Quantitative text analysis: Describing and Comparing Text

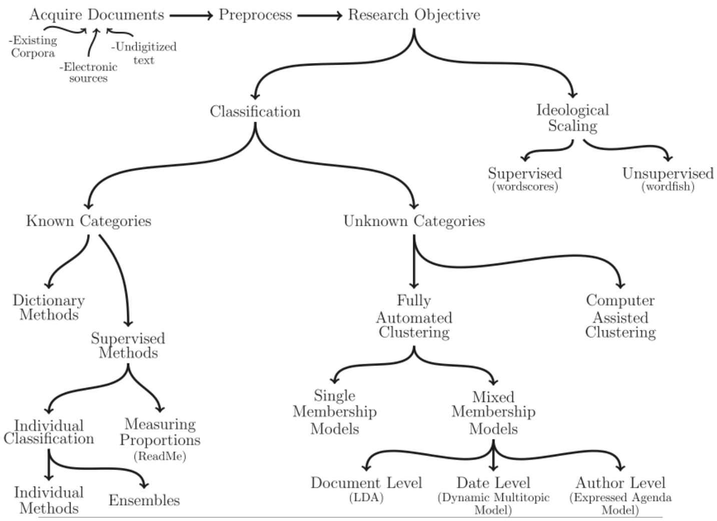
Blake Miller

MY 459: Quantitative Text Analysis

March 2, 2021

Course website: lse-my459.github.io

Overview of text as data methods



Outline

- Describing a single document
- Comparing documents
  - Similarity metrics: cosine, Euclidean, Jacquard, edit distance
  - Clustering methods: k-means clustering, hierarchical clustering

Quantities for describing a document

- **Length** in characters, words, lines, sentences, paragraphs, pages, sections, chapters, etc.
- **Word (relative) frequency** counts or proportions of words
- **Lexical diversity** (At its simplest) involves measuring a *type-to-token ratio* (TTR) where unique words are types and the total words are tokens
- **Readability statistics** Use a combination of syllables and sentence length to indicate "readability" in terms of complexity

Outline

- Describing a single document
- Comparing documents
  - Similarity metrics: cosine, Euclidean, Jacquard, edit distance
  - Clustering methods: *k*-means clustering, hierarchical clustering

Comparing documents

- The idea is that (weighted) features form a vector for each document, and that these vectors can be judged using metrics of similarity
- A document's vector for us is simply (for us) the row of the document-feature matrix
- The question is: how do we measure distance or similarity between the vector representation of two (or more) different documents?

Characteristics of similarity measures

Let A and B be any two documents in a set and $d(A, B)$ be the distance between A and B.

1. $d(x, y) \geq 0$ (the distance between any two points must be non-negative)
2. $d(A, B) = 0$ iff $A = B$ (the distance between two documents must be zero if and only if the two objects are identical)
3. $d(A, B) = d(B, A)$ (distance must be symmetric: A to B is the same distance as from B to A)
4. $d(A, C) \leq d(A, B) + d(B, C)$ (the measure must satisfy the triangle inequality)

Euclidean distance

Between document A and B where j indexes their features, where y_{ij} is the value for feature j of document i

- Euclidean distance is based on the Pythagorean theorem
- Formula

$$\sqrt{\sum_{j=1}^{j}(y_{Aj} - y_{Bj})^2} \tag{1}$$

- In vector notation:

$$\|\mathrm{y}_A - \mathrm{y}_B\| \tag{2}$$

- Can be performed for any number of features J (where J is the number of columns in of the dfm, same as the number of feature types in the corpus)

Cosine similarity

- Cosine distance is based on the size of the angle between the vectors
- Formula
$$\frac{\mathrm{y}_A \cdot \mathrm{y}_B}{\|\mathrm{y}_A\| \|\mathrm{y}_B\|} \tag{3}$$
- The $\cdot$ operator is the dot product, or $\sum_j y_{Aj} y_{Bj}$
- The $\|\mathrm{y}_A\|$ is the vector norm of the (vector of) features vector y for document A, such that $\|\mathrm{y}_A\| = \sqrt{\sum_j y_{Aj}^2}$
- Nice property: independent of document length, because it deals only with the angle of the vectors
- Ranges from -1.0 to 1.0 for term frequencies, or 0 to 1.0 for normalized term frequencies (or tf-idf)

Jacquard coefficient

- Similar to the Cosine similarity
- Formula

$$\frac{\mathrm{y}_A \cdot \mathrm{y}_B}{\|\mathrm{y}_A\| + \|\mathrm{y}_B\| - \mathrm{y}_A \cdot \mathrm{y}_B} \tag{4}$$

- Ranges from 0 to 1.0

Edit distances

- Edit distance refers to the number of operations required to transform one string into another for strings of equal length
- Common edit distance: the Levenshtein distance
- Example: the Levenshtein distance between "kitten" and "sitting" is 3
  - kitten $\rightarrow$ sitten (substitution of "s" for "k")
  - sitten $\rightarrow$ sittin (substitution of "i" for "e")
  - sittin $\rightarrow$ sitting (insertion of "g" at the end).
- Hamming distance: for two strings of equal length, the Hamming distance is the number of positions at which the corresponding characters are different
- Not common, as at a textual level this is hard to implement and possibly meaningless

Other uses, extensions

- Used extensively in information retrieval
- Summary measures of how far apart two texts are – but be careful exactly how you define "features"
- Some but not many applications in social sciences to measure substantive similarity — scaling models are generally preferred
- Can be used to generalize or represent features in machine learning, by computing similarities between textual (sub)sequences without extracting the features explicitly (as we will do in a second)

Outline

- Describing a single document
- Comparing documents
  - Similarity metrics: cosine, Euclidean, Jacquard, edit distance
  - Clustering methods: *k*-means clustering, hierarchical clustering

The idea of "clusters"

- Essentially: groups of items such that inside a cluster they are very similar to each other, but very different from those outside the cluster
- "unsupervised classification": cluster is not to relate features to classes or latent traits, but rather to estimate membership of distinct groups
- groups are given labels through post-estimation interpretation of their elements
- typically used when we do not and never will know the "true" class labels
- issues:
  - how many clusters?
  - which features to include?
  - how to compute distance is arbitrary

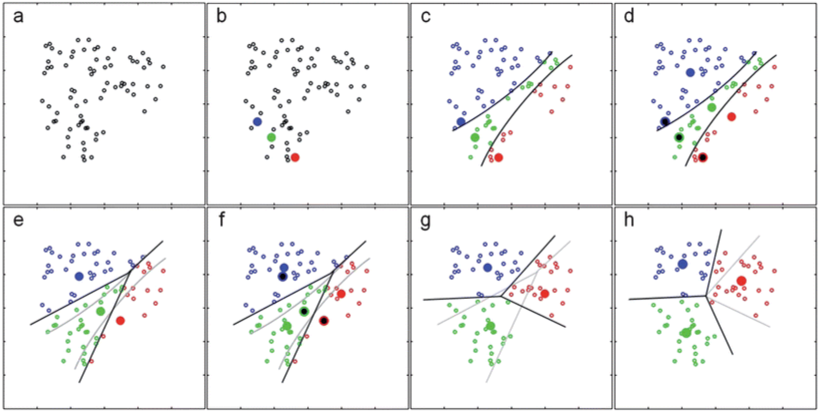
k-means clustering

- Essence: assign each item to one of *k* clusters, where the goal is to minimised within-cluster difference and maximize between-cluster differences
- Uses random starting positions and iterates until stable
- *k*-means clustering treats feature values as coordinates in a multi-dimensional space
- Advantages
  - simplicity
  - highly flexible
  - efficient
- Disadvantages
  - no fixed rules for determining *k*
  - uses an element of randomness for starting values

algorithm details

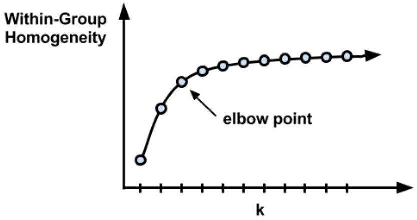
1. Choose starting values
    - assign random positions to k starting values that will serve as the “cluster centres”, known as “centroids” ; or,
    - assign each feature randomly to one of k classes
2. assign each item to the class of the centroid that is “closest”
    - Euclidean distance is most common
    - any others may also be used (Manhattan, Minkowski, Mahalanobis, etc.)
    - (assumes feature vectors are normalized within document)
3. update: recompute the cluster centroids as the mean value of the points assigned to that cluster
4. repeat reassignment of points and updating centroids
5. repeat 2–4 until some stopping condition is satisfied
    - e.g. when no items are reclassified following update of centroids

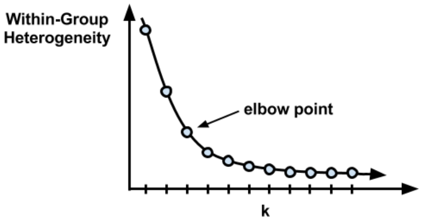
k-means clustering illustrated



choosing the appropriate number of clusters

- very often based on prior information about the number of categories sought
  - for example, you need to cluster people in a class into a fixed number of (like-minded) tutorial groups
- a (rough!) guideline: set $k = \sqrt{N/2}$ where N is the number of items to be classified
  - usually too big: setting k to large values will improve within-cluster similarity, but risks *overfitting*
- "elbow plots": fit multiple clusters with different k values, and choose k beyond which are diminishing gains





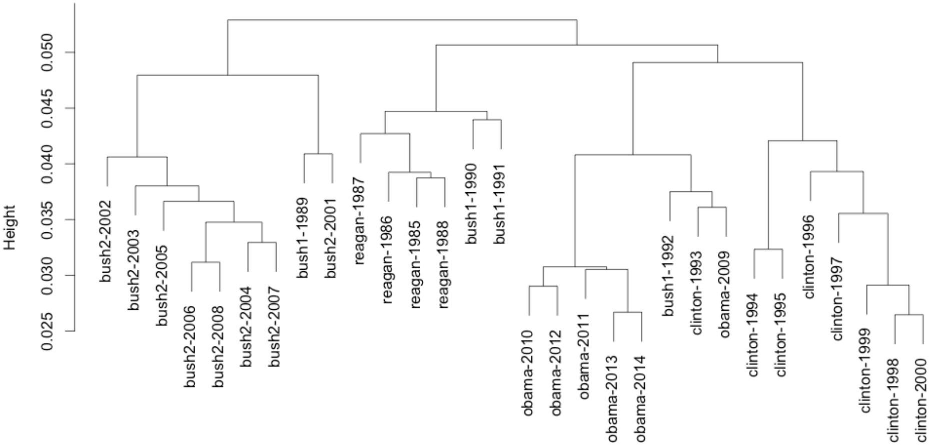
Other clustering methods: hierarchical clustering

- *agglomerative*: works from the bottom up to create clusters
- like *k*-means, usually involves *projection*: reducing the features through either selection or projection to a lower-dimensional representation
  1. SVD methods, such PCA on a normalised feature matrix
  2. usually simple threshold-based truncation is used (keep all but 100 highest frequency or tf-idf terms)
- Can be done at the document level, but also at the feature level (creating clusters of features)

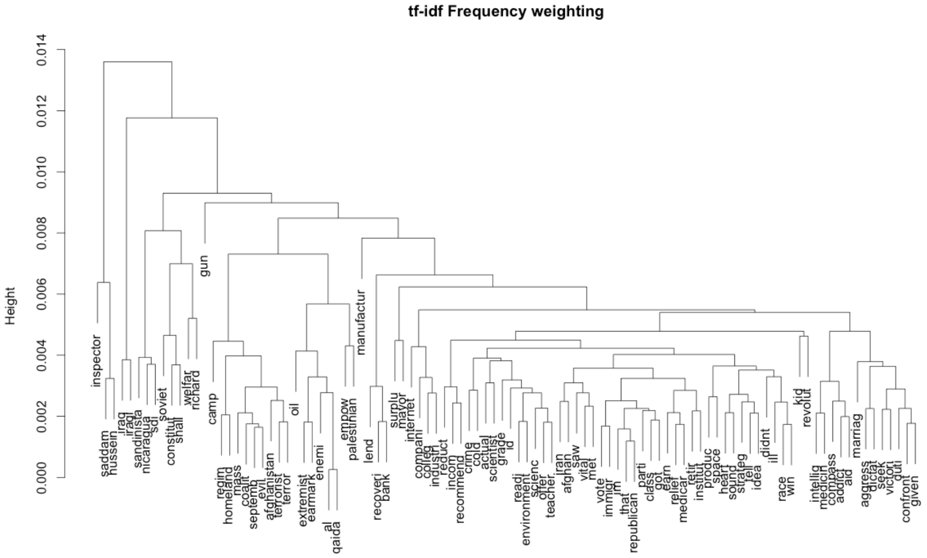
hierarchical clustering algorithm

1. start by considering each item as its own cluster, for n clusters
2. calculate the $N(N-1)/2$ pairwise distances between each of the n clusters, store in a matrix D_0
3. find smallest (off-diagonal) distance in D_0, and merge the items corresponding to the i, j indexes in D_0 into a new "cluster"
4. recalculate distance matrix D_1 with new cluster(s). Options for determining the location of a cluster include:
   - centroids (mean)
   - most dissimilar objects
   - Ward's measure(s) based on minimising variance
5. repeat 3–4 until a stopping condition is reached
   - i.e. all items have been merged into a single cluster
6. to plot the *dendrograms*, need decisions on ordering, since there are $2^{(N-1)}$ possible orderings

Dendrogram: Presidential State of the Union addresses



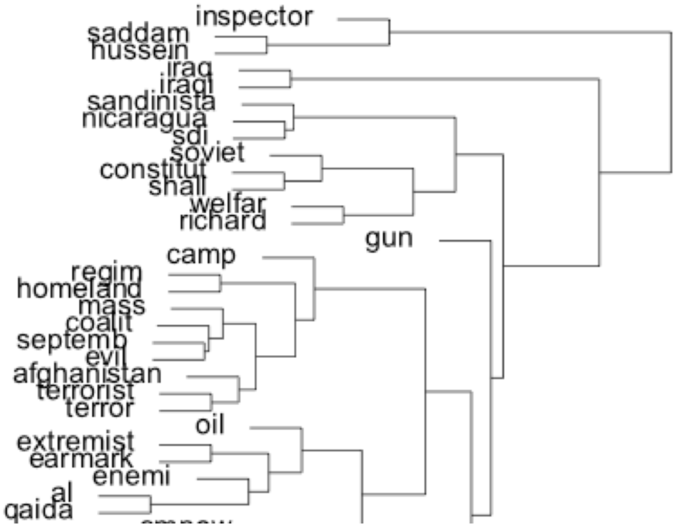
Dendrogram: Presidential State of the Union addresses



pros and cons of hierarchical clustering

- advantages
  - deterministic, unlike *k*-means
  - no need to decide on *k* in advance (although can specify as a stopping condition)
  - allows hierarchical relations to be examined (usually through *dendrograms*)
- disadvantages
  - more complex to compute: quadratic in complexity: $O(n^2)$ – whereas *k*-means has complexity that is $O(n)$
  - the decision about where to create branches and in what order can be somewhat arbitrary, determined by method of declaring the "distance" to already formed clusters
  - for words, tends to identify collocations as base-level clusters (e.g. "saddam" and "hussein")

Dendrogram: Presidential State of the Union addresses



inspector
saddam
hussein
iraq
iraqi
sandinista
nicaragua
sdi
soviet
constitut
shall
welfar
richard
gun
camp
regim
homeland
mass
coalit
septemb
evil
afghanistan
terrorist
terror
oil
extremist
earmark
enemi
al
qaida