BA2: Digital Korea

Week 4: From Words to Numbers

Steven Denney

Korean Studies
Leiden University

February 23, 2026

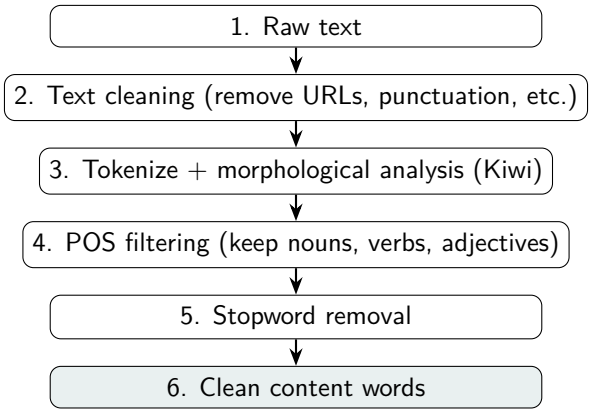
Today's Agenda

1. Review: Preprocessing pipeline & DataCamp
2. From words to numbers: BoW, TF, DF, and the DTM
3. TF-IDF: finding distinctive words
4. Concept review
5. Break
6. Hands-on: Preprocessing to visualization in Orange
7. Assignment & looking ahead

Review & Check-In

Preprocessing Review: The Pipeline

Last week we built a six-step pipeline. Let's walk through it again.



Why Korean Needs a Special Tool

English

- Spaces separate words
- "the economy" → two tokens
- Simple splitting works reasonably

Korean

- Particles attach to words
- 경제를 = 경제 + 를
- Space-splitting misses this

Kiwi is a morphological analyzer that breaks Korean into meaningful units.

Without it, every form of "economy" — 경제를, 경제의, 경제가 — counts as a separate word.

POS tags we keep

NNG (general nouns), **NNP** (proper nouns) by default.
Optionally: **VV** (verbs), **VA** (adjectives) for richer analysis.

Quick Recall

1. What does **tokenization** do?
   → Breaks text into individual units (tokens)
2. Why do we **filter by POS tags**?
   → To keep meaningful content words and remove grammatical particles, endings, etc.
3. What are **stopwords**?
   → High-frequency words that carry little analytical meaning (e.g., 있다, 하다, 것)
4. After all six steps, what do we have?
   → Clean content words, ready to count and analyze

About “Lemmatization”

Lemmatization and Morphological Analysis

The problem: The same word appears in many forms.

- English: *running*, *runs*, *ran* → **run**
- Korean: 배웁니다, 배워요, 배워, 배운다 → **배우다**

In English, this step is called **lemmatization** — reducing words to their dictionary form (the *lemma*).

For Korean, we call it **morphological analysis** (형태소 분석):

- Korean “glues” particles, endings, and suffixes onto stems
- A tokenizer like **Kiwi** decomposes words into morphemes and returns base forms automatically
- e.g., 배웁니다 → 배우/VV + ㅂ니다/EF

DataCamp: Introduction to R

Chapter 1: Intro to Basics & Chapter 2: Vectors

Key skills you practiced:

- Variables and assignment (`x <- 5`)
- Data types: numeric, character, logical
- Basic arithmetic in R
- Creating vectors with `c()`
- Naming, selecting, and comparing vector elements

Why this matters

These are the building blocks. When we eventually work with word counts and TF-IDF scores in R, they'll be stored as **vectors** — the same structures you practiced creating and manipulating.

Any questions or difficulties?

Today's Question

After preprocessing, we have clean content words.

How do we turn words into numbers a computer can work with?

From Words to Numbers: The Bag-of-Words Model

The Bag-of-Words Model

The metaphor: Cut every word from a document, drop them into a bag, shake it up.

What you keep

- Word identity
- Word count

What you lose

- Word order
- Grammar / context

Example

"국민이 경제를 걱정합니다" vs. "경제가 국민을 걱정합니다" — after preprocessing, both become {국민, 경제}. Identical in BoW.

At scale (749 speeches), frequency patterns alone are powerful enough to distinguish presidents, time periods, and speech types.

What BoW Produces: The Document-Term Matrix

BoW turns each document into a row of word counts. Stack the rows and you get the **document-term matrix (DTM)**:

	국민	경제	민주	올림픽	통일	안보
노태우 연설 1	5	3	2	1	4	0
김영삼 연설 1	4	6	0	0	2	1
김대중 연설 1	3	4	5	0	1	0
문재인 연설 1	6	2	3	0	0	2

Sparsity: Most cells are 0. Our real corpus has 749 rows $\times$ thousands of columns, 95–99% zeros. Normal and expected.

Key insight

The computer no longer sees Korean — it sees rows of numbers. Every method we learn from here on starts from this matrix.

TF and DF: Two Ways to Read the Matrix

Term Frequency (TF)
Read **one cell**: a word's count in one document. "How often does 경제 appear in *this* speech?"
Raw counts favor longer documents, so we can **normalize** by document length.

Document Frequency (DF)
Read **one column**: in how many documents does a word appear at all? "In how many speeches does 경제 appear?"
High DF = common. Low DF = rare.

Term	DF	Interpretation
국민	∼700 / 749	Almost every speech
올림픽	∼15 / 749	Very few speeches

Think

Which word tells you more about what makes a speech *distinctive*?

From Text to Numbers: The BoW Widget in Orange

In Orange, the **Bag of Words** widget builds the DTM for you. It offers two modes:

Stage	What you have	What can use it
After preprocessing	Corpus (tokenized text)	Word Cloud, browsing
After BoW (Count)	DTM with raw counts	Distances, clustering, PCA
After BoW (TF-IDF)	DTM with weighted values	Same, but distinctive words surfaced

Count vs. TF-IDF

Count: each cell = raw word count (the DTM you just saw).
TF-IDF: same matrix but *weighted* — common words down-weighted, rare words boosted. We'll learn how next.

TF-IDF: Finding Distinctive Words

The Problem with Raw Counts

국민 has a high count in almost every speech.

High count $\neq$ distinctive.

We need a method that:

- **Down-weights** words that appear in every document (common, generic)
- **Up-weights** words that are rare across the corpus but frequent in specific documents (distinctive)

$$\Rightarrow \textbf{TF-IDF}$$

IDF: The Intuition

Inverse Document Frequency measures how *rare* a word is across the corpus.

- Word appears in **every** document → IDF ≈ 0 (not distinctive)
- Word appears in **very few** documents → IDF is high (distinctive)

Term	Appears in...	Distinctive?
국민	700 / 749 speeches	Not really
올림픽	15 / 749 speeches	Very much

Core idea: Rarity = informativeness.

IDF: The Formula

$$\text{IDF}(t) = \log\left(\frac{N}{\text{DF}(t)}\right)$$

- N = total documents (749); $\text{DF}(t)$ = documents containing word t
- If DF is close to N, the ratio ≈ 1 and $\log \approx 0$ (common word)
- If DF is small, the ratio is large and $\log$ is high (rare word)
- The log just compresses the scale — don't worry about the math details

Term	DF	Calculation	IDF
국민	700 / 749	$\log(749/700)$	0.03 (very low)
올림픽	15 / 749	$\log(749/15)$	1.70 (high)

TF-IDF: Putting It Together

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

In plain English: How frequent is this word HERE $\times$ how rare is it EVERYWHERE ELSE?

Worked example — one speech

올림픽 appears 3 times (TF $= 3$), IDF $= 1.70$: TF-IDF $= 3 \times 1.70 = \mathbf{5.10}$ (distinctive!)

국민 appears 8 times (TF $= 8$), IDF $= 0.03$: TF-IDF $= 8 \times 0.03 = \mathbf{0.24}$ (generic)

The takeaway

TF-IDF answers: **what words make THIS document different from the others?**

What TF-IDF Tells You

High TF-IDF:

- Frequent in this document
- Rare across the corpus
- $\Rightarrow$ *Distinctive*

Low TF-IDF:

- May be frequent, but...
- Appears in many documents
- $\Rightarrow$ *Generic*

- Orange calculates TF-IDF **automatically** in the Bag of Words widget
- You do not need to compute it by hand
- But understanding the logic matters for **interpreting your results**

Key Concepts Review

Make sure you can explain each of these:

Concept	What it means
BoW	Bag of Words — represent text by which words appear and how often, ignoring word order
TF	Term Frequency — how many times a word appears in one document (normalize by document length for fair comparison)
DF	Document Frequency — in how many documents does a word appear? High DF = common word
DTM	Document-Term Matrix — the table (documents $\times$ words) that the BoW widget creates. This is how the computer "sees" your corpus
IDF	Inverse Document Frequency — measures word rarity across the corpus. Rare words get high IDF
TF-IDF	TF $\times$ IDF — highlights words that are frequent HERE but rare EVERYWHERE ELSE

Break

We will resume in 10 minutes.

Hands-On: Preprocessing to Visualization

Pair Activity: Overview

Work in pairs. We'll walk through this together.

Part 1: Get the preprocessing pipeline running in Orange

Part 2: Build visualizations and compare Count vs. TF-IDF

What you need

- Orange Data Mining open on your laptop
- Presidential speeches CSV (from the Data page)
- Preprocessing script for your OS (from the Data page)

Part 1: Get Preprocessing Running

Build this workflow step by step:

0. **File** widget: load `president_speeches_democratic_era.csv`
1. **Corpus** widget: connect to File; set the text column
2. **Python Script** widget: paste the preprocessing script for your OS
   - Set `TEXT_COLUMN` to match your corpus column name
   - Run the script — this may take a minute the first time
3. **Corpus** widget (second one): reload to pick up the processed text
4. **Preprocess Text**: set tokenization to *by whitespace*; load the stopword list

Check your work

Browse the output of Preprocess Text. Do you see clean Korean content words?

Part 2: Count vs. TF-IDF — Three Views

Remember: before the Bag of Words widget, you have text. After it, you have numbers.

Step 1: Raw text → Word Cloud

- Connect a **Word Cloud** directly to Preprocess Text output
- This shows raw word frequencies from the text itself

Step 2: BoW (Count) → Word Cloud

- Add a **Bag of Words** widget set to **Count**
- Connect a new Word Cloud — compare with Step 1

Step 3: BoW (TF-IDF) → Word Cloud

- Switch Bag of Words to **TF-IDF** — what changes?

Also try: **Bar Plot**, **Distributions**, or **Statistics** widgets for deeper exploration.

Discuss with Your Partner

1. What words dominate the raw-count / Count word clouds?
2. What happens to those words when you switch to TF-IDF?
3. Name one word that becomes *more* prominent with TF-IDF. Why?
4. Try filtering by president — what words are distinctive for each?

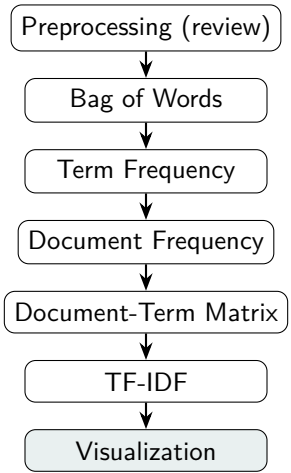
Connect back to the concepts

- **Count** shows you what's *frequent* (TF)
- **TF-IDF** shows you what's *distinctive* (frequent here, rare elsewhere)
- The word cloud changed because TF-IDF down-weighted the words with high DF

We'll debrief as a class after.

Wrapping Up

Today's Arc


Preprocessing (review)
Bag of Words
Term Frequency
Document Frequency
Document-Term Matrix
TF-IDF
Visualization

Assignment

Orange workflow (due before next class):

1. Corpus → Python Script → Preprocess Text → Bag of Words (TF-IDF)
2. Connect at least **two visualization widgets**
3. Take a screenshot of your workflow

Submit: Create a `week04/` folder in your repository. Add your `.ows` file + screenshot. Commit and push via GitHub Desktop.

R Programming: DataCamp: Introduction to R — Chapters 4 (Factors) & 5 (Data Frames). Due by start of next class.

Readings

Grimmer, Roberts & Stewart — Chapters 6–7.

Looking Ahead

Week 5: Practice & Deepen — Hands-On Lab

- Extended practice with BoW/TF-IDF workflows
- Comparing presidents, interpreting visualizations
- Come with your assignment done — we'll build on it

Week 6: Midterm Review & Assessment

- Everything from Weeks 1–5

Thank you!

s.c.denney@hum.leidenuniv.nl