Text as Data: Homework 1

May 22, 2019

In this homework assignment we're going to analyze the first presidential debate from the 2012 election and then assess the similarity of the statements

I recommend you use the code from Masha Krupenkin's python guide as a reference. It is found on the course github called `python_guide.docx`.

In particular, consult the web scraping with BeautifulSoup for details about scraping.

Problem 1

To analyze the debate, we first need to load the debate and parse the content. On the course `github`, you'll find the file `debate1.html`. Download the file and open it in a browser. We will use `BeautifulSoup` to parse the HTML file containing the debate transcript.

1) Load the webpage into `Python` and use `BeautifulSoup` to create a searchable version of the debate. What `HTML` tags can you use to identify statements?

2) Note that not all of the statements contain information about the speaker. Devise a rule to assign the unlabeled statements to speakers.

   For substantive reasons, we would like to define a single statement as any *uninterrupted* speech from a candidate. We'll say a candidate is interrupted when the transcript says that a new speaker has begun. In other words, cross talk doesn't count as an interruption.

3) With this rule in mind, create a nested `list`. The outer list will have an entry for each statement. Each statement will have its own list where the (1) the speaker is identified and (2) the text (and not the tags) of the statment is stored. Remember, some statements are split among several tags; these will need to be concatenated according to the rule you devised above. Remember to filter out notes about audience behavior. Create a parallel `list` that records the speaker of each statement.

- Now, we're going to create a document term matrix.

  i) For each statement assign it a unique number (place in debate)

ii) For each statement identify the speaker (e.g. candidate speaking)

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

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

b) Apply the Porter Stemmer to the tokenized documents.

c) Use the stop words from
`'http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-`
Apply the Porter Stemmer to this list of stop words and discard all stemmed stop words from the speeches.

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

e) 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, all statements).

iv) Write a document-term matrix, where each row contains
`Statement #, Speaker, Uni_Count`$_1$`, Uni_Count`$_2$`, ..., Uni_Count`$_{\texttt{un\_count}}$`, Tri_Count`$_1$`, Tri_Count`$_2$`, ..., Tri_Count`$_{\texttt{tri\_count}}$

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

v) Write the document term matrix to a csv file. Remember that you'll need to reformat the trigram `tuples` so that you don't end up with extra commas in your column names. Use the function from the class notes to take the `tuple`, like
`'wabash', 'college', 'best'`
and converts it to
`wabash.college.best`

Analyzing Document Similarity

Using the document-term matrix, let's compare the statements.

1) Create the following six square matrices:

i) Euclidean distance between documents

ii) Euclidean distance between documents with tf-idf weights

iii) Cosine similarity between documents

iv) Cosine similarity between documents with tf-idf weights

v) Normalize the rows of the document term matrix. For row i,

$$\boldsymbol{x}_i^* = \frac{\boldsymbol{x}_i}{\sum_{j=1}^{1500} x_{ij}}$$

Then apply the `Gaussian` kernel to the normalized matrices

vi) Use the same normalization, but now with tf-idf weights. Apply the Gaussian kernel.

2) Using the matrices, identify the most similar (smallest distance) and dissimilar (greatest distance) statements. Read the pairs of statements—do they appear to actually be similar? Which method appears to perform best?

3) Using cosine similarity, compare the average similarity of statements within and between candidates. Report the results in a table.

Bonus Question

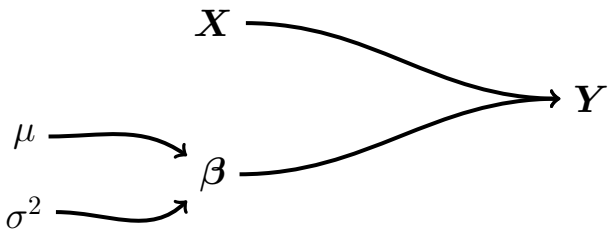
Problem 3: Probit Regression with a Prior

Suppose that we assume the following data generation process

$$\begin{aligned} Y_i &\sim \text{Bernoulli}(\pi_i) \\ \pi_i &= \Phi(\boldsymbol{X}_i\boldsymbol{\beta}) \\ \beta_j &\sim \text{Normal}(\mu, \sigma_j^2) \end{aligned}$$

with $\boldsymbol{X}_i = (1, x_i)$ for all i ($i = 1, \ldots, N$), $\boldsymbol{\beta} = (\beta_1, \beta_2)$, and $\Phi(\cdot)$ is the cumulative normal distribution function.

We might equivalently write a directed acyclic graph as,



This is very similar to the model described in class, but now we have added a *prior* on $\boldsymbol{\beta}$. This slightly alters the objective function:

$$\begin{aligned} p(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}) &\propto p(\boldsymbol{\beta}|\mu, \sigma^2) \times p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\beta}) \\ &\propto \prod_{j=1}^{2} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\beta_j - \mu)^2}{2\sigma^2}\right) \times \prod_{i=1}^{N} \Phi(\boldsymbol{X}_i\boldsymbol{\beta})^{Y_i}(1 - \Phi(\boldsymbol{X}_i\boldsymbol{\beta})^{1-Y_i} \end{aligned} \tag{1}$$

In this problem, we will examine how the prior on $\boldsymbol{\beta}$, and in particular the values we set for μ and σ^2, alters our inferences about $\boldsymbol{\beta}$.

a) Analytically, write out the $\log(p(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}))$.

b) In `R` create a function for the log of Equation 1.

c) Using the synthetic data and the optim guide from class, use `optim` to find $\widehat{\boldsymbol{\beta}}$ with $\mu = 0$ and $\sigma^2 = 1000$

d) Set $\mu = 1$ and then vary σ^2. Using a `for` loop, store estimates of how β_2 changes as you vary σ^2 from 10 to 0.01. Plot β_2 against σ^2 and describe what happens as σ^2 varies.