

Text Mining Trump Speeches Part 3: Topic Modeling, Trump fixated on Hillary



By Ted Kwartler, Contributing Data Science Writer - ODSC | 10/26/2016

Tags: Politics (<https://www.opendatascience.com/tag/politics/>) , **text analytics** (<https://www.opendatascience.com/tag/text-analytics/>)

At the start of this blog series I had no idea how the campaign dialogue would evolve. I am surprised how the discussion has distanced even farther from issues to personal affairs leaving many Republicans to abandon Donald Trump altogether. The dialogue has turned visceral with Donald Trump saying “the shackles are off” meaning he can say whatever he wants now. I find it both shocking, mildly entertaining and absolutely embarrassing! I guess democracy can be messy sometimes.

In this blog post I revisit the original 10 speeches from Donald Trump. If you missed the background check out the first (<https://www.opendatascience.com/blog/trump-and-clinton-speeches-step-1-text-mining/>) and second (<https://www.opendatascience.com/blog/trump-and-clinton-speeches-step-2-trump-is-a-negative-sophomore-and-clinton-is-a-positive-one/posts>). Within those posts you can download the speech files and follow along yourself. Keep in mind you may want to get more speeches since the campaign tone has *somehow* gotten more negative!

In this post I am going to make an interactive visual based on a topic model. Topic modeling is a method to observe word frequencies and fit a model to a predefined number of topics. A document or in our case a speech is a mixture of topics. While the results are a bit messy, in practice topic modeling is good for tagging documents in an archive and for generally understanding what a document is about without actually reading it.

I am going to make a d3 graphic from the **LDavis** package. The d3 graphic will let you interact with your topics and explore how different each is. Of course there are other compelling visuals that can be made with a topic model output but I hope this gets you started when doing it yourself.

No matter your party affiliation I hope you find the text mining approaches in the series to be informative. I wouldn't draw *too* many conclusions from this small sample set...I mean to only educate on some cool uses of R!

What is topic modeling?

Topic modeling finds word clusters within documents based on probability. In contrast to document classification or supervised learning, you not have to explicitly define document classes like “spam” versus “not spam”. Instead “K” topics are identified as the algorithm observes word frequency distributions among all the documents. After the topics are identified each document is given a probability for being part of a particular topic. When I do topic modeling I find that most often a document is a varied mixture of topics. For example a speech may end up being 20% about the economy, 20% about security and 60% attacking the other candidate.

Here is a fairly “non-mathy” explanation. Suppose this is a corpus of single sentence documents.

Show entries

Search:

Number	Document
1	I like to watch baseball and football on TV.
2	I watched baseball and Shark Tank yesterday.
3	Open source analytics software is the best.
4	R is an open source software for analysis.
5	I use R for baseball analytics.

Showing 1 to 5 of 5 entries

[Previous](#) [Next](#)

In this example we may have 2 topics which is a mix of word probabilities.

- Topic A: 30% baseball, 20% football, 15% watched, 10% TV...
- Topic B: 40% software, 10% open, 10% source, 10% analytics...

When reviewing the topic model terms you have to add your own expertise to make sense of the topic’s general meaning. So for this example

- Topic A: something to do with sports and also watching TV
- Topic B: something to do with open source analytics software

In this example it is easy to see the topic mixture once I apply my general understanding of the topic.

Show entries

Search:

Number	Document	Topic Mix
1	I like to watch baseball and football on TV.	100% Topic A – sports and TV
2	I watched baseball and Shark Tank yesterday.	100% Topic A – sports and TV
3	Open source analytics software is the best.	100% Topic B – open source analytics
4	R is an open source software for analysis.	100% Topic B – open source analytics
5	I use R for baseball analytics.	40% Topic A – sports and TV, 60% open source analytics

Showing 1 to 5 of 5 entries

[Previous](#) [Next](#)

Notice how the last document is a mixture? Document 5 mentions one of the sports, “baseball”, but not the “basketball” and no “TV”. So the document is not completely topic A. Also document 5 mentions “analytics” and “R” but not “open source”. Thus this document is a mixture of both A and B.

Your first topic model with Donald Trump speeches

Jumping in let’s load the libraries needed for topic modeling. There are multiple packages devoted to the subject. This blog post uses `lda` and **LDavis**. LDA stands for *Latent Dirichlet Allocation*. This is the method I described above observing word frequencies and then assigning probabilities for the topics to the documents. If you want a very technical but robust explanation check out David Blei’s Princeton paper here (<https://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf>). The **LDavis** package is a great for making an interactive and compelling topic model visualization. In terms of the text mining project workflow most often I use `qdap` and `tm` for preprocessing functions rather than regular expressions. The remaining `data.table`, `pbapply` and `rvest` are used for data manipulation.

```

>library(pbapply)
library(data.table)

library(lda)

library(LDAvis)

library(rvest)

library(qdap)

library(tm)

```

Similar to the previous blog post I load all candidate speeches using a custom function called `speech.read`. Assuming all candidate closed caption files are in the same working directory this function will read and organize the speeches. In the code below I specify a candidate string to search the folder for. Then a list of speeches is created using `fread` with `pbapply`. I like `fread` because it is faster than `read.csv` especially if you are dealing with a lot of text data. Lastly the function assigns the file names to each list element.

```

speech.read<-function(candidate){
  candidate.files<-list.files
  candidate.speeches<-pbapply(candidate.files,fread)
  names(candidate.speeches)<-candidate.files
  return(candidate.speeches)
}

```

In this example, the end result is a list with 10 elements. Each element is an individual speech data frame with the seconds the words were said and the words themselves. Below I am passing in the string “Trump” to create the list object `trump`.

```
trump<-speech.read('Trump')
```

For this analysis I do not need the “start” column containing seconds. One way to extract each list element’s “word” column is with `pluck`. I like it because it is a simple way to extract list elements by name. The code below uses `pluck` to create `all.trump`.

```
all.trump<-pluck(trump,'word')
```

The `all.trump` object is a list of 10 individual speeches. Applying other functions over list elements requires `lapply` or `pbapply`. Each of the following preprocessing steps is within `pbapply` so I can see the progress being made but either works. The text needs to be collapsed from a vector to a character string so `paste` and `collapse` are used in tandem. Next any contractions like “I’m” are changed to “I am” with `replace_contraction`. This aides in stop word removal and term aggregation. Next the base R function `tolower` is applied to the 10 speeches. Lastly `tm`’s `removePunctuation` is used.

```

all.trump<-pbapply(all.trump, paste,collapse=' ')
all.trump<-pbapply(all.trump,replace_contraction)
all.trump<-pbapply(all.trump,tolower)
all.trump<-pbapply(all.trump,removePunctuation)

```

Next I create a vector of stopwords. Stopwords are commonly occurring words that do not add much value to a text mining analysis. Words like “the” and “she” are used so often they can become clutter to an analysis. It is a good idea to customize your stopwords by concatenating words appropriate to the channel. Here I added words like “America” and “Americans.” Then the `stop.words` vector is passed to `removeWords`.

```

stop.words<c(stopwords('SMART'),'united','states','ohio','new','york',
'pennsylvania','america','american','americans','people','country','trump')

all.trump<-pbapply(all.trump,removeWords,stop.words)

```

I had to create `blank.removal` to avoid an error with `LDAvis`. When words are removed from the documents the `LDAvis` package still counts them as words but without any characters. This throws an error when making the visual. The function works by splitting each word by a space, then using `subset` to capture words with `nchar(x) > 0`. Finally, `blank.removal` uses `paste` and `collapse` to put the entire text back together without the empty “words.”

```

blank.removal<-function(x){
  x<-unlist(strsplit(x,' '))
  x<-subset(x,nchar(x)>0)
  x<-paste(x,collapse=' ')
}

```

```
all.trump<-pblapply(all.trump,blank.removal)
```

Next the `lda` package provides `lexicalize`. This returns a list of unique terms which represent the entire vocabulary used among the 10 speeches. It also returns each individual speech's use of a term from the vocabulary. Next, the `word.counts` function creates an array of words and their total use. For example the word "began" was used 3 times while "great" was said 141 times. Lastly the `doc.length` represents the total number of words in a speech.

```
trump.lex<-lexicalize(all.trump)

wc <- word.counts(trump.lex$documents, trump.lex$vocab)

doc.length<- document.lengths(trump.lex$documents)
```

This section defines the hyperparameters for the LDA model. `K` represents the number of topics to identify. It is a best practice to adjust this number so that the results can be meaningful. The `num.iter` is the number of iterations the model should perform. The specific model I am creating samples the text so the larger the iterations the more reliable the outcome. Next `alpha` represents the prior *document-topic* distributions and `eta` is the parameter setting the prior *topic-term* distributions.

```
k <- 6

num.iter <- 25

alpha <- 0.02

eta <- 0.02
```

Here I create a `set.seed` number for reproducibility. Using `lda.collapsed.gibbs.sampler` I pass in the documents, number of topics, vocabulary, iterations, alpha and eta.

```
set.seed(2016)

fit <- lda.collapsed.gibbs.sampler(documents = trump.lex$documents, K = k, vocab = trump.lex$vocab, num.iterations =
num.iter, alpha = alpha, eta = eta)
```

You can review the prototypical words in each topic with the code below. In the screen shot you can see the first topic is about jobs, the second about Mexico, walls, the third about "gonna," "rebuild" and "great" etc.

```
top.topic.words(fit$topics, 10, by.score=TRUE)
```

	[1,]	[2,]	[3,]	[4,]	[5,]	[6,]
[1,]	"jobs"	"make"	"gonna"	"clinton"	"good"	"november"
[2,]	"deals"	"big"	"great"	"state"	"win"	"world"
[3,]	"years"	"job"	"lot"	"dollars"	"vote"	"back"
[4,]	"remember"	"year"	"energy"	"money"	"work"	"millions"
[5,]	"deal"	"great"	"things"	"hillary"	"time"	"worse"
[6,]	"africanamerican"	"time"	"bad"	"cities"	"support"	"schools"
[7,]	"place"	"mexico"	"disaster"	"government"	"long"	"jobs"
[8,]	"made"	"wall"	"rebuild"	"children"	"percent"	"crime"
[9,]	"stop"	"economic"	"day"	"workers"	"talking"	"love"
[10,]	"trade"	"special"	"business"	"safe"	"word"	"immigration"

(<https://www.opendatascience.com/wp-content/uploads/2016/10/text-words.png>)

To create the interactive plot using `LDavis`, you need to estimate the document-topic distributions. This is done in the code snippet here to create the `theta` object and referencing `alpha`. You also need to estimate the topic term distributions, `phi`, using `eta`.

```
theta <- t(pblapply(fit$document_sums + alpha, 2, function(x) x/sum(x)))

phi <- t(pblapply(t(fit$topics) + eta, 2, function(x) x/sum(x)))
```

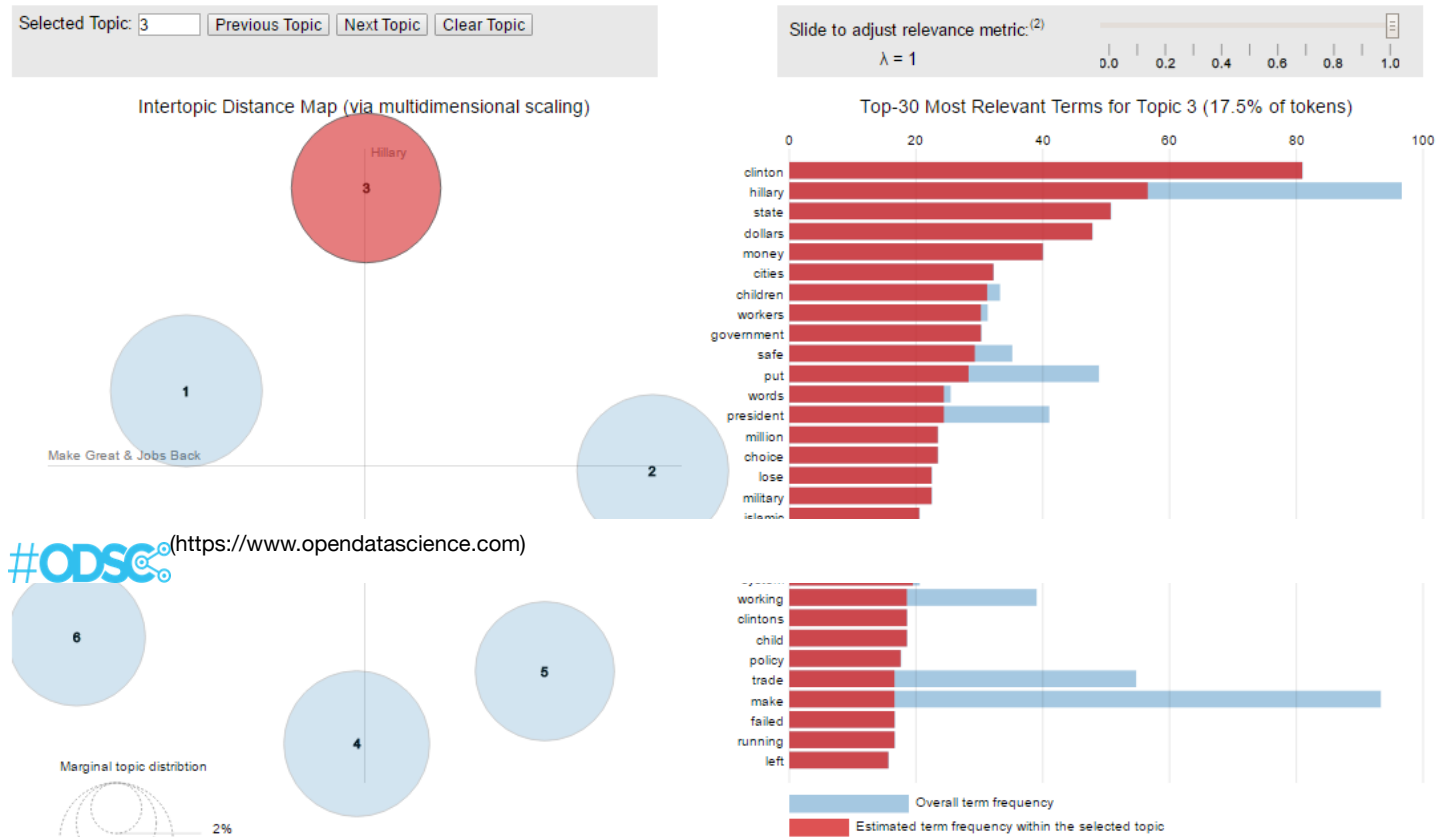
Almost done! Now you need to create the JSON object to be fed into the javascript visualization. I added the x and y labels after making sense of the output.

```
speech.json <- createJSON(phi = phi, theta = theta,
doc.length = doc.length
vocab = trump.lex$vocab
term.frequency = as.vector(wc)
plot.opts = list(xlab ="Make Great & Jobs Back", ylab = "Hillary"))
```

Now you can simply call `serVis` to open a browser with the topic models. You quickly create a github gist of the output by adding `as.gist = TRUE` within the function call. You can also save all the underlying files to a folder by adding `out.dir = 'visual'`. However your computer's security settings may keep the file from opening up properly from a local file. Instead I prefer to use my immediate R call or gist.

serVis(speech.json)

Here is a screen shot of the visual or a link to my gist (<http://bl.ocks.org/kwartler/raw/2551c894d7d8633bed4aeddffde7712b/>). When you navigate to the you can interact with the visual. As I reviewed the visual I saw how speeches along the Y axis showed decreasing mentions of Hillary. So the principal component analysis had some relationship to Trump's Hillary mentions. No topics below the X axis mentioned her. Along the X axis the topics had a relationship to either "good" or "great" and bringing back jobs. As you explore the visual you can start to pick up on similar topic relationships and differences. For example you can see how topic six is about Mexico, and a wall.




(<https://www.opendatascience.com/wp-content/uploads/2016/10/interactive-graph-TED.png>)The interactive LDA visualization showing topics and word frequencies.


Conclusion


So far you have seen how to create the interactive visualization with Trump speeches. You can go back and adjust the hyper parameters or perform the same analysis using Clinton's speeches. In the final installment of text mining on Trump and Clinton speeches, I will show you how to add another topic modeling visual using a treemap. Then we will create a pyramid plot comparing word usage between candidates.


©ODSC 2016

WHAT IS OPEN DATA SCIENCE?

 Username *

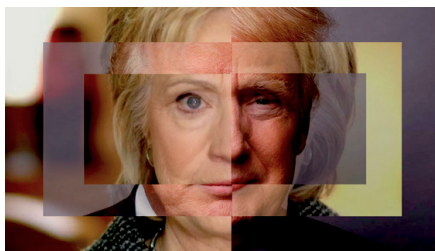
 Email *

 Password *

 Repeat Password *

Get Newsletter & Full Access

LATEST POSTS



Text Mining Trump
Speeches Part 3: Topic
Modeling, Trump fixated on
Hillary

(<https://www.opendatascience.com/blog/text-mining-trump-speeches-part-3-topic-modeling-trump-fixated-on-hillary/>)

10/26/2016



Introduction to Python

(<https://www.opendatascience.com/blog/introduction-to-python/>)

10/25/2016



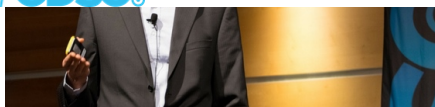
Three Challenges for Open
Data Science

(<https://www.opendatascience.com/conferences/three->

#ODSC

(<https://www.opendatascience.com>)

10/25/2016



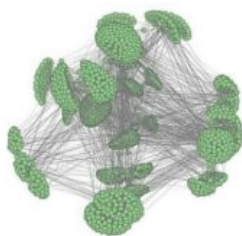
POPULAR POSTS



Riding on Large Data with
Scikit-learn

(<https://www.opendatascience.com/blog/riding-on-large-data-with-scikit-learn/>)

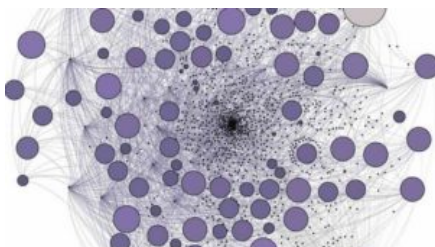
👁 9494 views



R or Python for data
science?

(<https://www.opendatascience.com/blog/r-or-python-for-data-science/>)

👁 9388 views



Data Science on Twitter

(<https://www.opendatascience.com/blog/data-science-on-twitter/>)

👁 5931 views

RELATED POSTS
