BA2: Digital Korea

Week 7: Clustering

From Description to Discovery

Steven Denney

Korean Studies
Leiden University

March 16, 2026

Today's Agenda

1. From description to discovery
2. What is clustering?
3. Distance and similarity (review)
4. Hierarchical clustering
5. K-means clustering
6. Comparing the two approaches
7. Break
8. Demo: Clustering Korean corpora in Orange

From Description to Discovery

Where We Are

The journey so far

Weeks 1–5: Description

- Preprocessing text
- Bag of Words / TF-IDF
- Word clouds, bar charts
- Concordance, subsetting

You learned to **look at** your data.

Weeks 7–10: Discovery

- **Clustering** (today)
- Word embeddings
- Sentiment analysis
- Topic modeling

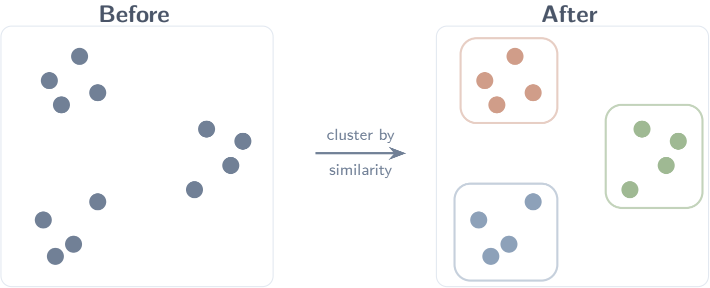
Now you learn to **find structure** in your data.

The shift

Description asks: "What words appear?" Discovery asks: "What **patterns** emerge?"

You Already Know How to Cluster

You group similar things together every day — without thinking about it.



- Organizing a bookshelf: cookbooks here, novels there, textbooks over there
- Sorting photos on your phone: trips, family, food
- A music app grouping songs into playlists from your listening habits

Why Cluster Text?

What clustering can find and show to us in a corpus

Now imagine you have **hundreds of documents** — too many to read and sort by hand.

Clustering algorithms do the sorting for you, grouping documents by shared vocabulary, themes, or style.

Korean Studies examples

- Do presidential speeches cluster by **president** or by **policy theme**?
- Do history textbooks cluster by **era** (Colonial, Authoritarian, Democratic)?
- Do newspaper editorials cluster by **political leaning**?

Supervised vs. Unsupervised Learning

Supervised

- You provide **labels** (e.g., positive/negative)
- The computer learns to predict them
- Example: sentiment classification

Unsupervised

- **No labels** — the computer finds structure on its own
- You discover groups you didn't define in advance
- Example: **clustering**

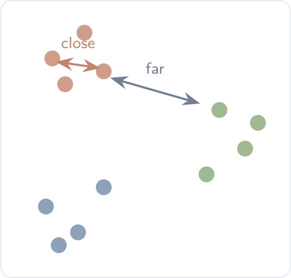
Key point

Clustering is **unsupervised**. There are no "correct answers" here — you discover structure, then *interpret* whether it makes sense.

What Is Clustering?

What Makes a Good Cluster?

A **cluster** is a group of items that are more alike *within* than they are *across* groups. Good clusters have two properties:



1. **Cohesion** — items within a cluster are close together

2. **Separation** — clusters are far apart from one another

In text-as-data: documents with similar vocabulary end up in the same cluster. Documents with different vocabulary end up in different clusters.

Distance and Similarity

From Documents to Distances

Before we can cluster, we need to measure how similar two documents are.

Preprocessed Text → Bag of Words (TF-IDF) → Distances → Clustering

- Each document becomes a **vector** of numbers (you learned this in Weeks 3–4)
- We compute **pairwise distances** between all documents
- The clustering algorithm uses these distances to form groups

Euclidean vs. Cosine Distance

Euclidean distance
Straight-line distance between two points.

$$d(A, B) = \sqrt{\sum_i (a_i - b_i)^2}$$

- Sensitive to **magnitude**
- A long document and a short document on the same topic can seem far apart

Cosine distance
Measures the *angle* between two vectors, ignoring length.

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

- Ignores **document length**
- Focuses on the *direction* — the mix of words, not how many

Breaking Down Euclidean Distance

For your reference — a closer look at the formula

$$d(A, B) = \sqrt{\sum_i (a_i - b_i)^2}$$

Symbol	Meaning
a_i, b_i	TF-IDF weight of word i in Doc A and Doc B
$(a_i - b_i)^2$	Square the difference for each word
$\sum$	Add up the squared differences across every word

Read it as: for each word, measure how different the two documents are, square it, add them all up, take the square root.

Problem for text: a long document and a short document on the same topic will seem far apart because magnitude differs. This is why we use cosine instead.

Breaking Down Cosine Similarity

For your reference — a closer look at the formula

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

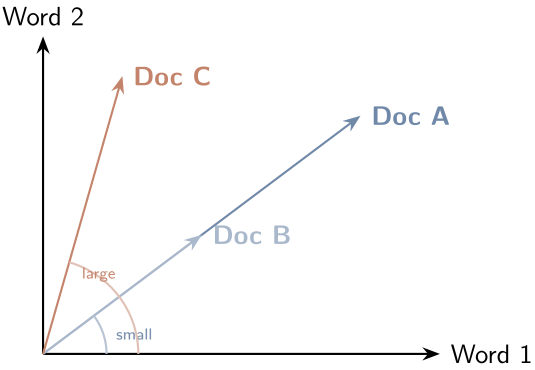
Part	What it does
Top ($A \cdot B$)	Multiply matching word weights and add them up
Bottom ($\|A\| \, \|B\|$)	Divides by document lengths, canceling out size

Result: $1 =$ identical word mix. $0 =$ completely different.

Cosine distance $= 1 -$ similarity. This is what Orange computes.

Cosine Similarity: Intuition

Angle between vectors, not distance between points



What you are seeing:

Each axis is a word. Each arrow is a document's TF-IDF vector.

- **A & B**: same direction, different lengths. **Small angle** = similar.
- **A & C**: different directions. **Large angle** = dissimilar.

Key: Cosine ignores arrow length (document size) and only measures the angle.

Hierarchical Clustering

What Is Hierarchical Clustering?

Plain explanation

Groups documents based on shared vocabulary patterns. Think of it as building a "family tree" of documents by similarity.

How it works (agglomerative / bottom-up):

1. Start with every document as its own cluster
2. Find the two **most similar** clusters
3. **Merge** them into one
4. Repeat until everything is in a single cluster

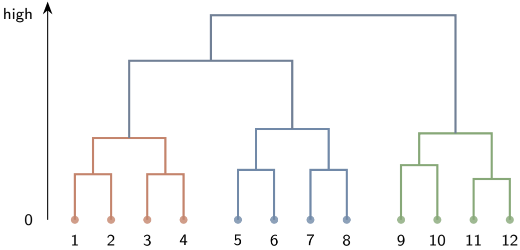
- Documents within the same cluster → similar content
- Documents in different clusters → different topics or styles

Building a Dendrogram

The algorithm finds the two closest points, merges them, and repeats

The data (same 12 points)

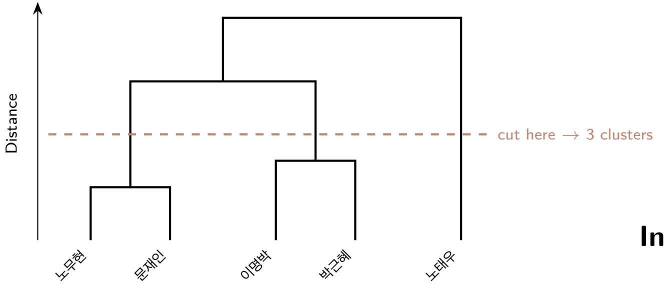
The dendrogram (merge history)



1. Each point starts as its own cluster
2. Find the closest pair, merge them
3. Repeat until one cluster remains

Low merges = similar. **High** merges = different. The tree records the entire history.

Reading a Dendrogram



How to read it:

- **Lower** merges = more similar
- **Higher** merges = more different
- "Cut" horizontally to choose the number of clusters

In this example:

- Progressive presidents cluster together
- Conservative presidents cluster together
- Cutting at the dashed line gives 3 groups

Linkage: How to Measure Distance Between Groups

Once two points merge into a group, how do we measure the distance from that group to other groups? This is the **linkage** method.

Ward's linkage (what we will use)

Merges the two groups that increase total variance the *least*. Tends to produce compact, evenly-sized clusters. A good default for exploration.

Other options exist — **average** (mean distance between all pairs) and **complete** (distance between farthest members) — but Ward's is the most common starting point and what Orange uses by default.

Hierarchical Clustering: Strengths and Limitations

Strengths

- No need to choose k in advance
- The dendrogram shows the **full structure** — you decide where to cut
- Good for exploring relationships at multiple levels
- Intuitive visual output

Limitations

- Slow for large corpora (computes all pairwise distances)
- Merges are **final** — once joined, clusters cannot be split
- Requires a distance matrix as input
- Sensitive to linkage choice

Best for

Small to medium corpora where you want to **explore** the structure and see relationships at different levels of granularity.

K-Means Clustering

What Is K-Means?

Plain explanation

Partition your documents into exactly k groups by finding cluster centers (centroids) and assigning each document to its nearest center.

The algorithm:

1. Choose k (the number of clusters you want)
2. Place k centroids randomly in the vector space
3. **Assign** each document to the nearest centroid
4. **Recompute** each centroid as the mean of its assigned documents
5. Repeat steps 3–4 until the centroids stop moving (**convergence**)

The goal: minimize the total distance between each document and its cluster's center.

K-Means: The Objective

For your reference — what the algorithm is trying to do

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

Symbol	Meaning
k	Number of clusters
x_j	A document's TF-IDF vector
μ_i	The centroid (average) of cluster i
$\|x_j - \mu_i\|^2$	Squared distance from document to its cluster center

In plain English: make each document as close as possible to the center of its own cluster.

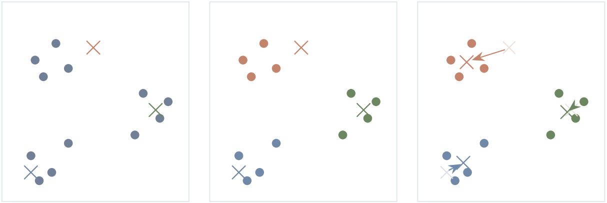
K-Means: Step by Step

Same 12 points — now partitioned by k-means with $k = 3$

1. Place centroids
Pick $k = 3$ random starting points

2. Assign
Color by nearest center

3. Recompute
Move centers to group means



Repeat assign → recompute until centers stop moving. Here the groups match immediately — with messier data, it takes a few rounds.

Choosing k: How Many Clusters?

K-means requires you to specify the number of clusters in advance. How do you choose?

Approaches

1. **Domain knowledge** — you have a hypothesis (e.g., 5 presidents → try $k = 5$)
2. **Try multiple values** and compare
3. **Silhouette scoring** — let the data guide you

Silhouette score

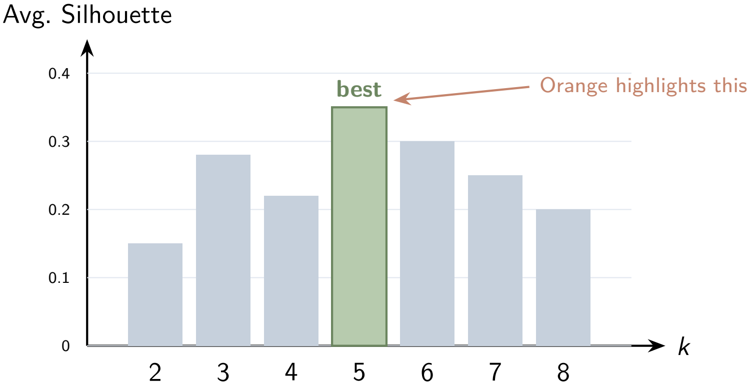
- Ranges from -1 to $+1$
- **High** (≈ 1): point fits well in its cluster
- **Low** (≈ 0): point is between clusters
- **Negative**: probably in the wrong cluster

Silhouette scoring helps, but the "right" k also depends on your **research question**. A statistically optimal k is not always the most interpretable.

Silhouette Scores: Comparing *k* Values

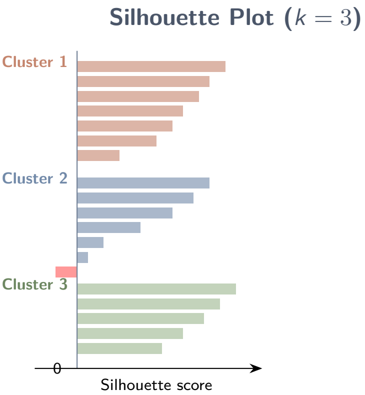
This is what Orange shows you in the K-Means widget

Orange runs k-means for every *k* in the range and shows the average silhouette score for each. Higher is better.



Silhouette Scores: Per-Document View

How well does each document fit its assigned cluster?



Reading the plot:

Each horizontal bar is one document, grouped by cluster and sorted by score.

- **Long bars** (near 1): strong fit
- **Short bars** (near 0): borderline
- **Negative bars**: probably in the wrong cluster

Cluster 3 has uniformly long bars — a strong cluster. **Cluster 2** has one negative bar — that document might belong elsewhere.

Make this in Orange with the **Silhouette Plot** widget.

K-Means: Strengths and Limitations

Strengths

- Fast — scales well to large corpora
- Simple and intuitive
- Works directly on vectors (no distance matrix needed)
- Easy to interpret: each document gets one label

Limitations

- Must choose k in advance
- Results depend on **random initialization** (run it multiple times)
- Assumes roughly **spherical** clusters of similar size
- Cannot capture nested or overlapping structure

Best for

Larger corpora where you want a **clean partition** into groups — especially when you have a hypothesis about the number of clusters.

Comparing the Two Approaches

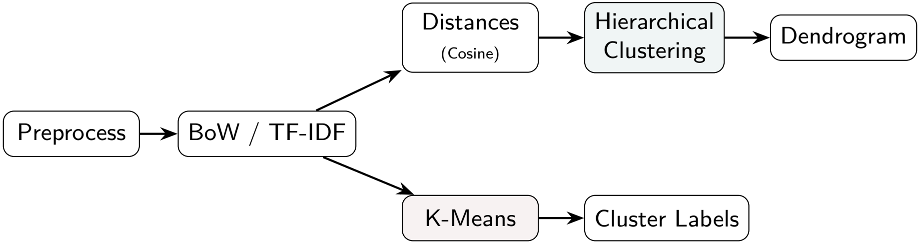
Hierarchical vs. K-Means

	Hierarchical	**K-Means**
Input	Distance matrix	Vectors (or distance matrix)
Output	Dendrogram (tree)	Flat cluster labels
Choose k?	No — cut the dendrogram after	Yes — must specify in advance
Speed	Slow (all pairwise distances)	Fast (iterative)
Deterministic?	Yes	No (random initialization)
Best for	Exploration, small corpora	Partitioning, larger corpora

Use **hierarchical clustering** to explore structure. Use **k-means** for clean partitions — or use both and compare.

The Full Pipeline

From text to clusters



Considerations for Text

Pitfalls and Practical Tips

1. **Garbage in, garbage out** — clustering is only as good as your preprocessing and distance metric. Use TF-IDF weighting and cosine distance for text (as we have been doing).

2. **Clusters are not "truth"** — they reflect patterns in the data, not objective categories. Always ask: does this grouping make sense for my research question?

3. **Try different settings** — vary the number of clusters, linkage method, or distance metric. If the results change dramatically, the structure may be weak.

From Description to Discovery

The interpretive shift

You are moving from counting words to identifying patterns that reflect underlying thematic or temporal structure in historical texts.

- **Descriptive:** Which words are frequent or distinctive?
- **Analytical:** Which documents are similar or different?
- **Interpretive:** What do these groupings tell us about historical narratives?

Before we try this ourselves, let's walk through the key settings we will use in Orange.

Setting Up for the Demo

Bag of Words: Our Settings

Turning text into numbers the clustering algorithm can use

We will use two settings in the Bag of Words widget:

Setting	**Value**	**Why**
Term Frequency	Count	Raw word counts — our starting point
Document Frequency	IDF	Down-weights words that appear everywhere (e.g., 역사); highlights what's distinctive
Regularization	None	Not needed — cosine distance already accounts for document length

Distances and Hierarchical Clustering

Settings for the demo

Distances widget

- Compare: **Rows** (documents)
- Metric: **Cosine**

Cosine distance measures the *angle* between two TF-IDF vectors. Documents with similar word profiles end up close together, regardless of length.

This is the natural choice for text data, where documents vary widely in length.

Hierarchical Clustering widget

- Linkage: **Ward**
- Annotations: **nikh_period** (or another metadata column)
- Selection: **Top N: 3** (cuts into 3 clusters)

Ward produces compact, evenly-sized clusters. You can also select clusters manually by clicking on the dendrogram branches.

K-Means Settings

Settings for the demo

Key settings

- Clusters: **From 2 to 8**
- Preprocessing: **Normalize columns** checked
- Initialization: **KMeans++**
- Re-runs: **10**

Orange tries every *k* in the range and scores each. Leave other settings at defaults.

Reading the silhouette scores

- Orange shows a **silhouette score** for each *k*
- Higher score = better-defined clusters
- The highlighted row is Orange's best guess
- But always ask: does this *k* make sense for my data?

You can visualize results with a **Silhouette Plot** to see how well individual documents fit their assigned cluster.

Break

We will resume in 10 minutes.