Replication for "Separating the Wheat from the Chaff: Applications of Automated Document Classification Using Support Vector Machines"

Vito D'Orazio, Steven Landis, Glenn Palmer, and Philip Schrodt
October 29, 2013

Support for this programming was provided by National Science Foundation Grant SES-0719634 "Improving the Efficiency of Militarized Interstate Dispute Data Collection using Automated Textual Analysis" and SES-0924240 "MID4: Updating the Militarized Dispute Data Set 2002-2010" (Political Science). Redistribution and use of the perl programs in source and binary forms, with or without modification, are permitted under the terms of the GNU General Public License: `http://www.opensource.org/licenses/gpl-license.html`.

Basics

1. These are instructions for *replication* and not intended for new applications. For new applications, we suggest using the PreText software, available at `vitodorazio.weebly.com`, and its associated documentation.

2. The suite runs in a Unix environment, so these instructions assume basic familiarity with Unix. The phrase "Run" used here means "Run the command in a Unix terminal."

3. This documentation generally focuses on *how* to run the programs, with only limited information on *what* is being done. The programs produce an assortment of auxiliary files, and these are documented (more or less) internally in the individual programs.

4. Almost all of the programs are in perl, except the SVM classifier, which is available from `http://svmlight.joachims.org`. This may need to be compiled on whatever machine you are using—see instruction on that site. MID4 used SVM-Light, version 6.02.

5. Due to copyright issues, we cannot provide the LexisNexis documents used in our research. The following replication, however, assumes these documents exist and have been retrieved according to the process specified in the "LexisNexis Retrieval Documentation."

Required Directory Structure

The directory for the initial processing of the LexisNexis (LN) downloads, the /Data_YYYY directory, is to be structured as follows:

- AP_2002
- BBC_2002
- Power_Search_2002
- Search_String_2002

Each year's worth of downloads would be situated within directories labeled according. Each of these directories have the following internal structure:

April_2002 December_2002 January_2002 June_2002
May_2002 October_2002 August_2002 February_2002
July_2002 March_2002 November_2002 September_2002

In other words, the stories are in individual directories by month. The actual LN downloads are located within these directories. Two additional directories also need to be present in the same directory as /Data_YYYY:

1. Programs
    - contains the programs `genf.pl`, `dups.pl`, and `break.pl`
2. Process
    - This is where the formatted data files end up.

It is very important to ensure that the directory is structured properly because the shell script, `MID.format.sh`, has this directory structure hardcoded and will not know where to look if the directory is different.

Step One: Text Formatting

1. Place `MID.format.sh` in a data directory (e.g., AP_2002) and make sure it has the execute permission set (this can be accomplished by running `chmod +x MID.format.sh`)
2. Move to that data directory (e.g., `cd AP_2002`)
3. Execute the shell script
    - `./MID.format.sh fileprefix`
    - where `fileprefix` is a command line argument that will identify the output files for the individual data sources

Preparing the SVM Input

The formatted files are now in the Process directory. Move the following files to the directory:

1. `MID.CD.newcases.2b03.pl`
    - program that generates SVM input
2. `LN2001all.vocab.index.txt`
    - vocabulary for text representation
3. `CountryCodes.MIDNLP.txt`
    - actor dictionary for dyad identification
4. `MID.main.pbs`
    - pbs script for submitting the job on LION-X clusters
    - Note: `MID.pbs.sh` simple executes `perl MID.CD.newcases.2b03.pl`. Without access to LION-X clusters, executing that file directly will bypass `MID.main.pbs`

Run:

- `ls > class.filter.files`
- `qsub MID.main.pbs`

This produces an output file named `MIDSVM.new.output.txt`; typically you will want to manually change the file prefix, e.g., to LN2002. The program also produces an assortment of other files (rejects, domestic cases, dyad identifiers) which can be ignored in the rest of the processing—see the internal documentation.

After manually changing the file prefix to LN2002 (for example), move `LN2002.new.output.txt` to the directory where SVM-Light has been installed.

- e.g., `mv LN2002.* ../../svm*`

Running SVM

Decide on a file prefix for the combined output: this will be used in several of the programs and is denoted by ¡fileprefix¿. In the examples, the prefix is LN2002. To replicate SVM's classification, you need the files `svm_classify` and `2003.model`. `svm_classify` is part of SVM-Light, and `2003.model` is the training model developed for the MID4 classification and provided as part of our replication materials.

Run:

```
./svm_classify LN2002.new.output.txt 2003.model <fileprefix>.predict
```

After this has finished, copy `LN2002.predict` (assuming "LN2002" is the ¡fileprefix¿) and `LN2002.new.output.txt` to the Process directory. Also in the Process directory should be the perl scripts `merge.cd.results.pl` and `merge.LN.texts.pl`.

Post-SVM Processing

Next step is to go through the predictions and sort the positives and negatives. Inside the Process directory, run:

```
perl merge.cd.results.pl <fileprefix>
```

This will produce a file called `LN2002.positive.txt` which contains the IDs for the cases that were positively classified as containing MID-relevant reports, a file called `LN2002.negative.txt` which contains the negative cases (not used in further processing), and a file called `LN2002.pos.features`. This last file contains the features vectors for all documents classified as positive and is used in the transduction phase of the process.

After sorting the positives and negatives, these vectors are linked back to the news stories they represent. This step requires the file `MID.ISO.CODES.txt`. Run:

```
perl merge.LN.texts.pl <fileprefix>
```

Sorting the remaining files is accomplished by running:

```
perl sort.texts.pl <fileprefix>
```

Phase II: Transduction

Random Selection

To use transduction to classify the remaining documents, we randomly draw a sample of stories from those positively classified in the previous phase. This is done using `random250.r`:

1. Open R and set your working directory
2. Place `random250.r` and `summary.sorted.<fileprefix>` in your working directory
3. Modify `random250.r` so that it is reading the correct file
4. In R: `source(‘‘random250.r’’)`

This produces a summary file of 250 randomly drawn observations. The file is named `summary.random250.r`. The next step in the process is to read through these 250 stories and manually code them as either TRUE or FALSE. All TRUE stories should be placed into a .txt file called `<fileprefix>.random250.true.txt` and all FALSE stories should be placed into a .txt file called `<fileprefix>.random250.false.txt`.

Establishing Features Vectors

With 250 stories in 2 text files, we need to establish the features vectors that will be used by Joachim's SVM-Light program. Recall that in the Induction Phase we use `MID.CD.newcases.2b03.pl` to create features vectors for every single story we have downloaded, and we use the same program to establish the features vectors here. First,

however, we need to ensure that the input for `MID.CD.newcases.2b03.pl` looks exactly like it did in the Induction Phase. To do so, we reverse-format the text using the file `reverseformat.pl`.

Create a directory named "Learning" and place the following files inside:

1. `MID.CD.newcases.2b03.pl`
2. `LN2001all.vocab.index.txt`
3. `CountryCodes.MIDNLP.txt`
4. `reverseformat.pl`
5. `label_vector.pl`
6. `<fileprefix>.random250.true.txt`
7. `<fileprefix>.random250.false.txt`

To reverse-format, inside the Learning directory run:

```
ls <fileprefix>.random250.true.txt > files.list
perl reverseformat.pl files.list
mv reversed.0 true.positive.0
ls <fileprefix>.random250.false.txt > files.list
perl reverseformat.pl files.list
mv reversed.0 false.positive.0
```

To establish features vectors, inside the Learning directory run:

```
ls true.positive.0 > class.filter.files
perl MID.CD.newcases.2b03.pl
mv MIDSVM.output.txt true.SVMinput.txt
ls false.positive.0 > class.filter.files
perl MID.CD.newcases.2b03.pl
mv MIDSVM.new.output.txt false.SVMinput.txt
```

Labeling and Appending Features Vectors

Since every vector in both files is coded as a '0', or unknown as to whether or not the story is TRUE or FALSE, we need to go in and change the '0' to a '+1' if the vector corresponds to a TRUE story and a '-1' if the vector corresponds to a FALSE story. This is rather simple since all the TRUE vectors are in one text file and all the FALSE vectors are in another. `label_vector.pl` is a program that will read through a file of vectors and recode the '0' as either a '+1' or a '-1'. Note that '+1' or '-1' is hardcoded into `label_vector.pl` and will have to be modified by the user. Also hardcoded are

filenames (the program is short and the necessary changes are obvious). After this is done, the user will append the labeled features vectors to the unlabeled Phase I positive features vectors.

1. Update `label_vector.pl` to label TRUE vectors and output the file `<fileprefix>.true.features`
2. `perl label_vector.pl`
3. Update `label_vector.pl` to label FALSE vectors and output the file `<fileprefix>.false.features`
4. `perl label_vector.pl`
5. Open the two features files as well as `<fileprefix>.pos.features` in the Process directory and stack all the features vectors. Save this file as `<fileprefix>.trans.features`

It is important that the true and false features vectors be appended to the *bottom* of `<fileprefix>.pos.features`.

Classifying with Transduction Using SVM-Light

At this point we have created a file with all the positive features vectors from the Induction Phase as well as 250 randomly selected and manually coded features vectors. We now use transduction to classify all the positive features vectors from the Induction Phase based on our coding of the randomly selected 250.

1. copy `<fileprefix>.trans.features` to the svm_light directory
2. `./svm_learn <fileprefix>.trans.features <fileprefix>.trans.model`
3. `./svm_classify <fileprefix>.trans.features <fileprefix>.trans.model <fileprefix>.trans.predict`

The necessary file for further processing is `<fileprefix>.trans.predict`. Copy this file into the Process directory where the other files are located. `cd` to the Process directory and repeat the final post-SVM steps:

1. `mv <fileprefix>.pos.features <fileprefix>.trans.new.output.txt`
2. `perl merge.cd.results.pl <fileprefix>.trans`
3. `perl merge.LN.texts.pl <fileprefix>.trans`
4. `perl sort.texts.pl <fileprefix.trans`