

Text as Data: Homework 3

Jeff Ziegler

August 17, 2017

In this homework we will analyze a collection of news stories from the New York Times from the November 1-3, 2004 (the day before, of, and after the 2004 general election). This data come from the New York Times Annotated Corpus and is for academic use only. We have done some preprocessing in order to simplify the homework tasks.

1 Preprocessing and Creating a Document-Term Matrix

- a) From the course github, download `nyt_ac.json`
- b) Using the `JSON` library in python, import the data. Use `type` to explore the structure of this data. How are this data organized?
- c) Extract the title and text from each story. Create an individual document for each story and write each of the files to a new directory
- d) Using the loaded `json` file, create a document term matrix of the 1000 most used terms. Be sure to:
 - Discard word order
 - Remove stop words
 - Apply the porter stemmer
- e) Include in your document-term matrix the *desk* from which the story originated, which we will include later

```
1 # import libraries
2 import json
3 import nltk
4 from nltk.corpus import stopwords
5 import re
6 import os
7 from urllib import urlopen
8
```

```

9 # a) download {nyt-ac.json} from gitHub
10 with open('Documents/Git/WUSTL.textAnalysis/nyt-ac.json') as data_file:
11     NYTjson = json.load(data_file)
12
13 # b) inspect the structure of the data
14 type(NYTjson)
15 # shows that json object is a list
16 # check the first elements as example
17 NYTjson[0].keys()
18 # two initial dictionaries (body, meta)
19 NYTjson[0]['body'].keys()
20 # within body, info like title and text
21
22 # c) write each story into new .txt file
23 for story in range(0, len(NYTjson)):
24     # give title based on order
25     with open('Documents/Git/WUSTL.textAnalysis/NYTstories/' +
26              str(story) + '.txt', 'w') as file:
27         # since we know where the relevant info is stored
28         # use dump to write json as string by key
29         file.write(json.dumps(NYTjson[story]['body']))
30
31 # check that files wrote properly
32 os.listdir('Documents/Git/WUSTL.textAnalysis/NYTstories/')

```

```

['0.txt',
 '1.txt',
 '10.txt',
 '100.txt',
 '101.txt',
 '102.txt',
 '103.txt',
 ...]

```

```

1
2 # d) task: Using NYTjson file , and create a DIM
3 # of the 1000 most used terms
4 # be sure to:
5 # discard word order
6 # remove stop words
7 # apply porter stemmer
8
9 # create function to use Porter stemmer
10 def porterStem(unstemmedList):
11     return [nlk.stem.PorterStemmer().stem(words) for words in unstemmedList]
12
13 # load a set of stop words from nlkt
14 # with the other stop word additions
15 stopWords = stopwords.words('english')

```

```

16 # apply Porter stemmer to stop words
17 stopWords = porterStem(stopWords)
18
19 # edit the text of pressReleases
20 for story in range(0, len(NYTjson)):
21     # create key in each NYT story with cleaned text
22     NYTjson[story][ 'body' ][ 'cleanText' ] = []
23     # remove capitalization
24     textTokens = NYTjson[story][ 'body' ][ 'body_text' ].lower()
25     # discard punctuation by removing non-word characters
26     textTokens = re.sub( '\W', ' ', textTokens )
27     # and apply Porter stem to tokenized story
28     textTokens = porterStem(nltk.word_tokenize(textTokens))
29     # remove stop words
30     textTokens = [x for x in textTokens if x not in stopWords]
31     # then append unigram tokens
32     NYTjson[story][ 'body' ][ 'cleanText' ] = textTokens
33
34 # now that we have cleaned text, need to create DIM
35 # create empty dictionaries to be filled with counts of unigrams
36 unigramDict = {}
37
38 # for each story
39 for story in range(0, len(NYTjson)):
40     # add counts to totals
41     for word in NYTjson[10][ 'body' ][ 'cleanText' ]:
42         if word not in unigramDict:
43             unigramDict[word] = 1
44         else:
45             unigramDict[word] += 1
46
47
48 # sort unigrams and trigrams into new lists
49 mostNunigrams = []
50
51 # create function to take the most used words
52 def extractTopN(topsList, mostNgrams):
53     # loop over dictionary and append new list
54     # by value, rather than key
55     # then sort list
56     return sorted(topsList, key=topsList.get, reverse=True)
57
58 # extract the most used unigrams
59 mostNunigrams = extractTopN(unigramDict, mostNunigrams)
60 # take only the top 1000
61 mostNunigrams = mostNunigrams[:1000]
62
63 # now that we have the most 1000 used unigrams
64 # create DIM matrix and write it to .csv
65 # task: we need to check whether each of the top 1000 words
66 # is in each NYT story, and count their frequency

```

```

67 # will iterate over each NYT story
68 # first , open up .csv writer
69 with open('Documents/Git/WUSTL-textAnalysis/NYTstoriesDTM.csv', 'wb') as f:
70     w = csv.writer(f)
71     # create header to be written to .csv as variable names
72     # 1st column is desk name, preceding columns
73     # represent top 1000 unigrams
74     csvHeader = mostNunigrams
75     csvHeader.insert(0, 'desk')
76     # write header first
77     w.writerow(csvHeader)
78     # then, we need to create counts of all the words
79 # in each document (NOT across authors or
80 # documents like the previous problem)
81 # so, for each story
82     for story in range(0, len(NYTjson)):
83         # create a clear row
84         rowEntry = []
85         # and give each unigram count in that story
86         for unigram in mostNunigrams:
87             rowEntry.append(NYTjson[story][ 'body' ][ 'cleanText' ].count(unigram))
88         rowEntry.insert(0, NYTjson[story][ 'meta' ][ 'dsk' ])
89         # write row
90         w.writerow(rowEntry)

```

Clustering Methods

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

```

1 # Problem 1
2
3 # load libraries and .csv files
4 library(foreign); library(fpc)
5 NYTstoryDTM <- read.csv("~/Documents/Git/WUSIL-textAnalysis/NYTstoriesDTM.csv"
6     , row.names=NULL)[, -2]
7
8 # a) task: use kmeans function to creat
9 # plot of the kmeans objective function as the
10 # number of clusters varies from 2 to N-1
11
12 # the data must be scaled for comparison
13 # we'll just use euclidean distance
14 euclideanNYTdtm <- as.matrix(dist(NYTstoryDTM[, -1], method = "euclidean"))
15
16 # to see which number of clusters is appropriate
17 # we want to vary the objective function of a range of K
18 # first , create k cluster points for 2 to N-1
19 # cluster points represent centroids for cluster j (c-j)
20 objectiveFunc <- (nrow(euclideanNYTdtm) - 1 * sum(apply(euclideanNYTdtm, 2,
21     var)))

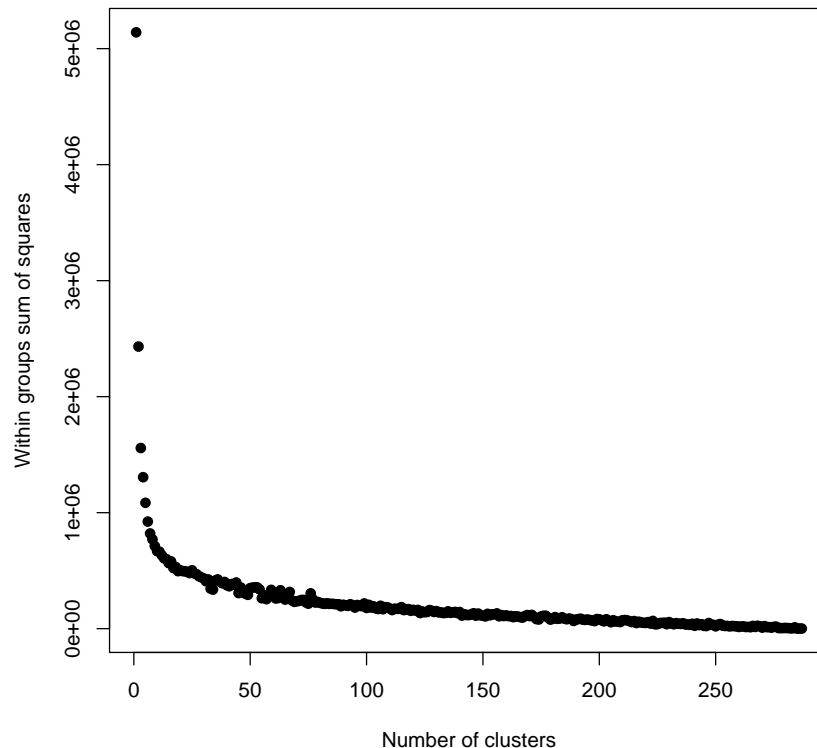
```

```

20 # then assign each object to the group w/ closest centroid
21 # after all objects have been assigned
22 # recalculate positions of the cluster centroids.
23 # repeat until centroids no longer move
24 for(i in 2:nrow(euclideanNYTdtm) - 1){
25   objectiveFunc[i] <- sum(kmeans(euclideanNYTdtm, centers = i)$withinss)
26 }
27 # last, plot objectiveFunc
28 pdf("~/Documents/Git/WUSIL_textAnalysis/HW3kmeansClusters.pdf")
29 plot(2:nrow(euclideanNYTdtm) - 1, objectiveFunc, pch=19, xlab="Number of
    clusters", ylab="Within groups sum of squares")
30 dev.off()

```

Figure 1: kmeans objective function as number of clusters varies.



- 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 θ_k is the cluster center for cluster k and define $\bar{\theta}_{-k} = \frac{\sum_{j \neq k} \theta_j}{K-1}$ or the

average of the centers not k . Define

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

```

1
2 # b) apply K-Means with 6 clusters
3 # set seed
4 set.seed(4); NYTclusters6 <- kmeans(euclideanNYTdtm, 6)
5
6 # c) task: label each cluster using computer and hand methods
7 # i) the final cluster centers are computed as the mean for each
   feature
8 # within each final cluster
9 # in other words, they reflect the characteristics of the "exemplar"
   document for each cluster
10
11 # so, if theta_k is the cluster center for cluster k
12 # mean(theta_{-k}) = (sum_(theta_j)/(K-1))
13 # or the average of the centers not k
14 # then Diff_k = theta_k - mean(theta_{-k})
15 # use the top ten words from Diff_k to label the clusters
16
17 # so, first make data frame of cluster centers
18 clusterCenters <- as.data.frame(NYTclusters6$centers)
19 # write a function that:
20 # finds theta_k, which is row k of clusterCenters
21 # and theta_{-k}, which is rows not-k of cluster centers divided by
   number of clusters - 1
22 # to create Diff_k
23 top10words <- function(k, clusters){
24   internalFunc <- function(k, clusters){
25     Diff_k <- as.data.frame(centers[k,] - colSums(centers[-(k),])/(
       max(clusters)-1))
26     # finally, take top 10 words
27     return(colnames(Diff_k[,order(Diff_k,decreasing=T)][1:10]))
28   }
29   # set up matrix for top 10 words
30   keywords <- matrix(NA, nrow=10, ncol=k)
31   # take top 10 words for each cluster
32   for (i in 1:k){
33     keywords[,i] <- internalFunc(i, clusterCenters)
34   }
35   return (keywords)
36 }
37 # execute top10words function
38 top10words(6, clusterCenters)

```

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

```

1 # ii) sample and read texts assigned to each cluster and produce a hand label
2 # sample 2 stories from cluster 1
3 exploreClusters <- function(seed, k, nDocs){
4   set.seed(seed)
5   for(i in sample(which(NYTclusters6$cluster==k),2)){
6     suppressWarnings(print(readLines(paste("~/Documents/Git/WUSTL_textAnalysis
7       /NYTstories/",i, ".txt", sep=' '))))
8   }
9 }
10 # these articles seem to be more human interest, rather than hard-hitting news
11 exploreClusters(5, 3, 2)

```

```

[1] "title": "Secret of Success Lies in Minnesota",
"body_text": "From the fourth play of the game, when a small
band of Giants seized an opportunity as the ball bounced free
on the Minnesota Vikings' turf, the Giants played Sunday as if
they were the only ones in on a secret... <truncated>
[1] "title": "Arts, Briefly", "body_text": "Managing Janet Jackson
Janet Jackson, below, may have caused a furor with her breast-baring
wardrobe malfunction during the halftime show at the Super Bowl,
but her album ''Damita Jo,'' issued by Virgin in March, didn't... <truncated>

```

2 Dictionary Classification Methods

- a) Download the list of positive (<http://www.unc.edu/~ncaren/haphazard/positive.txt>) and negative (<http://www.unc.edu/~ncaren/haphazard/negative.txt>) stop words from Neil Caren's website.
- b) Calculate a positive score and a negative score for each document and the difference between each score using the dictionaries

```

1 # Problem 2
2
3 # download list of positive and negative stop words
4 # create function to load sentimental dictionaries
5 def loadWords(type, stemmer):
6   # open url specifying positive or negative dictionary
7   url = urlopen('http://www.unc.edu/~ncaren/haphazard/' + type + '.txt').read
8   ()
9   # since they are in .txt files, we need to split each word
10  # create the unstemmed dictionary
11  unstemmedDict = url.split('\n')
12  # determine which stemmer should be used
13  # (1) Porter
14  if stemmer=='Porter':
15    # for each word in dictionary, stem

```

```

15     stemmedDict = [ nltk.stem.PorterStemmer().stem(word) for word in
    unstemmedDict]
16 # (2) Snowball
17 elif stemmer=='Snowball':
18     # for each word in dictionary, stem
19     stemmedDict = [ nltk.stem.SnowballStemmer('english').stem(word) for word in
    unstemmedDict]
20 # (3) Lancaster
21 elif stemmer=='Lancaster':
22     stemmedDict = [ nltk.stem.LancasterStemmer().stem(word) for word in
    unstemmedDict]
23 else:
24     stemmedDict = unstemmedDict
25 # return both stemmed and unstemmed dictionaries
26 return [unstemmedDict, stemmedDict]
27
28 # get basic positive and negative, porter stemmed dictionaries
29 positiveWords = loadWords('positive', stemmer='Porter').pop(1)
30 negativeWords = loadWords('negative', stemmer='Porter').pop(1)
31
32 # create function that will easily check how many words are in
33 # corresponding dictionary list
34 def wordCount(inputStatement, dictionaries):
35     return len([x for x in inputStatement if x in dictionaries])
36 # create function to pull necessary info from each statement
37 def storyInfo(story, documentContent):
38     # first, need to discard punctuation
39     removedPunctuation = re.sub('\W', ' ', str(NYTjson[story]['body']['cleanText
    ']))
40 # capitalization
41 removedCaps = removedPunctuation.lower()
42 # and tokenization
43 reducedStatements = nltk.word_tokenize(removedCaps)
44
45     # append documentContent with relevant info
46 documentContent.append({
47 # add to statementIter
48 'electionTiming': NYTjson[story]['meta']['publication_day_of_month'],
49 'desk': NYTjson[story]['meta']['dsk'],
50 # record the number of --- in statements w/ no punctuation, caps,
51 # and reduced tokens:
52 # non-stop words so that we can look at rates in analysis
53 'NonstopWords': len([x for x in reducedStatements if x not in stopWords]),
54     # number of words in each positive and negative using:
55     # (1) Porter stem
56     'NposPorter': wordCount([nltk.stem.PorterStemmer().stem(y) for y in
    reducedStatements], positiveWords),
57     'NnegPorter': wordCount([nltk.stem.PorterStemmer().stem(y) for y in
    reducedStatements], negativeWords),})
58
59 # create empty list to fill with statement info

```



```

60 storyCharacteristics = []
61 # execute story info extraction on each story
62 for i in range(0, len(NYTjson)):
63     # execute statementInfo function for each statement
64     storyInfo(i, storyCharacteristics)
65
66 # with data now assigned to dictionary
67 # write content to .csv to analyze in R
68 with open('Documents/Git/WUSTL_textAnalysis/storySentimentInfo.csv', 'wb') as
    f:
69     w = csv.DictWriter(f, fieldnames=('electionTiming', 'desk', 'NonstopWords',
    'NposPorter', 'NnegPorter'))
70     w.writeheader()
71     for item in storyCharacteristics:
72         w.writerow(item)
73

```

- c) How does the score change before and after the election? How does the score vary across desks?

```

1 # Problem 2
2
3 # c) How does the score change before and after the election?
4 # How does the score vary across desks?
5 # first, load data
6 storyInfo <- read.csv("~/Documents/Git/WUSTL_textAnalysis/storySentimentInfo.
    csv")
7 # then create new dataframe w/ word rate for each column
8 rateDF <- cbind(storyInfo[,c(1:2)], storyInfo[,c(4:5)]/storyInfo[,3][row(
    storyInfo[,c(4:5)])])
9 # see if positive word rate
10 posToneLM <- lm(NposPorter ~ as.factor(electionTiming) + desk, rateDF)
11 negToneLM <- lm(NnegPorter ~ as.factor(electionTiming) + desk, rateDF)
12 summary(posToneLM); summary(negToneLM)
13 # generally, it doesn't seem like there are more positive
14 # words over the course of the election (day before the election is reference)
15 # some desks like health and culture are more "upbeat"
16 # and then the editorial, national, and foreign ("hard news") is slightly
17 # more negative

```

Table 1: Regression results of sentimental change.

	<i>Dependent variable:</i>	
	NposPorter	NnegPorter
	(1)	(2)
Election day	−0.003 (0.002)	−0.001 (0.002)
Day after election	0.001 (0.002)	−0.002 (0.002)
Editorial Desk	0.013*** (0.004)	0.015*** (0.003)
Foreign Desk	−0.005 (0.003)	0.013*** (0.003)
Health and Fitness	0.016*** (0.006)	0.011** (0.005)
Metropolitan Desk	−0.001 (0.006)	0.010* (0.006)
National Desk	0.003 (0.002)	0.007*** (0.002)
Science Desk	0.002 (0.006)	0.001 (0.006)
Sports Desk	0.002 (0.003)	−0.0002 (0.003)
The Arts/Cultural Desk	0.011*** (0.003)	0.005** (0.003)
Constant	0.054*** (0.002)	0.033*** (0.002)
Observations	288	288
R ²	0.144	0.144

Note:

*p<0.1; **p<0.05; ***p<0.01

3 Supervised Learning with Naive Bayes

- Using the version of Naive Bayes outlined on slide 24 of lecture 14, write a function to estimate $p(C_k)$ and θ_k for an arbitrary collection of categories. Hint: to compute the probability of a document from a category, note you can work with the log of the probability equivalently.
- Let's focus on documents that came from Business/Financial desk and National Desk. Using leave-one out cross validation, calculate the accuracy of Naive Bayes to calculate the label.

c) Compare the performance of Naive Bayes to the performance of 2 of the following 3 algorithms using 10-fold cross validation:

- LASSO
- Ridge
- KRLS

How does Naive Bayes compare?