Text as Data: Homework 2

Assigned 1/23, Due 1/31

In this homework assignment we're going to use a regular expression to search and cluster the press releases of two senators—Richard Shelby, a current Republican senator from Alabama, and Jeff Sessions, now the Attorney General, but previously Shelby's colleague. To make this comparison, we're going to download a bigger collection of Senate press releases and then focus on the releases from Shelby and Sessions.

We encourage you to spend some time processing these texts this week, because we will use this collection for the next homework assignment as well.

Downloading the Data

The press release collection are stored here:
`https://github.com/lintool/GrimmerSenatePressReleases`

Download the collection as a `.zip` file, unzip the file on your computer.

Creating a Document-Term Matrix

We're going to use the files from Richard Shelby and Jeff Sessions to make two different kinds of Document-Term Matrices. The first will consider only the 1000 most used unigrams, while the second (separate) DTM will use the 500 most common trigrams. To create the document-term matrices, use the following recipe.

1) Create two nested dictionaries for both the Shelby and Sessions press releases. The nested dictionary should contain, for each press release:

   - Month of release
   - Year of release
   - Day of release
   - Author (either Shelby or Sessions)
   - The text of the press release

   To create the nested dictionary:

   i) Use `os.listdir` to create lists of both the Sessions and Shelby press releases

   ii) The file names are formatted as `DayMonthYearAuthorNumber.txt`. Devise a parsing rule to extract the month, year, day, of the releases

   iii) Store all the information in a nested dictionary

2) We are first going to search for terms used in Shelby and Sessions' press releases. Specifically we are going to look for the terms `fire department`, `immigration`, and `nomination`

   - Using `re.findall` write a regular expression to identify `fire department`, `immigration`, and `nomination` in each press release. Be sure to catch all instances and only those instances.
   - Create a csv. The csv should have five columns, those should be: (1) author of press release (2) press release date (3) Number of times `fire department` appears in press release (4) Number of times `immigration` appears in press release and (5) Number of times `nomination` appears.
   - For each press release, record (1) author of press release, (2) press release date, (3) Number of times `fire department` appears, (4) Number of times `immigration` appears, and (5) Number of times `nomination` appears.
   - Load your data set into `R` and calculate the average number of times Shelby and Sessions use each term. What differences do you notice?

3) Next, we will find the 1000 most used unigrams and the 500 most used trigrams, after removing/simplifying a set of words

   i) discard punctuation, capitalization, and use `word_tokenize` to split the text on white space

   ii) Apply the Porter Stemmer to the tokenized documents.

   iii) Use the stop words from
   `‘http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-lis`
   Append to the list:

      - shelby
      - sessions
      - richard
      - jeff
      - email
      - press
      - room
      - member
      - senate

Apply the Porter Stemmer to this list of stop words and discard all stemmed stop words from the press releases.

iv) Form the list of trigrams using the `trigrams` function from `NLTK`

v) Use a python dictionary to count the number of times each unigram is used and a second dictionary to count the number of times each trigram is used. These should be counts over the *whole corpus* (that is, both senators' press releases).

4) Identify the 1000 unigrams used most often and the 500 most often used trigrams. If you're writing trigrams to a csv to analyze somewhere else, be sure to represent each `tuple` without commas.

5) Write a document-term matrix, where each row contains
`Speaker`, `Count`$_1$, `Count`$_2$, ..., `Count`$_{1000}$
for unigrams, and
`Speaker`, `Count`$_1$, `Count`$_2$, ..., `Count`$_{500}$
for trigrams.

Remember, if `foo` is a list, you can count the number of times `x` occurs with
`foo.count(x)`

6) Write the document term matrix for the unigrams and trigrams to separate .csv files. Remember that you'll need to reformat the trigram `tuples` so that you don't end up with extra commas in your column names. We recommend defining a function in python that takes a `tuple`, like
`'wabash', 'college', 'best'`
and converts it to
`wabash.college.best`

Clustering Methods

1) Using the `kmeans` function, create a plot of the `kmeans` objective function as the number of clusters varies from 2 to $N - 1$.

2) Apply K-Means with 6 clusters, being sure to use `set.seed` to ensure you can replicate your analysis

3) Label each cluster using computer and hand methods:

i) Suppose $\boldsymbol{\theta}_k$ is the cluster center for cluster k and define $\bar{\boldsymbol{\theta}}_{-k} = \frac{\sum_{j \neq k} \boldsymbol{\theta}_j}{K-1}$ or the average of the centers not k. Define

$$\text{Diff}_k = \boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_{-k}$$

Use the top ten words from Diff$_k$ to label the clusters

ii) Sample and read texts assigned to each cluster and produce a hand label

4) Using a mixture of multinomials (code provided), estimate a mixture with 6 components. Be sure to set your seed.

5) Compare the mixture of multinomial clustering to the kmeans clustering. Specifically create a `Confusion` matrix. Using the `table` function in `R`, compare the cluster assignments of `kmeans` and a mixture of multinomials. What do you notice?