Week 7: Text Similarity and Clustering

LSE MY459: Quantitative Text Analysis
https://lse-my459.github.io/

Ryan Hübert

Slides last updated: 3 March 2025

Outline for today

- String similarity
- Vector space model of language
- Distance metrics
- Similarity metrics
- Clustering methods

String similarity

Edit distance

Edit distance: number of operations needed to transform one string into another

- ➜ Useful for measuring **text reuse** (e.g., **plagiarism detection**)
- ➜ Can generalise to vectors (e.g., documents)

Most common edit distance metric is **Levenshtein distance**

Example: Levenshtein distance between `kitten` and `sitting` is 3

- ➜ `kitten` → `sitten` (substitution of “s” for “k”)
- ➜ `sitten` → `sittin` (substitution of “i” for “e”)
- ➜ `sittin` → `sitting` (insertion of “g” at the end).

You can calculate Levenshtein distance in R with `adist()` function, e.g. `adist("kitten", "sitting")` returns 3

Source for example: Wikipedia

Vector space model of language

Modeling language, continued

We've seen one way to model language: using probabilistic models

- ➜ E.g., language as draws from multinomial or Poisson distribution

Another common way to model language uses ideas from linear algebra

The basic idea: a document is mathematically represented as a **vector** in a J-dimensional **vector space**

- ➜ Each document vector is $\mathbf{W}_i$ from the dfm $\mathbf{W}$

This conception of language is not *new* for us, but we just haven't exploited ideas from linear algebra yet

Why this model?

Useful for contexts where **document similarity** is important

Why?

- ➜ Because vectors are *geometric* representations
- ➜ They can be characterised in terms of points in “space” where there are notions of distance and direction
- ➜ The “closeness” of documents has mathematical meaning

Keep in mind:

- ➜ Here, we’re focused on comparisons across documents
- ➜ Vector representations are also useful for characterising *features* within documents, e.g. word embeddings (coming weeks)

Vector representation for a simple corpus

Consider a simple example of three documents:

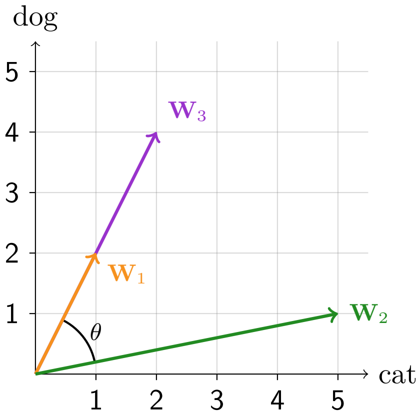
- Document 1: `cat dog dog`
- Document 2: `dog cat cat cat cat cat`
- Document 3: `dog dog cat dog cat dog`

The document feature matrix would look like:

```
             features
docs           cat dog
  Document 1    1   2
  Document 2    5   1
  Document 3    2   4
```

So: $\mathbf{W}_1 = (1, 2)$, $\mathbf{W}_2 = (5, 1)$ and $\mathbf{W}_3 = (2, 4)$

Vector representation for a simple corpus



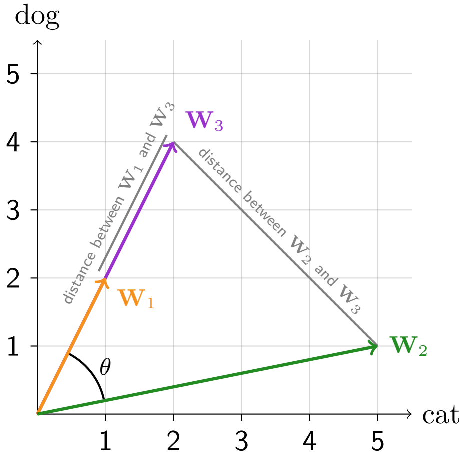
Core question: how do we *compare* these documents?

1. Distance: absolute separation between points
2. Similarity: directional closeness of vectors
3. Clustering: identify groups of similar documents

Distance metrics

Distance: absolute separation between points

Examples of distance between documents:


dog
cat
$\mathbf{W}_1$
$\mathbf{W}_2$
$\mathbf{W}_3$
θ
distance between $\mathbf{W}_1$ and $\mathbf{W}_3$
distance between $\mathbf{W}_2$ and $\mathbf{W}_3$
1
2
3
4
5
1
2
3
4
5

Euclidean distance

The previous slide illustrated the **Euclidean distance** between documents $\mathbf{W}_1$ and $\mathbf{W}_3$, and between $\mathbf{W}_2$ and $\mathbf{W}_3$

- ➜ Euclidean distance is based on the Pythagorean theorem

For two documents $\mathbf{W}_A$ and $\mathbf{W}_B$ with J features:

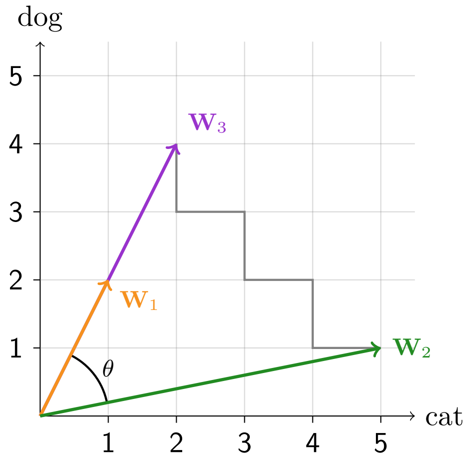
$$d_E(\mathbf{W}_A, \mathbf{W}_B) = \|\mathbf{W}_A - \mathbf{W}_B\| = \sqrt{\sum_j (W_{Aj} - W_{Bj})^2}$$

So, for $\mathbf{W}_2 = (5, 1)$ and $\mathbf{W}_3 = (2, 4)$:

$$\|\mathbf{W}_2 - \mathbf{W}_3\| = \sqrt{(5-2)^2 + (1-4)^2} \approx 4.24$$

Other distance metrics

The **Manhattan distance** (or **taxicab distance**) is calculated assuming you can only travel in right angles



Other distance metrics

It is calculated with this formula:

$$d_T(\mathbf{W}_A, \mathbf{W}_B) = \sum_j |W_{Aj} - W_{Bj}|$$

So, for $\mathbf{W}_2 = (5, 1)$ and $\mathbf{W}_3 = (2, 4)$:

$$d_T(\mathbf{W}_2, \mathbf{W}_3) = |5 - 2| + |1 - 4| = 6$$

Other distance metrics include:

- ➜ **Minkowski distance** (of which Euclidean is a special case)
- ➜ **Mahalanobis distance**

Key: use of distance metrics is up to the user!

Similarity metrics

Similarity: directional closeness

Document similarity: how directionally "close" are documents?

It's not obvious how to measure closeness, example from GRS:

- ➜ Are a positive review and a negative review of the same film "closer" than two positive reviews of different films?

Four desirable properties for document similarity measures:

1. Maximum similarity: comparing document to itself
2. Minimum similarity: two documents with no words in common
   - ➜ These are **orthogonal** documents
3. Similarity increases as more of same words used
4. Symmetry: similarity of $\mathbf{W}_A$ to $\mathbf{W}_B$ is the same as similarity of $\mathbf{W}_B$ to $\mathbf{W}_A$

The inner product

One straight-forward similarity metric: **inner product**

For two documents $\mathbf{W}_A$ and $\mathbf{W}_B$ with J features:

$$\mathbf{W}_A \cdot \mathbf{W}_B = W_{A1}W_{B1} + W_{A2}W_{B2} + \cdots + W_{AJ}W_{BJ}$$

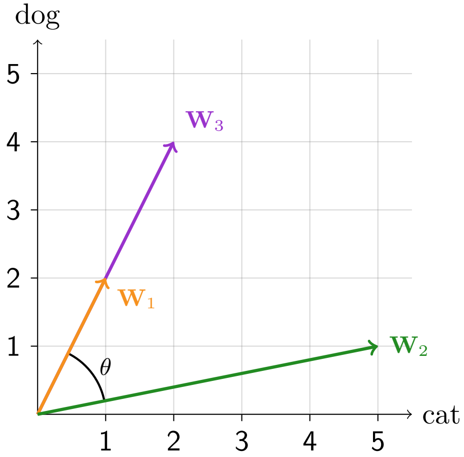
So, for $\mathbf{W}_2 = (5, 1)$ and $\mathbf{W}_3 = (2, 4)$:

$$\mathbf{W}_2 \cdot \mathbf{W}_3 = 5 \times 2 + 1 \times 4 = 14$$

Major downside: inner product is very sensitive to vector magnitude

- ➜ Vector magnitude is the length of vector—NOT same as document length
- ➜ Problem because similarity is about *directional* closeness

The inner product



$\mathbf{w}_1 \cdot \mathbf{w}_2 = 1 \times 5 + 2 \times 1 = 7$ $\quad$ $\mathbf{w}_2 \cdot \mathbf{w}_3 = 5 \times 2 + 1 \times 4 = 14$

Cosine similarity

To deal with this, let's *normalise* each document vector before calculating the inner product

- ➜ We will normalise by the vector magnitude
- ➜ A vector's magnitude is just the Euclidean distance between the vector and itself, notated as $\|\mathbf{x}\|$ for a vector $\mathbf{x}$

The magnitude of a vector $\mathbf{x}$ of length J can be calculated

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_J^2}$$

A document $\mathbf{W}_A$ can be normalised by dividing by its magnitude:

$$\frac{\mathbf{W}_A}{\|\mathbf{W}_A\|} = \frac{\mathbf{W}_A}{\sqrt{\sum_j W_{Aj}^2}}$$

Cosine similarity

The **cosine similarity** between two documents $\mathbf{W}_A$ and $\mathbf{W}_B$ is just the inner product of the normalised document vectors:

$$\text{cosine similarity}(\mathbf{W}_A, \mathbf{W}_B) = \frac{\mathbf{W}_A}{\|\mathbf{W}_A\|} \cdot \frac{\mathbf{W}_B}{\|\mathbf{W}_B\|}$$

Why is it called “cosine similarity”?

$$\text{cosine similarity}(\mathbf{W}_A, \mathbf{W}_B) = \frac{\mathbf{W}_A}{\|\mathbf{W}_A\|} \cdot \frac{\mathbf{W}_B}{\|\mathbf{W}_B\|} = \cos\theta$$

Cosine similarity is based on the size of the angle between vectors—their *directional* closeness!

Metric ranges from from 0 to 1, where 0 is least similar (“orthogonal”, 90° angles) and 1 is most similar

Cosine similarity

So, for $\mathbf{W}_2 = (5, 1)$ and $\mathbf{W}_3 = (2, 4)$:

$$\mathbf{W}_2 \cdot \mathbf{W}_3 = \frac{5 \times 2 + 1 \times 4}{\sqrt{(5^2 + 1^2)\,(2^2 + 4^2)}} \approx 0.614$$

And if we want to know the angle between these documents

$$\cos\theta = 0.614 \iff \theta = \arccos(0.614) \approx 0.9098 \text{ radians}$$

(This is approximately 52°)

There are other similarity metrics, but we won't go into detail

Clustering methods

The idea of "clusters" in the context of QTA

A **cluster** is a group of documents that are very similar to each other, but very different from documents outside the cluster

Clustering is "unsupervised classification", so clusters do not relate features to pre-existing classes or latent traits, but rather estimate membership of *pre-determined number of groups* (yikes!)

Groups formed by clusters are given labels through *post-estimation interpretation of their elements* (yikes!)

These methods are typically used when we do not and never will know the "true" class labels

- ➜ Clustering methods are not specific to QTA
- ➜ You can cluster words or other features (not just documents)

k-means clustering

Very common method for clustering: **k-means clustering**

The basic idea:

- ➜ Each document is assigned to one of K clusters
- ➜ Assignment of documents to clusters occurs in a way that minimises within-cluster difference and maximises between-cluster differences
- ➜ Uses random starting positions and iterates until stable
- ➜ Treats feature values as coordinates in a multidimensional space

Unfortunate conventions around notation: k versus K (be careful!)

k-means clustering

Advantages

- simplicity
- highly flexible
- efficient

Disadvantages

- no fixed rules for determining K
- uses an element of randomness for starting values, so can get different clusters each time you run the algorithm!

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, but others can be used
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

Calculating centroids

Assume DFM $\mathbf{W}$ with N documents and J features, to be clustered into K clusters

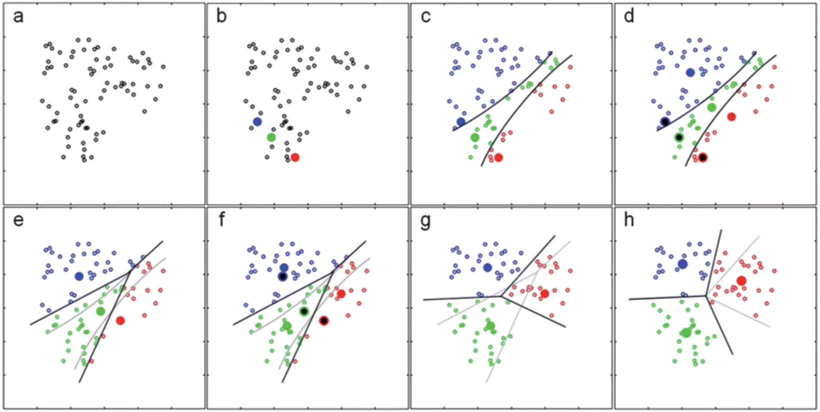
- $\pi_{ik} \in \{0, 1\}$ indicates whether document i is assigned to cluster k

The centroid of a cluster k is calculated as

$$\boldsymbol{\mu}_k = \frac{\overbrace{\sum_N \pi_{ik} \mathbf{W}_i}^{\text{J total word counts across docs in cluster}}}{\underbrace{\sum_N \pi_{ik}}_{\text{docs in cluster}}}$$

Calculation yields a J-length vector $\boldsymbol{\mu}_k$ of “mean” word counts for each feature j (taking mean across documents in cluster)

k-means clustering illustrated

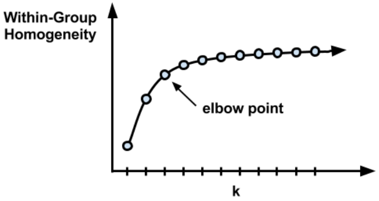


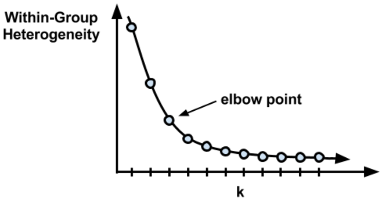
Choosing the appropriate number of clusters

Major point of analyst discretion: how many 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 documents to be clustered
  - ➜ usually too big: setting K too 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

Choosing the appropriate number of clusters

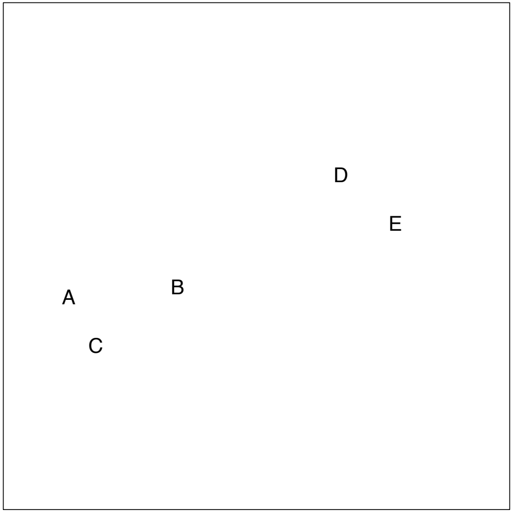




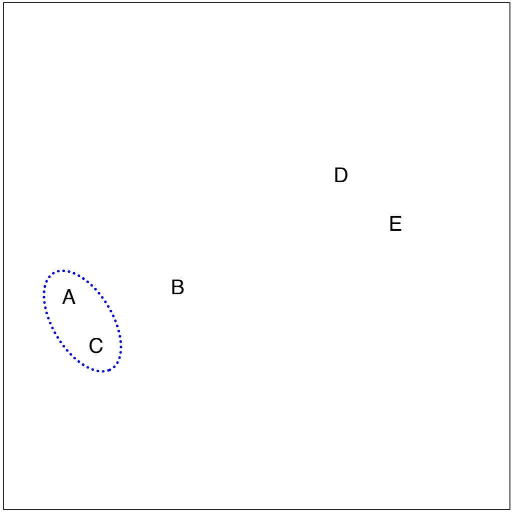
Hierarchical clustering

- k-means clustering requires us to pre-specify the number of clusters K
- Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K
- There are a lot of methods for hierarchical clustering; we'll focus on **bottom-up** or **agglomerative** clustering
  - This is the most common type of hierarchical clustering
  - A **dendrogram** is built starting from the leaves and combining clusters up to the trunk
- Still need to choose a number of clusters at some point (eek), but here, it's post estimation

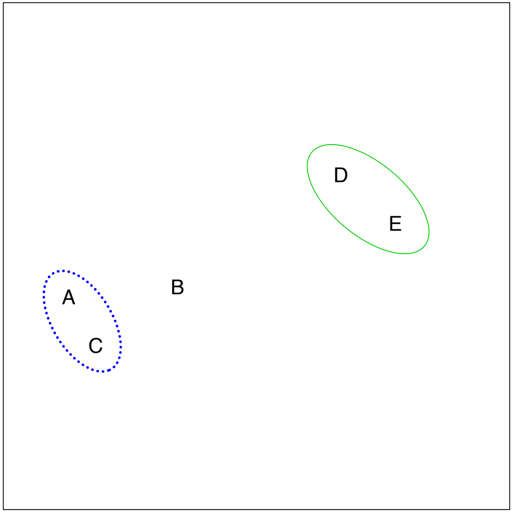
Hierarchical clustering: the idea



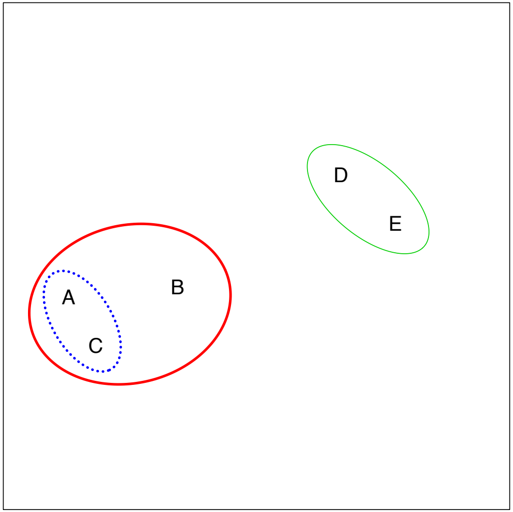
Hierarchical clustering: the idea



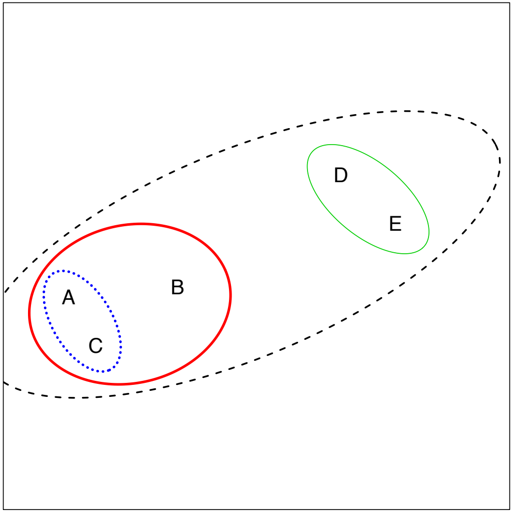
Hierarchical clustering: the idea



Hierarchical clustering: the idea



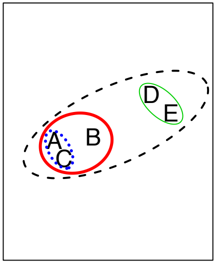
Hierarchical clustering: the idea

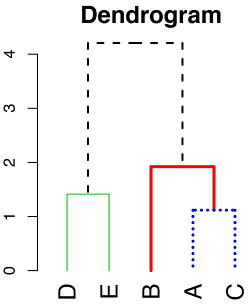


Hierarchical clustering algorithm

The approach in words:

1. Start with each point in its own cluster.
2. Identify the closest two clusters and merge them.
3. Repeat.
4. Ends when all points are in a single cluster.





Hierarchical clustering algorithm

1. Start with each document as its own "cluster" (N clusters)
2. Calculate 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 these two documents 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 and 4 until all items are in a single cluster

Hierarchical clustering algorithm

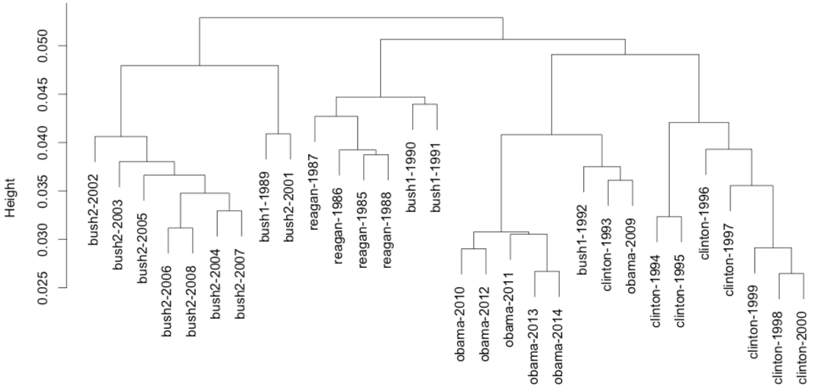
After you have finished this, you can create a **dendrogram**

This has a tree-like structure

You can “cut” it where you want—i.e., how many clusters do you want to consider

There isn’t a unique ordering of the leaves of a dendrogram, so making it look nice and interpretable can be tricky

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
  - 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

Learning from, and interpreting, clusters

Point of clustering: to *discover* new ways to classify a corpus of documents

This means:

1. Clusters are *corpus-dependent* — interpretation of clusters depends on document (and feature) selection
2. The way you *label* clusters is crucial, and potentially misleading

Labeling clusters

In my view, one of the most controversial parts of clustering (and topic models, next week) is **labeling**

Analyst takes a statistically-created contraption and attempts to make it meaningful with concise labels

This is necessary: the consumer/reader needs an interpretation!

But it can also be sketchy — people disagree on how to interpret texts!

Unfortunately: that's life. But we can try to be more principled...

Guiding principles adapted from GRS

1. Choose number of clusters *carefully*, *thoughtfully* and *transparently* as this will affect how you interpret the resulting clusters
2. Carefully read a random selection of documents in each cluster you estimate (between 10 and 30); take notes and synthesise notes into a corresponding label that matches content of documents
3. Have multiple people provide labels and compare; run experiments to see if subjects broadly agree with your interpretations
4. Provide list of most "distinctive" words of each cluster (if possible)
5. Check if your cluster labels/interpretations hold up after using different clustering approaches
6. Always provide all the details of your process in **replication materials**