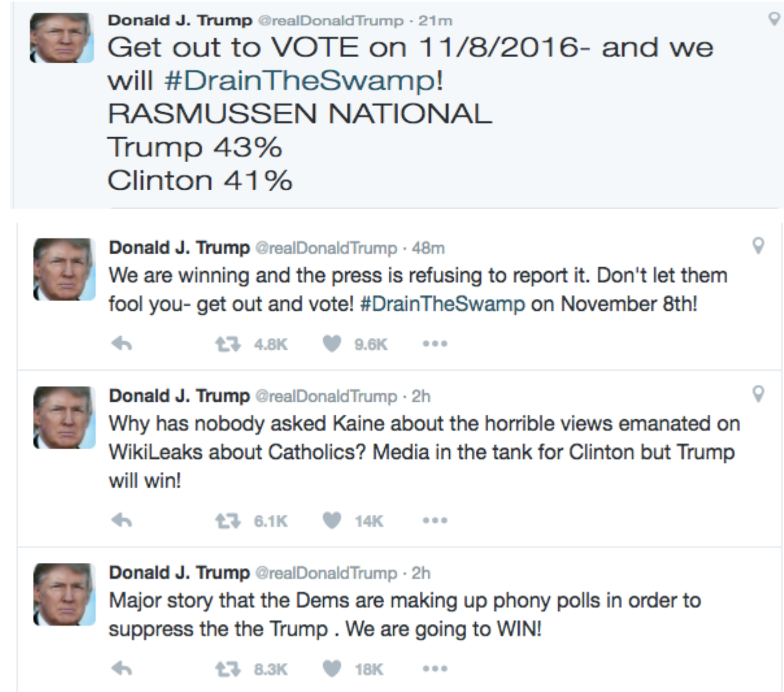
Text Mining



How can we analyze text?

- Unsupervised –
  - The documents are not labeled or classified
  - Look for clusters/groups of like documents
- Supervised –
  - The documents are labeled
  - Develop prediction method to predict label from text (and possibly other information)

Both approaches require us to transform the text into a quantitative form that we can analyze

Modified Tweets

- Get Out to VOTE - and we will win! Poll
- We are winning and the press is refusing to report it
- Media in the tank for Clinton but Trump will win!
- Dems are making phony polls to suppress the Trump. We are going to WIN!

Drop Stop Words (the, in, and, is, are..)

- Get Out **to** VOTE - **and** we **will** win! Poll
- We **are** winning **and the** press **is** refusing **to** report **it**
- Media **in the** tank **for** Clinton **but** Trump **will** win!
- Dems **are** making phony polls **to** suppress **the** Trump. We **are** going **to** WIN!

Drop Stop Words (the, in, and, is, are..)

- Get Out VOTE - we win! Poll
- We winning press refusing report
- Media tank Clinton Trump win!
- Dems making phony polls suppress Trump. We going WIN!

Stem Words
(winning to win; polls to poll; are to is)

- Get Out VOTE - we win! Poll
- We win**ning** press refus**ing** report
- Media tank Clinton Trump win!
- Dems mak**ing** phony poll**s** suppress Trump. We go**ing** WIN!

Stem Words
(winning to win; polls to poll; are to is)

- Get Out VOTE - we win! Poll
- We win press refus report
- Media tank Clinton Trump win!
- Dems mak phony poll suppress Trump. We go WIN!

Drop Punctuation
Convert to lower case

- **G**et **O**ut **VOTE** - we win**!** **P**oll
- **W**e win press refus report
- **M**edia tank **C**linton **T**rump win**!**
- **D**ems mak phony poll suppress **T**rump**.** **W**e go **WIN!**

Drop Punctuation
Convert to lower case

- get out vote we win poll
- we win press refus report
- media tank clinton trump win
- dems mak phony poll suppress trump we go win

Bag of words

- Unique words across all documents
- clinton dems get go mak media out phony poll press refus report suppress tank trump vote we win

Each Document is a word vector

	A	B	C	D	NumDocs
clinton	0	0	1	0	1
dems	0	0	0	1	1
get	1	0	0	0	1
go	0	0	0	1	1
…					
trump	0	0	1	1	2
vote	1	0	0	1	2
we	1	1	0	1	3
win	1	1	1	1	4
TOTAL	6	5	5	9	

What information do we lose with Word Vectors?

- Placement / juxtaposition of words
  - E.g., “damn” alone is typically a negative word, but “damn good” is a positive phrase; bigrams can help
- Meaning through punctuation –
  - use of exclamation point can distinguish between authors
- Small words
  - Some analyses focus on small words to identify authors

Similarity between documents

- Do documents use the same terms?
- Don’t care about common terms
- Want to control for the length of the document

Similarity between documents

- **Term frequency**: number of the words in a document are this term (nt of n)
- **Document frequency**: number of the documents that contain this term (Nd of N)
- Normalized vector:

V = nt/n * N/Nd) (term freq - inverse doc freq)

Or

V = nt/n * log((1+N)/ Nd)

	A	B	C	D	InvDocFrq
clinton	0	0	1/5	0	4
dems	0	0	0	1/9	4
get	1/6	0	0	0	4
go	0	0	0	1/9	4
…					
trump	0	0	2/5	1/9	2
vote	1/6	0	0	1/9	2
we	1/6	1/5	0	1/9	4/3
win	1/6	1/5	1/5	1/9	1
TOTAL	6	5	5	9	

tf-idf (log2)

	A	B	C	D	Log(idf)
clinton	0	0	2/5	0	4 -> 2
dems	0	0	0	2/9	4 -> 2
get	2/6	0	0	0	4 -> 2
go	0	0	0	2/9	4 -> 2
…					
trump	0	0	2/5	1/9	2 -> 1
vote	1/6	0	0	1/9	2 -> 1
we	2/30	2/25	0	2/45	4/3 -> 0.4
win	1/20	3/50	3/50	1/30	1 -> 0.3
TOTAL	6	5	5	9	

Distance between documents

- Dist(V, W) = ½(KL(V, AVG) + KL(W, AVG))
- where: KL stands for Kulback-Leibler measure

  KL(V, AVG) = sum(log(V/AVG) * AVG)
- and V = TF* IDF

Dissimilarity Matrix

	A	B	C	D
A	0	1.80	2.10	1.44
B	1.80	0	1.65	1.30
C	2.10	1.65	0	1.61
D	1.44	1.30	1.61	0

Multi-dimensional Scaling

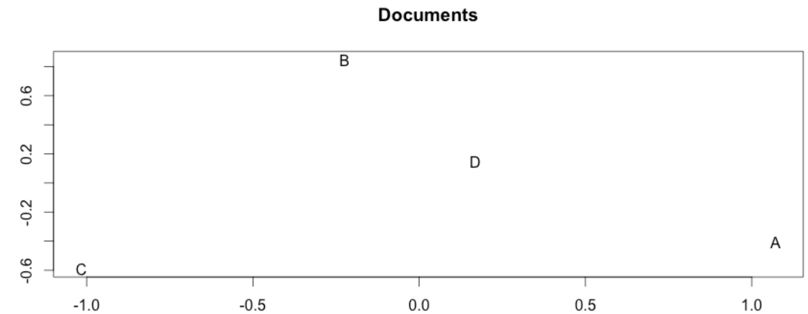
- Information visualization technique for high-dimensional data.
- Consider the matrix of dis-similarities above for the four documents.
- Assign locations in 2 dimensions so that the distances between documents is roughly preserved.

Example

	A	B	C
A	0	3	4
B	3	0	5
C	4	5	0

Could represent
as a triangle in
two dimensions

Our Documents



MDS

- Doesn’t produce unique representations of the data,
- Does give you the opportunity to compare objects (documents in our case)
- Look for clusters and gaps

Hierarchical clustering

- Build a binary tree that successively merges similar groups.
- This implies that we need a metric or measure of similarity between groups of points.
- There are various algorithms that can be used to create the binary tree.

Agglomerative Clustering

1. Start with each point in its own group.
2. Merge the two most similar groups.
3. Repeat step 2 until all groups have been merged into one

- Note that the similarity between two groups being merged at any stage must, by design, be decreasing because we merge less and less similar groups.

Measure of similarity between groups

- Single linkage: smallest distance between any point in one group and a point in the other group.
- Complete linkage: largest distance between any point in one group and a point in the other group.
- Average linkage: average distance between each point in one group and every point in the other group

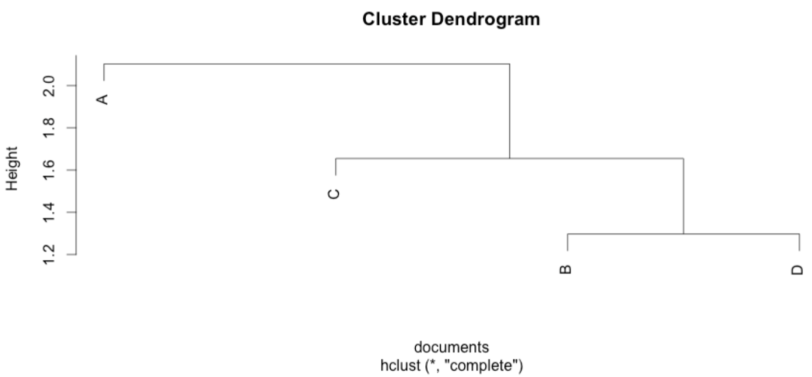
- Single linkage tends to result in chaining, where you successively add on one point to a group

- Complete linkage tends not to merge close groups when one point in one group is far from the other group.

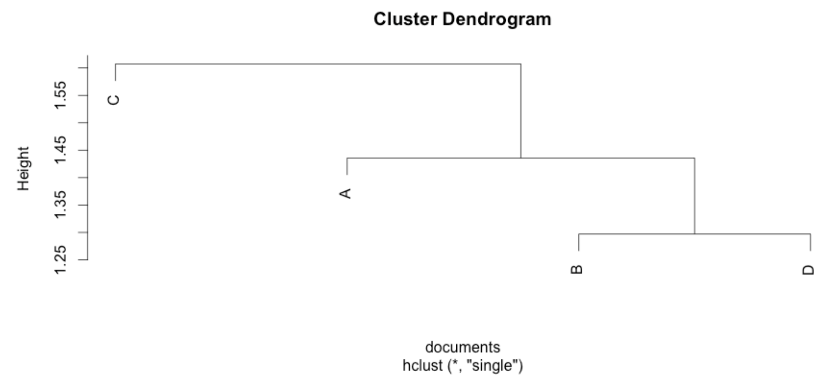
Dendrogram

- Useful visualization of the clustering process.
- Typically the tree is drawn such that the heights of the branches proportional to the dissimilarity between the two groups.
- This visual helps you see where a good place to "cut" the tree might be and create clusters

Complete linkage



Single linkage



Dendrogram

- Different definitions of similarity can give very different trees.
- The algorithm imposes a hierarchy on a set of data, even if there isn't one.

Your Turn

State of the Union addresses

State of the union speeches

- Use readLines() to read in the speeches
- Return value: character vector with one element/character string per line in the file
- Regular expressions to find ***
- Use *** to identify the date of the speech
- Use regular expressions to extract the year
- Use regular expressions to extract the month
- Use *** to extract the name of the president

State of the union speeches

- Chop the speeches up into a list there is one element for each speech.
- Each element is a character vector.
- Each element of the vector is a character string corresponding to a sentence in the speech

Word Vectors

- Eliminate apostrophes, numbers, and the phrase: (Applause.) from the text.
- Make all the characters lower case.
- Split the sentences up where there are blanks and punctuation
- Drop any empty words that resulted from this split
- Load the library Rstem and use the function wordStem() to stem words

- Find the bag of words
- Create a word vector for each speech
- Normalize the word vectors to get term frequencies

Analysis

- Exploratory analysis of the data:
  - Number of sentences, long words, political party

- Multidimensional scaling

- Hierarchical clustering