BA2: Digital Korea

Week 11 Assessment Study Guide — Covering Weeks 7–10

Dr. Steven Denney | Korean Studies, Leiden University | Spring 2026

Assessment format. The Week 11 assessment has two parts:

1. **Online quiz** (∼20 min) — multiple-choice questions on concepts from Weeks 7–10, administered via Brightspace during class.
2. **Application exercise** (∼20 min) — apply *at least two* of the four second-half methods (clustering, word embeddings, sentiment, LDA) to a question I pose, using a corpus I provide on the day. Come prepared to use *any* of them; you choose which to combine once you see the task. Upload your `.ows` file, one or two screenshots, and a short `.md` write-up describing what you did and what you found to your GitHub repo.

This guide covers the concepts and workflows you need to review.

1 Clustering (Week 7)

1.1 What is unsupervised learning?

In **unsupervised learning**, the algorithm is given data without labels and asked to find structure on its own. No one tells it what the groups should be. You provide the texts; the algorithm groups them; then you interpret what the groups mean.

1.2 Documents as vectors

Clustering takes a **Document–Term Matrix** (Week 4) as input: each document is a row, each term is a column, each cell is a count or TF–IDF weight. Every document is therefore a *vector* in word space. We almost always cluster on TF–IDF vectors rather than raw counts so that distinctive words drive the grouping.

1.3 Distances

Before you can cluster, you need to measure how far apart any two documents are.

Metric	What it measures	When to use it
Cosine	Angle between two vectors	Default for text: length-insensitive
Euclidean	Straight-line distance	Rarely for text: penalises long docs
Manhattan	Sum of absolute differences	Occasionally, for robustness to outliers

> **Why cosine for text?** A 5,000-word speech and a 500-word speech that talk about the same things should be judged similar. Cosine ignores vector length and looks only at the direction, so it treats both as near-identical if they emphasise the same terms.

1.4 Hierarchical clustering

Hierarchical (agglomerative) clustering builds a **tree of clusters** from the bottom up:

1. Start with every document as its own cluster.
2. Find the two closest clusters and merge them.
3. Repeat until one cluster contains everything.

The result is a **dendrogram** — a branching diagram where vertical height is the distance at which clusters merged.

Linkage decides how "close" two *clusters* are (once they contain more than one document):

Linkage	Definition
Ward	Merge the pair that increases within-cluster variance least (our default)
Single	Distance between the two nearest points across clusters (prone to chaining)
Complete	Distance between the two farthest points (compact, tight clusters)
Average	Mean of all pairwise distances

You pick the number of clusters *after* fitting, by cutting the dendrogram at a chosen height.

1.5 K-means clustering

K-means asks you to commit to a number of clusters *before* fitting:

1. Pick k.
2. Place k **centroids** randomly in the space.
3. Assign each document to the nearest centroid.
4. Move each centroid to the mean of its assigned documents.
5. Repeat steps 3–4 until nothing changes.

Because the initial centroids are random, two runs on the same data can produce slightly different results.

> **Hierarchical vs. k-means.** Hierarchical is exploratory: fit once, inspect the tree, decide k. K-means is decisive: you commit to k first, then fit. Use hierarchical when you don't know k; use k-means when k is fixed or when the corpus is large enough that the hierarchical tree is unwieldy.

1.6 Choosing k

There is no universal rule. Useful signals:

- For **hierarchical**: cut the dendrogram where branches are long (big gaps between successive merges).
- For **k-means**: a **silhouette score** (higher is better, $[-1, 1]$) measures how well each document fits its cluster versus its second-best alternative.
- Always read the clusters. A numerically "good" k that produces un-interpretable clusters is not useful.

1.7 Interpreting clusters

A cluster label is just an integer. Your job is to read each cluster's documents and vocabulary and decide what the group is about.

- **Word Cloud per cluster** (subset the corpus, then visualise) — catches obvious themes.
- **Top TF–IDF terms per cluster** — shows what makes each cluster *distinctive*.
- **Box Plot of cluster vs. metadata** — does each cluster map onto an era, president, or genre?

Widgets to know: Distances, Hierarchical Clustering, k-Means, Silhouette Plot, Corpus Viewer, Box Plot.

2 Word Embeddings (Week 8)

2.1 From sparse to dense

A BoW vector has one dimension per vocabulary word. For a 10,000-word corpus, every document is a 10,000-dimensional vector, and almost every entry is zero. That is a **sparse** representation.

A **word embedding** is a **dense** vector — typically 100 to 1,000 dimensions, every entry non-zero. The dimensions do not correspond to individual words; they encode latent features learned from large amounts of text.

	BoW / TF–IDF	**Embedding**
Length	Vocabulary size (huge)	Fixed (e.g. 768)
Density	Mostly zeros	Mostly non-zero
Captures	Which words appear	Meaning and context
"뉴스" vs. "기사"	Different columns, unrelated	Similar vectors, related

2.2 The distributional hypothesis

The intuition behind embeddings: *you shall know a word by the company it keeps.* Words that appear in similar contexts across a large corpus end up with similar vectors. 경제 ("economy") and 경제적 ("economic") will be close because they show up alongside the same neighbours.

2.3 Nearest neighbours

Once words are vectors, you can find the most similar word to any query word by computing cosine similarity to every other word and ranking the results. This is what the **Nearest Neighbors** widget does.

2.4 Document embeddings

For document-level analysis we need one vector per document, not per word.

- **Averaging**: take the mean of the word vectors in the document.
- **Transformer models** (BERT, KLUE BERT for Korean): pass the document through a pre-trained neural network, which outputs a single vector that summarises it.

The **Document Embedding** widget in Orange uses a transformer; for Korean, the pre-trained **KLUE BERT** model.

2.5 Visualisation with t-SNE

Embeddings are too high-dimensional to plot directly. **t-SNE** projects them down to 2D while trying to preserve neighbour relationships. Two documents that were close in the original space stay close in the 2D plot. Clusters on a t-SNE plot often correspond to semantically related groups.

> **t-SNE distances are not exact.** The 2D positions preserve local neighbourhoods, not global distances. Points that look far apart may or may not actually be far apart in the original space. Use t-SNE for exploration, not for measurement.

2.6 When embeddings beat BoW

Embeddings shine when:

- You care about **meaning** (semantic search: find similar *content*, not just shared words).
- The corpus uses many **synonyms or near-synonyms** that BoW would treat as unrelated.
- You want to do **downstream tasks** (clustering, classification) on a richer representation than a DTM.

Widgets to know: Document Embedding, Nearest Neighbors, t-SNE, Scatter Plot, Corpus Viewer.

3 Sentiment Analysis (Week 9)

3.1 Dictionary-based sentiment

A **sentiment dictionary** is two lists: positive words and negative words. To score a document, count how many of its tokens appear on each list and combine them. Common scoring formulas:

- Simple: $\text{score} = (\text{pos count}) - (\text{neg count})$
- Normalised: $\text{score} = \dfrac{\text{pos count} - \text{neg count}}{\text{pos count} + \text{neg count}}$
- Or with word-level polarity weights (some dictionaries rate words -2 to $+2$).

3.2 The KNU dictionary

For Korean we use the **KNU Sentiment Dictionary** (Park et al. 2018, Kunsan National University):

- `positive.txt`: 4,868 positive-polarity Korean words.
- `negative.txt`: 9,824 negative-polarity Korean words.

You load these into the **Sentiment Analysis** widget's "Custom Dictionary" mode.

3.3 Preprocessing for sentiment

Unlike topic modeling, we *do* keep verbs and adjectives for sentiment:

POS tag	Category	Why keep it?
NNG	Common noun	Topic words (경제, 정부, …)
NNP	Proper noun	Named entities
VV	Verb	Feelings often sit in verbs (사랑하다, 미워하다)
VA	Adjective	Valence-bearing (좋다, 나쁘다, 기쁘다)

This is why there is a separate `sentiment_preprocessing_*.py` script on the Data & Scripts page.

3.4 Strengths and limitations

Strengths	Transparent (you can audit every word that contributed), no training data needed, fast, reproducible.
Limitations	Ignores context (sarcasm, negation, idioms). No domain adaptation. *안 좋다* ("not good") is counted as positive because 좋다 is on the positive list.

> **Be careful with political or domain-specific text.** A general-purpose dictionary will flag neutral policy terms and miss domain-specific polarity. Always read a random sample of high-positive and high-negative documents to sanity-check what the dictionary is picking up on.

3.5 Subsetting and comparison

Two useful companion widgets:

- **Select Rows**: filter the corpus before scoring. Our Week 9 demo uses this to isolate high-engagement tweets (`retweets` $>$ 1000).
- **Box Plot**: compare sentiment distributions across metadata groups (time periods, speakers, engagement levels).

Widgets to know: Sentiment Analysis, Select Rows, Box Plot, Corpus Viewer.

4 Topic Modeling (LDA, Week 10)

4.1 The idea

Latent Dirichlet Allocation (LDA) is a probabilistic model that assumes:

- Each **document is a mixture of topics**. A textbook might be 40% ancient history, 30% colonial resistance, 30% everything else.
- Each **topic is a mixture of words**. One topic might put high probability on {고구려, 백제, 신라, 삼국}; another on {일제, 독립, 저항}.

LDA discovers both mixtures simultaneously by scanning the co-occurrence patterns of words across documents.

4.2 The two matrices

After fitting, LDA gives you two probability tables:

Matrix	Shape	What each row/cell means
β (or ϕ)	$K \times V$	Row k = probability distribution over words for topic k
θ (or γ)	$D \times K$	Row d = probability distribution over topics for document d

Every row sums to 1. β answers "What is topic k about?"; θ answers "Which topics does document d use?"

4.3 Choosing k

The number of topics k is a **research choice**:

- Too small $\rightarrow$ unrelated themes get blurred together.
- Too large $\rightarrow$ single themes get split into near-duplicates.
- **Coherence scores** (e.g. c_v) help rule out obviously bad k values. Higher is usually better.
- Always read the topics. A numerically good k that produces unreadable topics is not useful.

4.4 Preprocessing for LDA

For topic modeling we keep **nouns only** (NNG, NNP). Unlike sentiment analysis, which keeps verbs and adjectives, LDA works better when the vocabulary is restricted to topic-bearing words.

> **Why nouns only?** Topics are about *what a text is about*, and in Korean that is carried by nouns. Verbs like 하다 ("do") and 되다 ("become") appear everywhere and blur topics rather than separate them.

We also apply a **document-frequency filter**: drop terms that appear in very few documents (likely OCR noise) or in almost every document (too common to separate topics). This replaces what TF–IDF's IDF step would do. LDA itself consumes *counts*, not TF–IDF weights.

4.5 LDAvis

LDAvis is an interactive map of a fitted LDA model:

- **Left panel**: each topic is a circle. Size = how prevalent the topic is. Distance between circles = how dissimilar the topics are.
- **Right panel**: when you click a topic, a bar chart shows its top words. Red = how often the word appears in that topic. Grey = how often it appears in the whole corpus.
- λ **slider**: blends raw frequency ($\lambda = 1$) with distinctiveness ($\lambda = 0$). Start at $\lambda \approx 0.3$ to see what makes a topic specific.

4.6 LDA vs. clustering

Both are unsupervised and both produce groups, but they differ:

	Clustering	LDA
Assignment	Each doc in *one* cluster	Each doc is a *mixture* of topics
Word-level output	Top words per cluster (post-hoc)	β matrix: probabilities built in
Interpretability	Read docs + top words	Read top words; use LDAvis
When to use	Discrete groupings wanted	Themes that can overlap

> **LDA is a reading aid, not an oracle.** The model finds co-occurring words; *you* read them and decide what they mean. Every topic label you write is an interpretation.

Widgets to know: Topic Modelling, LDAvis, Corpus Viewer, Data Table (for θ), Box Plot (topic share by metadata).

5 The Four Methods at a Glance

Method	Input	Output	When to use it
Clustering	DTM with TF–IDF	One cluster label per document	You want discrete groups and no prior labels
Word embeddings	Corpus (raw or lightly preprocessed)	Dense vector per word *and*/or per document	You care about meaning, semantic search, or similarity
Sentiment (dict.)	Corpus + positive/negative lists	A sentiment score per document	You want to measure tone or valence transparently
LDA	DTM with counts (nouns only)	β (topics $\times$ words), θ (docs $\times$ topics)	You want themes that documents can *mix*

> **Two-method combinations that work well.** A few patterns you might combine for the application exercise:
>
> - **Clustering + sentiment**: do different clusters have different sentiment distributions?
> - **LDA + sentiment**: is any topic consistently more positive or negative than others?
> - **LDA + clustering**: do documents with similar topic mixtures cluster together?
> - **Embeddings + clustering**: run hierarchical clustering on document embeddings instead of TF–IDF.

6 Self-Check Questions

Work through these. If you can answer them confidently, you are well prepared.

6.1 Clustering

1. What is unsupervised learning, and how does it differ from supervised learning?
2. Why do we usually cluster on TF–IDF vectors rather than raw counts?
3. Why is cosine distance preferred over Euclidean for text?
4. What does cutting a dendrogram at different heights give you?

5. What is the Ward linkage rule, and when would you prefer it over single linkage?
6. Why can two runs of k-means on the same data give different clusters?
7. What does a silhouette score of 0.75 vs. 0.15 tell you?

6.2 Word Embeddings

1. What is the main limitation of BoW that embeddings address?
2. State the distributional hypothesis in one sentence.
3. What does it mean for two word vectors to be "close"?
4. What is the output of the Document Embedding widget?
5. Why use t-SNE *after* embedding? What does it show?
6. When would an embedding-based approach beat a BoW-based one for clustering?

6.3 Sentiment Analysis

1. What information does a sentiment dictionary hold?
2. How does the Sentiment Analysis widget compute a score?
3. Why do we keep verbs and adjectives in preprocessing for sentiment (unlike for LDA)?
4. Name one strength and one limitation of dictionary-based sentiment.
5. A tweet says *안 좋다* ("not good"). What will a simple dictionary method say about it, and why is it wrong?
6. How does a Box Plot by metadata help you interpret sentiment results?

6.4 Topic Modeling (LDA)

1. In LDA, a document is a mixture over _____ and a topic is a mixture over _____.
2. What do the β and θ matrices represent?
3. Why do we keep only nouns when preprocessing for LDA?
4. How does LDA differ from clustering in how it assigns documents to groups?
5. What does "the number of topics k is a research choice" mean in practice?
6. What does the λ slider in LDAvis control? Why start around 0.3?
7. A topic has top words that all appear in nearly every document. Is that topic useful? Why or why not?

---

*Focus on understanding the **why** behind each method, not just the how.*

A Glossary of Key Terms

Agglomerative clustering. Bottom-up hierarchical clustering: start with each document as its own cluster, merge the closest pair, repeat until one cluster remains.

β (beta, also ϕ). The topic–word matrix in LDA. Row k gives the probability of each word under topic k.

BERT. A family of transformer-based language models. Used by Orange's Document Embedding widget to produce document vectors. For Korean we use KLUE BERT.

Centroid. The mean vector of a cluster. K-means updates centroids iteratively.

Coherence score (c_v). A diagnostic for topic models. Measures how often the top words of a topic co-occur in short windows of text. Higher usually means more interpretable topics.

Cosine distance. $1 - \cos\theta$, where θ is the angle between two vectors. Length-insensitive; the default for text.

Dendrogram. Tree diagram produced by hierarchical clustering. Vertical height = the distance at which two clusters merged.

Dense vector. A vector with most entries non-zero. Word embeddings are dense; BoW vectors are not.

Dictionary-based sentiment. Scoring a document using predefined lists of positive and negative words.

Distributional hypothesis. Words that appear in similar contexts tend to have similar meanings. The foundation of word embeddings.

Document embedding. A single dense vector representing an entire document, usually produced by averaging word vectors or running the document through a transformer.

Euclidean distance. Straight-line distance between two vectors. Penalises differences in length; not ideal for text.

Hierarchical clustering. Builds a tree of clusters either agglomeratively (bottom-up) or divisively (top-down). We use agglomerative.

K-means. Partition-based clustering: commit to k, place k centroids, iteratively reassign documents and update centroids.

KLUE BERT. A Korean-pretrained BERT model used by Orange to embed Korean documents.

KNU dictionary. The Korean sentiment dictionary from Kunsan National University (Park et al. 2018), providing positive and negative word lists.

λ slider (LDAvis). Blends raw frequency ($\lambda = 1$) with distinctiveness ($\lambda = 0$) when ranking top words for a topic. Start at ~ 0.3.

Latent Dirichlet Allocation (LDA). A probabilistic topic model in which documents are mixtures of topics and topics are mixtures of words.

LDAvis. An interactive visualisation of a fitted LDA model: a 2D topic map on the left and a top-words bar chart on the right.

Linkage. The rule for measuring distance between two clusters (Ward, single, complete, average). Ward is our default.

Mixture model. A model in which each observation is a weighted combination of several components. LDA is a mixture model for documents.

Nearest Neighbors. An Orange widget that, given a query word or document vector, returns the most similar items by cosine similarity.

NNG / NNP. POS tags for common and proper nouns. Kept in preprocessing for both LDA and sentiment.

Silhouette score. A number in $[-1, 1]$ that indicates how well each point fits its assigned cluster versus its second-best alternative. Used to guide choice of k in k-means.

Sparse vector. A vector dominated by zeros. BoW and TF–IDF vectors are sparse.

t-SNE. A dimensionality-reduction technique that projects high-dimensional vectors to 2D while preserving local neighbourhoods. Used to visualise embeddings.

θ (theta, also γ). The document–topic matrix in LDA. Row d gives the probability of each topic in document d.

Topic. In LDA, a probability distribution over the vocabulary. The top words of the distribution are what you read to give it a label.

Unsupervised learning. Learning structure from data without labels. Clustering, embeddings, and LDA are all unsupervised.

VV / VA. POS tags for verbs and adjectives. Kept in preprocessing for *sentiment* but dropped for *LDA*.

Word embedding. A dense vector representation of a word learned from large text corpora. Words with similar meanings end up near each other.