Digital Humanities: Text-as-Data

Week 5 – From Words to Numbers: Vectorization and Clustering

Steven Denney
Leiden University

BA3 Korean Studies
November 14, 2025

WHAT IS CLUSTERING?

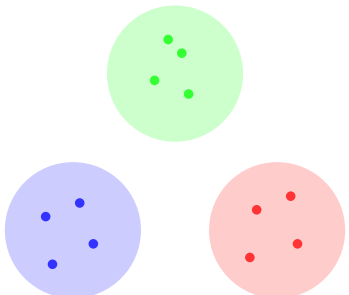
Thinking About Clusters Conceptually

- Clustering is about **grouping similar things together**.
- Each group — a **cluster** — contains items that are **more alike within** than they are **across** clusters.
- There are no "true labels" — we discover structure, not confirm it.
- In text-as-data:
  - Similar language or meaning $\Rightarrow$ same cluster.
  - Different vocabulary or topic $\Rightarrow$ different clusters.

Clustering = finding structure where none is pre-defined.

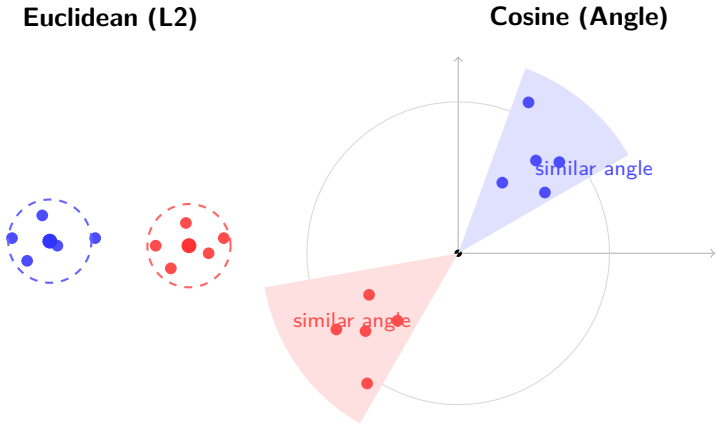
Cohesion and Separation

- Good clusters have two properties:
  1. **Cohesion** – items within a cluster are close together.
  2. **Separation** – clusters are far apart from one another.
- Distance or similarity measures provide the geometry that makes this possible.



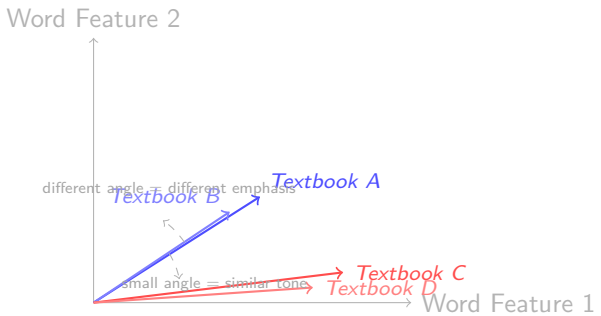
Distance Matters: Euclidean vs Cosine

- **Euclidean**: clusters are "near" a centroid in straight-line distance (circles/balls).
- **Cosine**: clusters share *direction* (angle) regardless of length (angular wedges).



Euclidean cares about absolute position around centroids; Cosine cares about angle (direction), ignoring length (good for text!).

Textbooks in Vector Space



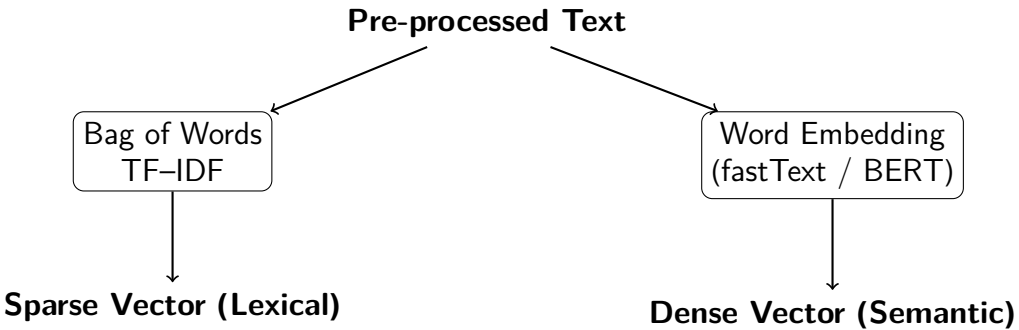
In our 51 textbooks, cosine similarity groups books that "speak" in similar proportions about nationalism, modernization, and identity.

FROM WORDS TO NUMBERS

Why We Need Numbers

- Computers cannot “read” language — they operate on numbers.
- Every text analysis method transforms text into a **numeric representation**.
- Once words are numbers, we can:
  - measure similarity (distance between documents),
  - discover structure (clustering),
  - and reduce complexity (topic modeling, embeddings).
- Today: two ways to vectorize text and two ways to cluster it.

Two Routes from Text to Vectors



Two paths for converting text into numerical representations for analysis.

Route 1: Bag-of-Words and TF–IDF

- **Bag-of-Words (BoW)**: represents each document by word counts.
- **TF–IDF** weights those counts:

$$\text{TF–IDF}(w, d) = \text{TF}(w, d) \times \log \frac{N}{n_w}$$

  - TF = frequency of word w in document d
  - IDF = rarity of w across all N documents
- Produces a high-dimensional, sparse vector — one value per term.
- Captures **lexical similarity**: shared vocabulary.

From TF–IDF to Distance

- Each document = a vector of TF–IDF weights.
- Compute pairwise similarity:

$$\text{cosine}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

- The **Distance widget** in Orange builds a matrix of all pairwise distances.
- Required for **Hierarchical Clustering**; optional for K-Means.

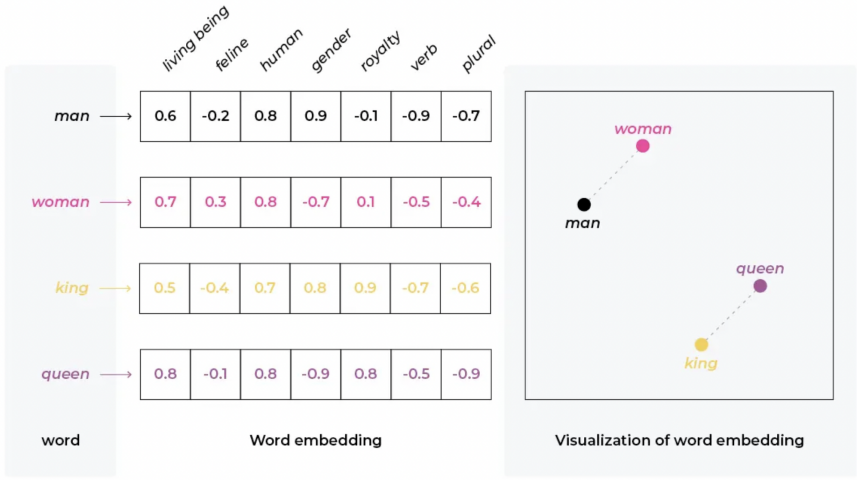
Route 2: Word Embeddings

- Learned from large corpora using neural networks (e.g., fastText, BERT).
- Each word or document = dense vector (300–768 dims).
- Similar meanings $\rightarrow$ vectors close together.
- Encodes **semantic similarity**, not just co-occurrence.

Vector Geometry of Meaning

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

- Arithmetic on embeddings reflects relationships learned from context.
- “Meaning” becomes a location in high-dimensional space.
- Enables geometric operations: distance, clustering, visualization.


living being
feline
human
gender
royalty
verb
plural
man
0.6
-0.2
0.8
0.9
-0.1
-0.9
-0.7
woman
0.7
0.3
0.8
-0.7
0.1
-0.5
-0.4
king
0.5
-0.4
0.7
0.8
0.9
-0.7
-0.6
queen
0.8
-0.1
0.8
-0.9
0.8
-0.5
-0.9
word
Word embedding
woman
man
queen
king
Visualization of word embedding

TF–IDF vs. Embeddings

	TF–IDF	**Embeddings**
Representation	Sparse term frequencies	Dense semantic vectors
Captures	Lexical overlap	Contextual meaning
Dimensionality	Very high (1 per word)	Moderate (hundreds)
Distance metric	Cosine / Euclidean	Cosine / Euclidean
Interpretation	Shared vocabulary	Shared meaning

WHAT WE DO WITH THESE VECTORS

Two Main Clustering Algorithms

Hierarchical Clustering

- Uses **distance matrix**.
- Merges closest pairs step-by-step.
- Output: dendrogram (tree of similarity).
- Linkage choices: average, Ward, etc.

K-Means Clustering

- Works directly on vectors.
- Groups around centroids to minimize variance:

$$\min \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

- Output: discrete cluster labels.

Hierarchical Clustering in Orange

1. **[Preprocess] → BoW → Distance → Hierarchical Clustering**
2. **[Preprocess] → Document Embedding → Distance → Hierarchical Clustering**

- The Distance widget computes pairwise cosine/Euclidean values.
- The Hierarchical widget builds and visualizes a dendrogram.
- Selecting a branch outputs that subset of documents.

K-Means in Orange

1. **[Preprocess] → BoW → K-Means**
2. **[Preprocess] → Document Embedding → K-Means**

- Algorithm:
  1. Choose k clusters.
  2. Randomly initialize centroids.
  3. Assign each point to nearest centroid.
  4. Recompute centroids until stable.
- Evaluate with **Silhouette Plot** and interpret with **Word Cloud / Extract Keywords**.

Putting It Together

1. Preprocess text $\rightarrow$ clean tokens.
2. Choose vectorization:
   - Path A: BoW $\rightarrow$ TF–IDF (lexical)
   - Path B: Document Embedding (semantic)
3. Cluster:
   - Hierarchical (needs Distance matrix)
   - or K-Means (directly on vectors)
4. Interpret clusters $\rightarrow$ Word Cloud / Keywords.

NEXT WEEK: TOPIC MODELING AS DIMENSION REDUCTION