_odd	X	X	X	X	X	X	
3	--remove_stopwords	X	X		X			
3	--do_stemming	X	X		X			
	--model	X	X	X	X	X	X	
	--output_code_col				X	X	X	
	--feature_dir	X			X			
	--result_dir				X			
	--preserve_case		X					
	--trainer		X					
	--language_model			X				
	--gram_size			X				
	--categories			X				
	--cluster_column							X
	--output							X

Notes:

1) Either the --input_file to specify input from a text file or the combination --datasource, --table_name, --id_column, --text_column, and --code_column to specify input from a database must be specified.
2) Only applies to input from a database. These parameters are ignored if --input_file is specified.
3) These parameters take an optional argument that specifies the language specific stemming algorithm or stop word list. No test is made to see if the language options chosen are the same. If omitted the original Porter stemming algorithm is used for stemming and an English list of stop words developed by Chris Buckley and Gerard Salton is used. If "english" is specified then an improved stemming algorithm is used for English and the English stop words developed by Porter are used. To not do stemming or stop word removal specify "false".

4) This is an optional parameter, but if specified it must be first. The value ***nnnn*** is the number of megabytes of heap space that will be allocated. If this parameter is omitted, the default is 1000 (or 1 gigabyte). If this value is greater than 1500, then the 64-bit run-time system will be used. Only specify this parameter with a value greater than 1500 on a computer running a 64-bit Windows operating system.
5) The datasource parameter must specify either an ODBC registered datasource or be a text file to reference a MySQL database as described above.