PreText
Software for the Representation of Text

Vito D'Orazio
School of Economic, Political, and Policy Sciences
University of Texas at Dallas
vjdorazio@gmail.com

PreText: Software for the Representation of Text

PreText is a software package written in Perl for representing text as data. It is a user-friendly tool designed to encourage text analysis by making representation simple. In this first release of the software, I have stuck primarily to basic methods of representation. In the future (meaning when I finish my dissertation) I hope to expand the software to include more methods.

As I began working with text documents downloaded from *LexisNexis* for the Militarized Interstate Dispute project in 2009, I noticed very quickly that there are *a lot* of decisions to be made with respect to representation. Unfortunately, these decisions are often glossed over by researchers without being given much thought. PreText is a tool intended to help make these decisions more transparent and to give users options with respect to the method for representation.

Other text representation software exists, such as what is made available by Stanford's Natural Language Processing Group and several of the Apache projects. In Python, text representation can be done using the *Natural Language Toolkit*, and in R it can be done using the *tm* package (and others). Some of this existing software will undoubtedly be faster and have more features and options available. However, such tools can also be difficult to work with for researchers without a deep knowledge of computer programming. Some requires users to do a reasonably large amount of programming before their usage is even possible.

In the social sciences, requiring a knowledge of computer programming is a barrier to text analysis, and this is unfortunate. Hopefully, Pretext is simple enough to help reduce this barrier. In any event, I'm always looking to improve this software, so please contact me with any issues or suggestions at vjdorazio@gmail.com.

 Programming for the original files supported by National Science Foundation Political Science Program 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."

Contents

1 Text Representation

To conduct any statistical analysis on text documents, the first step is to represent those documents as data. There are many ways to go about this (Manning, Raghavan and Schutze, 2008; Forman, 2003, 2008; Guyon and Elisseeff, 2003), and experiments have shown that the method for representation is consequential for the results (Dy and Brodley, 2004; Monroe, Colaresi and Quinn, 2008). While there is no *correct* method, it is important that the decisions made are transparent and justified within the context of the study.

PreText utilizes a simple approach, and before diving into the software it is beneficial to get to know that approach. In a very basic example, imagine we have a set of four documents $C = \{1, 2, 3, 4\}$. Let's say the documents read as follows:

1. The Baltimore Ravens won Superbowl XLVII.
2. The 49ers lost Superbowl XLVII.
3. The 49ers were beating the Ravens in Superbowl XLVII
4. The Detroit Lions did not make the playoffs.

We can begin representation by establishing a dictionary D_C of all the unique words in C. Here, $D_C = \{$*The, Baltimore, Ravens, won, Superbowl, XLVII, 49ers, lost, beat, the, in, Detroit, Lions, did, not, make*$\}$. Each document can be coded for whether or not they contain a term in D_C. In a term document matrix, this would look like Table 1.

Table 1: Example Term Document Matrix

term	Doc 1	Doc 2	Doc 3	Doc 4
The	yes	yes	yes	yes
Baltimore	yes	no	no	no
Ravens	yes	no	yes	no
won	yes	no	no	no
Superbowl	yes	yes	yes	no
XLVII	yes	yes	yes	no
49ers	no	yes	yes	no
lost	no	yes	no	no
were	no	no	yes	no
beating	no	no	yes	no
the	no	no	yes	yes
in	no	no	yes	no
Detroit	no	no	no	yes
Lions	no	no	no	yes
did	no	no	no	yes
not	no	no	no	yes
make	no	no	no	yes

This example highlights the bag-of-words model with a binary weighting scheme. Bag-of-words refers to the fact that we are looking at each word individually. The binary weighting scheme refers to the simple yes/no coding for each word in each document. Once included in the dictionary, each term is referred to as a token.

PreText offers the options of named entity recognition and removal, stopword removal, stemming, document frequency thresholding, and the normalized term frequency and term frequency inverse document frequency approach. Each of these options involve one of the following aspects of representation:

1. Which terms or phrases are included in the dictionary?
2. What do those terms or phrases look like as tokens?
3. How are the tokens weighted for each document?

To build some intuition for text representation, each of the above options is applied to the example in Table 1.

Named entity recognition (NER) is difficult (Ratinov and Roth, 2009); it is also vital for many practical applications of text analysis. NER in PreText works by extracting 1-, 2-, 3-, and 4-grams from the text and searching for a match in a list of named entities. Although this is not the most efficient method for NER, it provides the user with an intuitive method for controlling the named entities that are recognized.

Keeping with the example shown in Table 1, let's say we have a list of named entities, L, where $L = \{SuperbowlXLVII \rightarrow game, 49ers \rightarrow NFC, Ravens \rightarrow AFC, Lions \rightarrow NFC\}$. Each named entity is mapped to a more general category, a feature that has a variety of uses and is discussed in more detail below. For now, it will suffice to say that these names would be removed from the dictionary.

Stopword removal strips a document of commonly found words with the idea being that these words provide little to no meaningful information for further analysis (Luhn, 1958; Rijsbergen, 1979). Examples of such words include *the*, *of*, and *if*. Although some have argued against this practice (Monroe, Colaresi and Quinn, 2008), it is a relatively common approach. Depending on the stopword selection, the words that would most-likely be stripped from the dictionary in the example would be *The*, *the*, *in*, *did*, and *not*.

Although the NER used here recognizes 1-, 2-, 3-, and 4-grams, the stopword removal only uses 1-grams, or unigrams. The dictionary then consists of the remaining unigrams in the document. Unigrams drop some information because phrases consisting of multiple words hold more semantic content than individual words (Papka and Allan, 1998; Spirling, 2012). Furthermore, individual words or phrases can have different meaning depending on their context (Blair, 1992, 2003). However, the use of individual words as features has been shown to peform quite well in several general applications (Grimmer, 2010; Hopkins

and King, 2010; Lewis, 1992; Yang and Pedersen, 1997) as well as in the conflict-specific setting for which PreText was originally developed (Schrodt, Palmer and Haptipoglu, 2008; D'Orazio et al., 2012).

The dictionary is not finished, however. Now that the unigrams have been extracted, we can stem each token for the purpose of making words with the same root look identical. For example, plural and singular variations should have the same token, as should past, present, and future-tenses. Depending on which stemming algorithm is used, a word like *beating* would probably be represented by the token *beat*.

The number of tokens in the dictionary can easily exceed one hundred thousand, which is a problem for many statistical methods. Therefore, we take an ax to it using document frequency thresholding (DFT). DFT consists of removing from the dictionary the tokens that do not appear in some desired percent of documents. It is essentially a hack of a feature selection method that just seems to work in many applications. Despite the fact that more sophisticated feature selection methods exist, DFT continues to be widely used (Dasgupta et al., 2007).

Finally, with the dictionary created and the tokens constructed, each token in each document receives a weight. Salton and Buckley (1988) and, more recently from Political Science, Lowe (2008) discuss various term weighting methods. The binary weighting scheme is a simple yes/no depending on whether or not the token appears in the given document. Normalized term frequency (NTF), which is supported by PreText, is calculated as the number of times the token appears divided by the number of times the most common token in that document appears. NTF may also be calculated as the number of times the token appears divided by the total number of tokens in the document. Let $N_{i,c}$ equal the number of times token i appears in document c:

$$ntf(i,c) = \frac{N_{i,c}}{max(N_{i,c})} \tag{1}$$

Term frequency inverse document frequency (TFIDF) is a slighted more complicated method which incorporates the number of times the token appears across documents as well as within a given document. Let N_C equal the total number of documents in the corpus (the document set). Let $N_{C,i}$ equal the number of documents term i appears in. Although there are variations, PreText calculates TFIDF as:

$$tfidf(i,c) = ntf(i,c) * log\frac{N_C}{1 + N_{C,i}} \tag{2}$$

At this point, the text has be represented as data. The data may be written in a variety of formats, three of which are supported by PreText and discussed below.

2 The Basics for Using PreText

2.1 Options and Features

The software contains the following options for representing text as data:

- named entity recognition
  - Uses Phil Schrodt's `CountryCodes` file
  - Also used for actor prediction
- stopword removal
- word stemming
  - Porter's stemming algorithm
- document frequency thresholding
- term weighting
  - normalized term frequency and term frequency inverse document frequency

2.2 Files

The following files comes with the PreText download:

1. `LN_mdata_1.pl`
2. `text_tokens.pl`
3. `term_doc.pl`
4. `CountryCodes.111214.txt`
5. `stopwords.txt`
6. `porter.pm`
7. `process.sh`

2.3 Directory Structure

PreText requires a directory structure where the folder names appears exactly like that in Figure 1. The software does not create the required structure, but it will check to make sure it exists. If it does not exist, users will get an error message.

- docs → LN downloads *or* `documents.txt`
- PreText → seven files listed
- process → empty
- output → empty

The `output` and `process` folders must be empty when the program is run. Inside the `docs` folder should be all the files downloaded from LexisNexis *or* a structured file called `documents.txt`. The required structure for the `documents.txt` file is described in the

Figure 1: Directory Structure

next section. Inside the `PreText` folder should be the seven PreText files.

2.4 Running PreText

PreText is intended to be run in the Unix environment. On a Mac, go to Applications → Utilities → Terminal. Make sure Perl is installed and check the version number by typing `perl -v`. I have run PreText with v5.12.4.

In the terminal, type `cd` followed by the name of a directory to change directories. Type `ls` to view the files in the present directory. `cd` to the location of the PreText archive you have downloaded and decompress it by entering `tar -xf PreText_v1`. In the Mac Finder, you could also just double click on the `.tar` archive. Place the seven files into the `PreText` directory.

- `cd` to the `PreText` directory with the seven files
- Enter: ./process.sh Arg1 Arg2 Arg3 Arg4 Arg5

2.5 Arguments

When PreText is run, users specify which options to use for representation (seen above as Arg1... Arg5). This is done through a series of arguments passed to the bash shell script `process.sh`.

1. `CountryCodes.111214.txt` or NO
    - Inputting the `CountryCodes.111214.txt` file will exclude all recognized actors from inclusion in the dictionary. It will also record as metadata the two primary actors in the document.
    - Enter NO if you do not want to remove these named entitites. Entering NO will speed up the processing time considerably.
2. `stopwords.txt`
    - `stopwords.txt` will remove the words listed in the `stopwords.txt` file from the dictionary.
    - Delete the words in `stopwords.txt` if you do not want to remove stopwords.

3. `porter.pm` or NO
    - Inputting `porter.pm` will utilize Porter's stemming algorithm to construct the word tokens.
    - Enter NO if you do not want to stem words.
4. Some number between 1 and 100 for document frequency thresholding.
    - This argument tells the program the percentage of tokens to *keep* through DFT. Entering 100, therefore, is equivalent to not using any DFT.
5. Input one or more of the following weighting schemes/formats: NTFLONG, NTFWIDE, NTFSVM, TFIDFLONG, TFIDFWIDE, TFIDFSVM.
    - Printing multiple formats at the same time is supported by separating the formats by a comma with NO spaces. For example, NTFLONG,TFIDFLONG.
    - The wide format gets very large very fast. I don't suggest using it unless it is essential.

2.5.1 Example Arguments

./process `CountryCodes.111214.txt` `stopwords.txt` `porter.pm` 10 `NTFSVM,TFIDFSVM`
./process `NO` `stopwords.txt` `NO` 90 `TFIDFLONG`

2.6 PreText Output

Output files will be placed in the `output` directory. This includes the desired data files, such as NTFLONG or TFIDFSVM, a tab-separated spreadsheet containing metadata associated with each document (including the actors), a file of structured documents called `documents.txt`, and a file of tokenized documents called `tokens.txt`.

After the program is run, the `process` directory will contain a log file for your records. The name of the log file is literally the time at which the program was run. Inside the file is some basic information about the run, including the input documents from the `docs` directory and the user-defined arguments.

3 Document Formatting

PreText works either directly with LexisNexis downloads, in which case it formats the documents for you, or with a set of documents that you have formatted yourself. The structure is very simple:

> Metadata.
> >>>>>>>>>>>>>>>>>>>>>>>
> The text to be represented as data.
> <<<<<<<<<<<<<<<<<<<<<<<

Metadata is data about the document rather than a representation of the text in the document. The metadata is useful for doing things like sorting the documents by actor or date. The metadata should contain a key for each document that uniquely identifies the document. If representing LN downloads, each file should have a unique name and PreText will give each document a key. Otherwise, the key should be included in the metdata section as "Key: docID." Other than a key, the metadata may contain any other pertinent information about the document such as the document title, the date, the source, the number of characters, etc. Only the text between the arrows will be represented as data.

One of the reasons the metadata is important is because it is included in a tab-separated file which can be read into programs such as Microsoft Excel. This spreadsheet file has many uses for working with the document set. For example, extracting the date as a piece of metadata is beneficial because, once in Excel, the documents can be sorted by date. Actors are also beneficial for the same reason. Once sorted, copying the key and searching the `documents.txt` file for that key is a simple way to work through the document set.

3.1 LexisNexis

LexisNexis is an incredible resource for all types of researchers. Assuming either you or your institution has access to LN, it is possible to download batches of documents for research purposes. LN allows you to selection numerous options for doing so, but here is simple example.

After getting to the search page, enter the information for your query: your search string, the time period, the sources, etc. LN allows many various specifications in their search string. I think of this phase as subsetting the documents you *might* care about

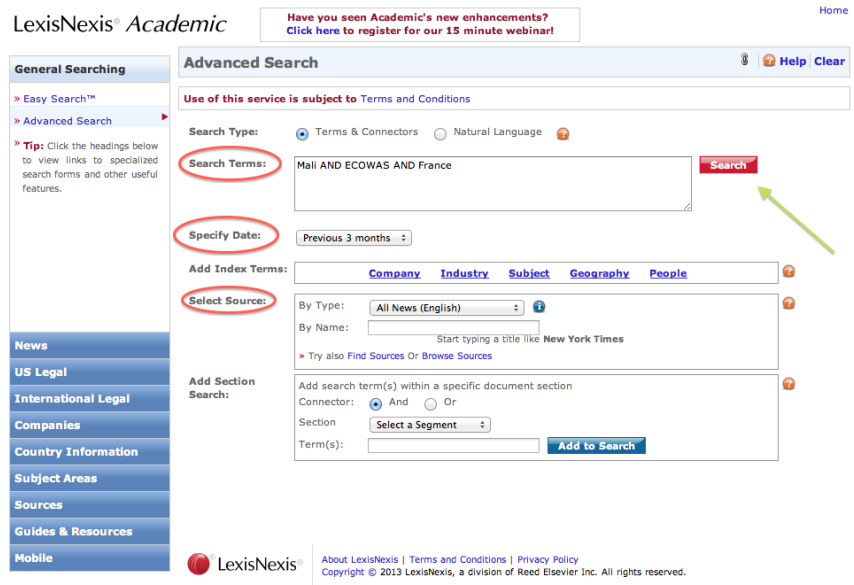

Figure 2: LN Interact

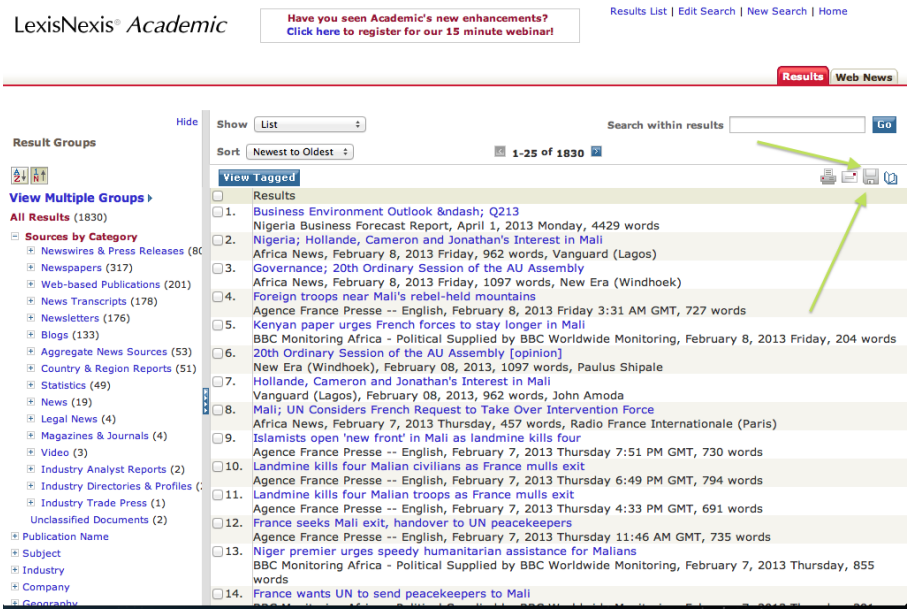

Figure 3: LN Interact

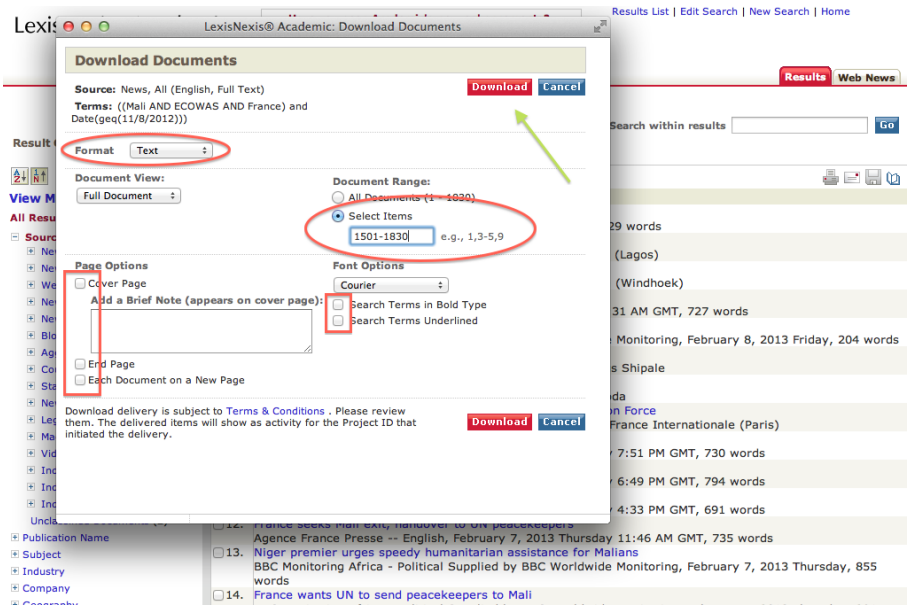

Figure 4: LN Interact

from the entire universe of documents stored by LN. Click the red search button in the upper-right, as shown in Figure 2.

LN will now deliver the results of your query. Clicking on the disk in the upper-right, as seen in Figure 3, will allow you to download the entire document set (or a portion of it). You can also select which documents to download by checking the boxes to the left. Clicking the disk brings up the screen shown in Figure 4.

LN then prepares the documents for download. We want to download the documents in the most basic format. Select the "text" format, uncheck all the boxes, and enter the number of the documents to be downloaded. Since only 500 documents can be downloaded at a time, you will often have to do this multiple times for the same search string. Once you have selected which documents to download, click the download button.

When LN is finished, you will see a screen similar to the one in Figure 5. Download the document set into a single text file by clicking the link. For use with PreText, move the downloaded file into the `docs` directory. You should also rename the file to something meaningful to you.

An example of a story in the downloaded file is shown in Figure 6. Notice, the formatting of LN provides easily-extractable metadata about this document. The source, the date, the title, and the body may all be extracted and saved.

After downloading as many LN documents as you want, all files should all be placed into the `docs` directory. To avoid confusion, each file should have a unique name. When PreText is run, the file `LN_mdata_1.pl` operates on this set of files. The individual documents from each files is copied into a single file where each document is structured

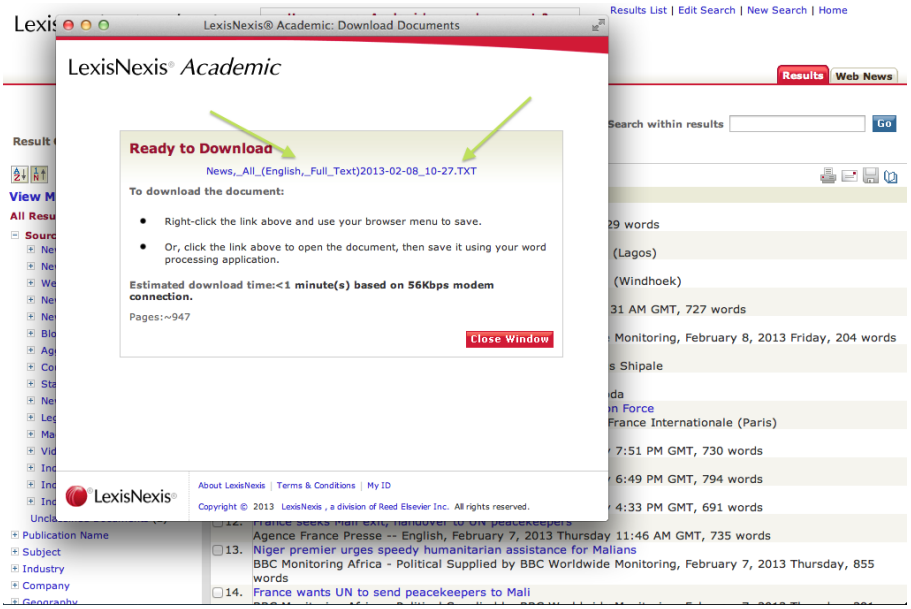

Figure 5: LN Interact

appropriately.

An example of what an LN document looks like after some formatting and metadata extraction can be seen in Figure 7. The information above the arrows will not be included when representing this document as data. What is included is all the text between the arrows, which serve as delimiters identifying the text to represent.

1501 of 1830 DOCUMENTS

Agence France Presse -- English

December 15, 2012 Saturday 9:15 PM GMT

Mali's new PM forms government

LENGTH: 251 words

DATELINE: BAMAKO, Dec 15 2012

Mali's new Prime Minister Diango Cissoko has formed his government, according to a decree read on state television Saturday, four days after he was named to the post when his predecessor Cheikh Modibo Diarra resigned under pressure from the country's ex-junta.

Cissoko told AFP on Friday he was working on the formation of a unity government representative of all parts of the troubled nation's society.

Defence Minister Colonel Yamoussa Camara, Foreign Minister Tieman Coulibaly and Economy Minister Tienan Coulibaly, who held posts in the previous administration, also joined the ranks of the new government, interim President Dioncounda Traore said in his decree.

Mali, once one of the region's most stable democracies, imploded as a Tuareg rebellion in its desert north prompted angry soldiers in March to overthrow the government.

As political paralysis took hold in Bamako, despite the setting up of an interim government earlier this year, the Tuareg and their Islamist allies continued a juggernaut which saw them seize all key northern towns and more than half the Malian territory.

The Islamists, tied to Al-Qaeda in the Islamic Maghreb, quickly sidelined their erstwhile Tuareg allies and took firm control in the region, where they have imposed brutal sharia law on residents.

The west African regional bloc ECOWAS is pushing for the deployment of a 3,300-strong intervention force to drive out the Islamists. It is backed by Western powers who fear the zone could become a haven for terrorists.

sd/gk/gd

LOAD-DATE: December 16, 2012

LANGUAGE: ENGLISH

PUBLICATION-TYPE: Newswire

Figure 6: LN Download

French forces clash with rebels near Gao in Mali
Key: 20130207-0-18-News,_All_(English,_Full_Text)2013-02-08_10-19.TXT
Date: 20130207
Source: Philippines News Agency
DATELINE: BAMAKO
Byline Not Found

>>>>>>>>>>>>>>>>>>>>>>
BAMAKO, Feb. 7 -- French soldiers clashed with armed rebels near Gao in northern Mali on Tuesday after the coalition forces of French and Malian troops had recaptured the major towns in their battle against the al-Qaida backed Islamist groups in the west African country, military sources and locals said.
France's Defense Minister Jean-Yves Le Drian confirmed the clashes in Paris on Wednesday and said it was an indication that "a real war" occurred in Mali.
"Yesterday there was a confrontation with some Islamists near Gao at a time when our soldiers who were being supported by the Malian forces were patrolling the towns that we have captured," the French defense minister said.
The town of Gao was recaptured by the French forces who were supported by the Malian and other African troops.
He said a residual group of the rebels fired rocket-launchers in Gao region.
"Therefore there's a real war. It's a war which today has enabled us to identify the hide-outs of the Islamists and we shall pursue them," Le Drian added.
After several months of diplomatic negotiation, France decided on Jan. 11 to intervene in Mali after the rebel groups captured the town of Konna in the Central region and were heading towards the capital, Bamako.
France has deployed 4,000 soldiers to Mali. The Economic Community of West African States (ECOWAS) members as well as some other African nations have pledged to dispatch anonther 4,000 to support the Malian government to regain ruling to the northern region that was under the rebels' control.
But the French defense minister said Wednesday that the French troops have reached the maximum. "We have 4,000 soldiers and we shall not go beyond that. But we shall start reducing them so that the African troops can take over," the minister said when he spoke on Europe 1 station.
"With 4,000 French and 4,000 African troops, we shall progressively start to hand over to African soldiers very soon," he added.
French Foreign Minister Laurent Fabius said in an interview with Le Metro daily newspaper that the French soldiers may start withdrawing from Mali as early as March.
"I think that as early as from March, if everything goes according to plan, there will be a reduction of the number of French troops," he affirmed.
However, Le Drian clarified in his radio interview that "the French soldiers will continue remaining in Mali until the country regains its territorial integrity and sovereignty (...) and until the moment when the Malian and African forces will be in a position to fully take over the operation against the Islamists." (PNA/Xinhua) Published by HT Syndication with permission from Philippines News Agency. For any query with respect to this article or any other content requirement, please contact Editor at htsyndication@hindustantimes.com
<<<<<<<<<<<<<<<<<<<<<<

Figure 7: A Formatted Document

4 Text Preparation

The next step is to prepare the text itself by constructing the dictionary. The Perl script called to prepare the text is `text_tokens.pl`. This script operates on he file `documents.txt`, which has either been produced by `LN_mdata_1.pl` or has been custom-created. The bash script executes this with the following command:

```
perl text_token.pl $1 $2 $3
```

Where $1 is `CountryCodes.111214.txt`, $2 is `stopwords.txt`, and $3 is `porter.pm`.

4.1 Named Entity Recognition

Phil Schrodt's `CountCodes.111214.txt` file is an XML file used for named entity recognition (NER).[1] NER can be very complicated, and much more advanced software exists for general NER than what is offered by PreText. However, what PreText does is pretty intuitive and can easily be manipulated to a set of named entities defined by the user.

Rather than recognize all named entities and have the user decide what to do with them, PreText uses an XML file to generate a list of named entities. `CountryCodes` is an example of the XML format used by PreText. It contains various political entitites, including countries, politicians, geographic features, and regions.

Figure 8 is a portion of what is contained in the `CountryCodes` file. Take note of the country code at the very top of the figure and of the nationalities listed in the middle-section of the figure. The names listed under "Nationality" (or any other category) is what PreText will search for. Every time one of those names is matched, the name is removed from further processing *and* the metadata for this particular document is updated to include another mention of "AFG," the country code.

Since this software was written for the Militarized Interstate Dispute project, the NER is based on mapping entities to a country name. However, this file may be easily manipulated to reflect something else, such as a mapping of businesses to their product type. All that would be necessary to do is change the "CountryCode" to something like "Technology" and then change the "Nationality" to something like "Businesses". Instead of listing Afghanistan nationalities, one would list technology businesses such as Google and Microsoft.

If NER is used, the output file `spreadsheet.tsv` will contain two columns of the two most-mentioned actors – not the most frequently matched names, but the more general

[1] Check `eventdata.psu.edu` for updated versions of this file.

```
<Country>
<CountryCode>AFG</CountryCode>
<CountryName>AFGHANISTAN</CountryName>
<COW-Alpha>AFG</COW-Alpha>
<COW-Numeric>700</COW-Numeric>
<FIPS-10>AF</FIPS-10>
<ISO3166-alpha2>AF</ISO3166-alpha2>
<ISO3166-numeric>004</ISO3166-numeric>
<ISO3166-alpha3>AFG</ISO3166-alpha3>
<Nationality>
    AFGHAN
    AFGANISTAN
    AFGHANESTAN
    AFGHANSITAN
    DOWLAT_E_ESLAMI_YE_AFGHANESTAN
    SOVIET_OCCUPIED_AFGHANISTAN
    AFGAHANISTAN
    AFGHANASTAN
    AFGANHISTAN
    AFGHINASTAN
    AFFGHANISTAN
    THE_GRAVEYARD_OF_EMPIRES
    AFGJANISTAN
    AFEGANISTAO
    CHAPARHAR
</Nationality>
<Capital>
    KABUL
</Capital>
</Country>
```

Figure 8: Country Codes Example

category – in the document. If left unmodified, geographic locations, cities, regions, etc. all count for the country in which they are located.

4.2 Stopwords

Stopwords are simply commonly used words. Often, these words are removed from further analysis because they are so common that they appear in high frequencies in every document, therefore not providing any useful information about individual documents.

The file `stopwords.txt` contains a list of some stopwords. Feel free to add or subtract words from this file as you like. Just note that any word contained in the file passed to PreText will be precluded from representation as data.

If you wish to not remove any stopwords, just delete everything in the `stopwords.txt` file and keep the input to PreText the same.

4.3 Stemming

Stemming consists of representing the root of a word as the token. Plurals, for example, appear identifical to the same word in singular form. Past, present, and future tenses may also appear identical. The extent and rules for stemming depends on the chosen stemming algorithm. PreText comes with the option of including the Porter Stemmer, an aggressive but effective stemming algorithm (Porter, 1980). Users may also enter NO and not do any stemming.

Many stemming algorithms exist in addition to the Porter Stemmer, and PreText can be easily modified to use a different stemming algorithm. If you have written a stemming algorithm or want to use one that somebody else has written, you need to know the following:

First, the name of the file should have the extension `.pm`. Second, the filename is used to identify the function, so the stemming function to be called should be the filename with the `.pm` extension. For example, the file `porter.pm` is included with PreText; `porter` is also the name of the function called.

Third, the function will get passed a single word to be stemmed. Using the Porter Stemmer, for example, the function `porter` will get passed a word such as “stems” and it will return “stem”. Therefore, any new stemming algorithm should be built to work with individual words and not sets of words or documents.

5 Representing Documents

The final step is to represent the text as data using the `term_doc.pl` file. As input, this files uses `tokens.txt`, which was written by `text_tokens.pl`. The bash script executes this script with the following command:

```
perl term_doc.pl $4 $5
```

Where $4 is the DFT threshold and $5 is the desired weighting scheme and format.

5.1 Document Frequency Thresholding

Document frequency thresholding (DFT) is a method of feature selection that subsets the tokens based on their frequency across documents. The threshold is user-defined, and therefore DFT may or may not be used. As the fourth argument passed to the bash shell script, enter a number between 1 and 100. Entering 1 will keep the top 1 percent of the tokens, measured by the number of documents the token appears in. Entering 100 will keep 100 percent of the tokens and is therefore equivalent to not using DTF.

5.2 Term Weighting and Output Formats

The fifth and final argument that is passed to the shell script is the term weight and output format. There are two term weighting methods: NTF and TFIDF. There are three data formats: LONG, WIDE, and SVM. Therefore, there are six possible combinations: NTFLONG, NTFWIDE, NTFSVM, TFIDFLONG, TFIDFWIDE, TFIDFSVM. Printing in multiple formats is supported by separating the names with a single comma and no space, such as: NTFLONG,TFIDFSVM.

The two term weighting schemes supported by PreText are the normalized term frequency and the term frequency inverse document frequency. Both formulas are described in the introductory section and are referred to as NTF and TFIDF.

There are three output formats supported by PreText. I refer to them as LONG, WIDE, and SVM.

LONG format prints the data in three tab-separated columns where the first is the document key, the second is the token, and the third is the weight – either NTF or TFIDF. Note that the LONG format with the NFT weighting scheme is the default format and a file of this type will always be produced when PreText is run. Text-data is often read into R using this format.

In the WIDE format, each row corresponds to a unique document. Each column corresponds to a token, and each cell corresponds to the weight for the given token in the

given document. The WIDE format gets very large very fast, and I do not suggest using it unless it is necessary. In the event that it is necessary, setting the DFT to a very low number is recommended.

The SVM format is the format for Thorsten Joachim's SVM^{Light} software. It is also the smallest output format of the three. Each row corresponds to a unique document and the tab-separated cells are written in the form termnumber:weight where termnumber is an arbitrary number referring to a particular token and the weight is the weight for that token in the given document. Each row in the SVM format ends with a $\#DocumentKey$. This is useful when using SVM^{Light}.

References

Blair, David C. 1992. "Information Retrieval and the Philosophy of Language." *The Computer Journal* 35(3):200–207.

Blair, David C. 2003. "Information Retrieval and the Philosophy of Language." *Annual Review of Information Science and Technology* 37(1):3–50.

Dasgupta, Anirban, Petros Drineas, Boulos Harb, Vanja Josifovski and Michael W. Mahoney. 2007. Feature Selection Methods for Text Classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

D'Orazio, Vito, Steven T. Landis, Glenn Palmer and Philip Schrodt. 2012. "Separating the Wheat from the Chaff: Applications of Automated Document Classification to MID.". Presented at the MidWest Political Science Meeting, 2011. Available at `http://vitodorazio.weebly.com/papers.html`.

Dy, Jennifer G. and Carla E. Brodley. 2004. "Feature Selection for Unsupervised Learning." *Journal of Machine Learning Research* 5:845–889.

Forman, George. 2003. "An Extensive Empirical Study of Feature Selection Metrics for Text Classification." *Journal of Machine Learning Research* 3:1289–1305.

Forman, George. 2008. Feature Selection for Text Classification. In *Computational Methods of Feature Selection*, ed. Huan Liu and Hiroshi Motoda. Boca Raton, FL: Chapman and Hall/CRC Press pp. 257–274.

Grimmer, Justin. 2010. "A Bayesian Hierarchical Topic Model for Political Texts: Measuring Expressed Agendas in Senate Press Releases." *Political Analysis* 18(1):1–35.

Guyon, Isabelle and Andre' Elisseeff. 2003. "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research* 3:1157–1182.

Hopkins, Daniel and Gary King. 2010. "A Method of Automated Nonparametric Content Analysis for Social Science." *American Journal of Political Science* 54(1):229–247.

Lewis, David D. 1992. Representation and Learning in Information Retrieval PhD thesis University of Massachusetts.

Lowe, Will. 2008. "Understanding Wordscores." *Political Analysis* 16(4):356–371.

Luhn, Hans Peter. 1958. "The Automatic Creation of Literature Abstracts." *IBM Journal of Research and Development* 2:159–165.

Manning, Christopher, Prabhakar Raghavan and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Monroe, Burt L., Michael P. Colaresi and Kevin M. Quinn. 2008. "Fightin' Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict." *Political Analysis* 16(4):372–403.

Papka, Ron and James Allan. 1998. Document Classification Using Multiword Features. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*.

Porter, Martin F. 1980. "An Algorithm for Suffix Stripping." *Program* 14(3):130–137.

Ratinov, Lev and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. CoNLL '09 Stroudsburg, PA, USA: Association for Computational Linguistics pp. 147–155.
URL: *http://dl.acm.org/citation.cfm?id=1596374.1596399*

Rijsbergen, C.J. van. 1979. *Information Retrieval*. London: Butterworth-Heinemann Press.

Salton, Gerard and Christopher Buckley. 1988. "Term Weighting Approaches in Automatic Text Retrieval." *Information Processing and Management* 24(5):513–523.

Schrodt, Philip A., Glenn Palmer and Mehmet Emre Haptipoglu. 2008. "Automated Detection of Reports of Militarized Interstate Disputes: The SVM Document Classification Algorithm." Presented at the Annual Meeting of the American Political Science Association, Toronto, Canada.

Spirling, Arthur. 2012. "U.S. Treaty Making with American Indians: Institutional Change and Relative Power, 1784–1911." *American Journal of Political Science* 56(1):84–97.
URL: *http://dx.doi.org/10.1111/j.1540-5907.2011.00558.x*

Yang, Yiming and Jan O. Pedersen. 1997. A Comparative Study on Feature Selection in Text Categorization. In *Machine Learning: Proceedings of the Fourteenth International Conference*. pp. 412–420.