Data Preprocessing

Chapter 2

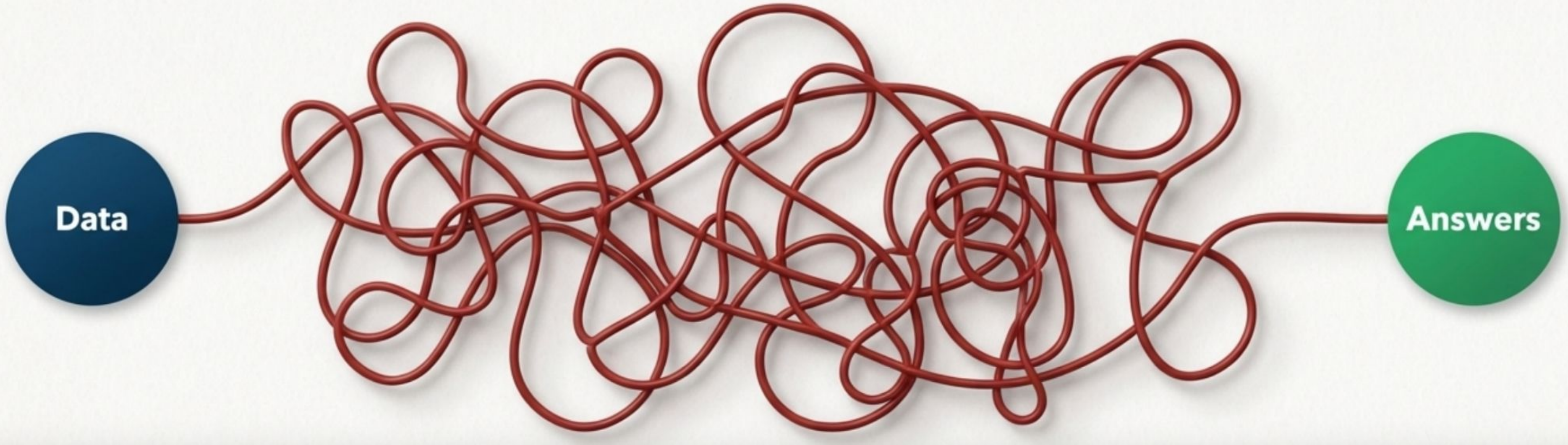
Data

In NLP, data is any form of natural language content (text or speech)

that a computer system can analyze:

- to perform tasks like translation, classification, summarization, sentiment analysis, or question answering

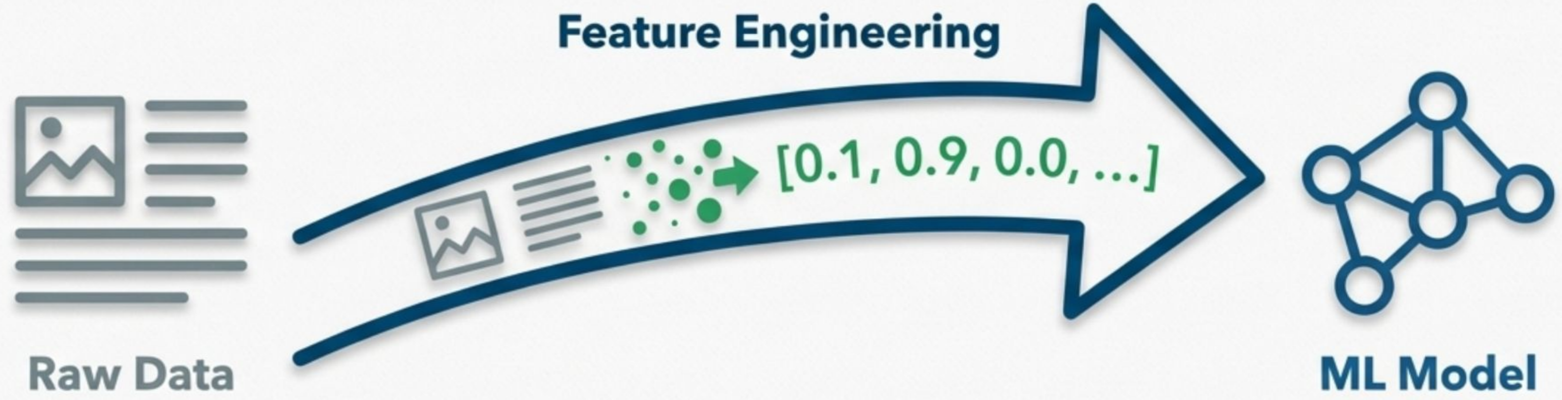
- Workflows with data are frequently multistage, iterative processes. The path is full of false starts and dead ends.
- Raw data is often not numeric. For example, the action “Alice bought *The Lord of the Rings* on Wednesday” is not numeric.
- Machine learning models, however, require numeric inputs. This creates a fundamental gap.



Feature Engineering

What is a Feature? A feature is a numeric representation of raw data.

What is Feature Engineering? The process of formulating the most appropriate features given the data, the model, and the task.



"Data preprocessing and feature engineering has been found to be the main driver of model performance in many applications." – Andreas C. Müller

Types of Data in NLP

1. Textual Data

- News articles
- Social media posts (e.g., tweets, comments)
- Product reviews
- Books, emails, and transcripts
- Chat logs and FAQs

2. Speech Data

- Voice commands (used in speech-to-text and virtual assistants)
- Phone call recordings
- Podcasts or meeting recordings

3. Structured NLP Datasets

Datasets labeled for specific tasks, such as:

- Sentiment labels (positive/negative)
- Named Entity labels (e.g., PERSON, LOCATION)
- Part-of-Speech tags (POS)
- Question-answer pairs

Data Preprocessing

It is a crucial step in any Natural Language Processing (NLP) pipeline

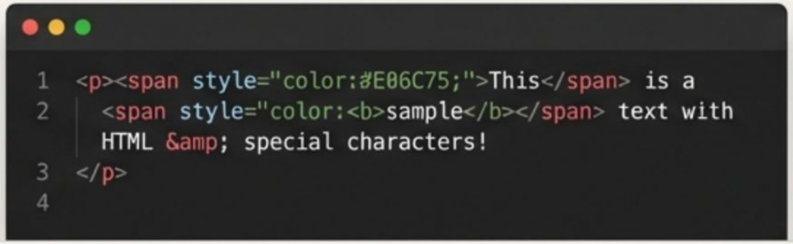
Raw text data is often noisy, inconsistent, and unstructured, making it unsuitable for machine learning models unless it is cleaned and transformed

Goals:

- Remove irrelevant information
- Normalize text to a standard format
- Improve model performance by reducing vocabulary noise

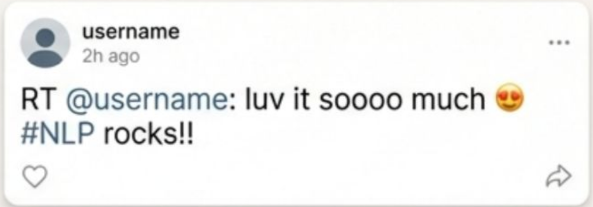
Raw text is inherently noisy, inconsistent, and lacks the structure required for computational analysis. The goal of **preprocessing** is to transform this chaos into clean, structured features.

Web Data



```
<p><span style="color:#E06C75;">This</span> is a
  <span style="color:<b>sample</b></span> text with
  HTML &amp; special characters!
</p>
```

Social Media Content



username
2h ago

RT @username: luv it soooo much 😍 #NLP rocks!!

Ambiguous Punctuation

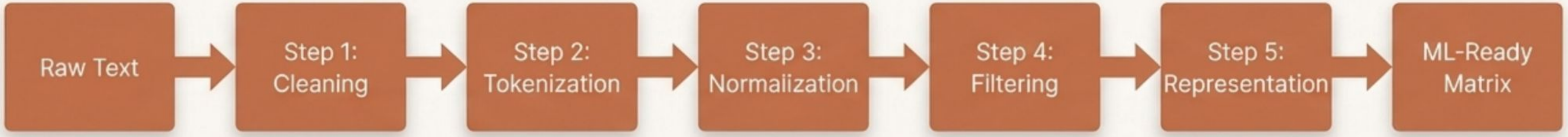
...plans to sublease its current headquarters at 55 Water St. . A spokesman declined...

Inflected Language

The runner runs quickly through the running track.

Without a systematic process, machines cannot distinguish signal from noise.

For decades, NLP practitioners have relied on a sequential pipeline to systematically clean and structure text for traditional machine learning models.



Each stage in this pipeline is a deliberate act of 'sculpting'—chipping away noise and refining the material to reveal the underlying features. We will walk through each of these foundational steps.

Basic Text Cleaning Steps

Lowercasing:

Convert all text to lowercase to avoid treating the same words differently

Input: Natural Language Processing

Output: natural language processing

Step 1: Cleaning the Canvas

The first step is to remove elements that are artifacts of the medium, not the message itself. This includes markup, links, and non-linguistic characters.

Before		After
`<p>This is a <b>sample</b> text...</p>`	→	This is a sample text...

```
Method: re.sub('<.*?>', '', text)
```

Removing URLs &

Check our site at https://example.com or email info@example.com	→	Check our site at **[URL]** or email **[EMAIL]**

```
Method: re.sub(r'http\S+|www\S+', '[URL]', text)
```

Removing Special Characters

...with HTML &amp; special	→	...with HTML special characters

```
Method: re.sub('[^a-\s]', '', text)
```

```
Method: re.sub('[^a-zA-Z0-9\s]', '', text)
```

This initial cleanup is foundational and prepares the text for more nuanced linguistic processing.

Basic Text Cleaning Steps

Remove Punctuation:

Punctuation does not usually contribute to semantic meaning in many NLP tasks

Input: Hello World!?

Output: Hello World

Basic Text Cleaning Steps

Remove Numbers:

In some cases, numbers are removed unless relevant to the analysis (e.g., dates, amounts)

Input: There are 365 days in a year

Output: There are days in a year

Basic Text Cleaning Steps

Remove extra white spaces:

Extra whitespaces are removed to standardize input

Input: Natural   Language Processing

Output: natural language processing

Tokenization

Tokenization is the task of splitting a text into meaningful segments,called tokens.

These segments could be words, punctuation, numbers, or other special characters that are the building blocks of a sentence.

Sentence Tokenization: Breaks text into sentences.

Word Tokenization: Breaks text into words.

Tokenization breaks down a stream of text into its constituent parts—words or sentences—which become the fundamental units for analysis.

Word Tokenization

Input:
"This is a sample sentence."

Output:
['This'] ['is'] ['a'] ['sample'] ['sentence'] ['.']

Sentence Tokenization

Input:
"This is the first sentence. And here's another one."

Output:
['This is the first sentence.'] ["And here's another one."]

Stop Words Removal

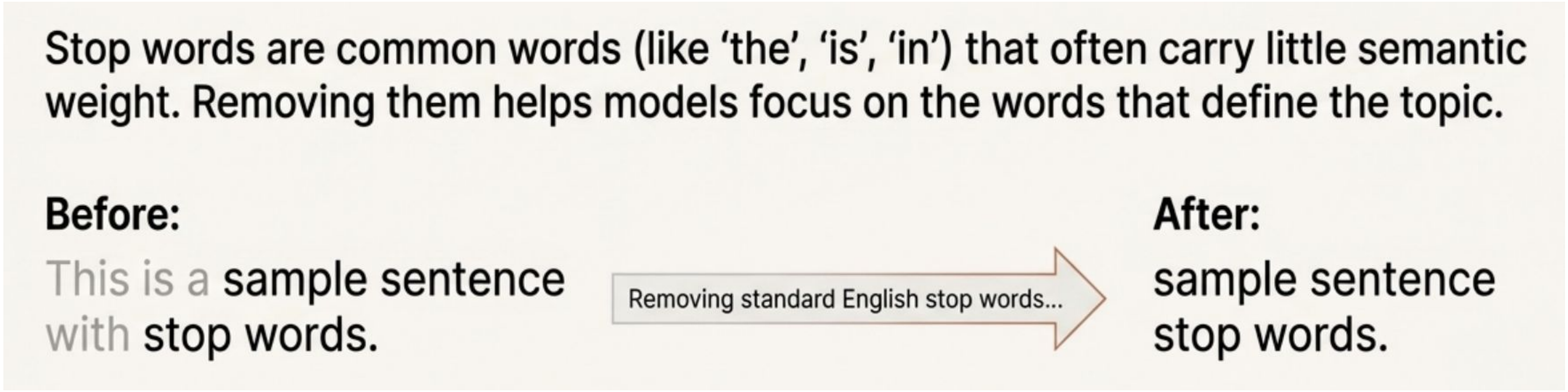
Some words that frequently appear in texts but they don’t contribute much to the overall meaning

Stop words are common words (e.g., "is", "and", "the") that carry less meaning and are often removed.

Example: There is a tree near the house

Without stopwords: tree near house

Stop words



Stop words are common words (like ‘the’, ‘is’, ‘in’) that often carry little semantic weight. Removing them helps models focus on the words that define the topic.

Before:

This is a sample sentence with stop words.

Removing standard English stop words...

After:

sample sentence stop words.

Stemming

It is a rules-based process that reduces words to their stem by using predefined rules to systematically strip away prefixes, suffixes or other affixes from words to reduce them to their base form

Stemming approach tends to reduce the word to a root form without any context

Input: "playing", "played", "player"

Output: "play", "play", "play"

Tool: Porter Stemmer, Snowball Stemmer, Lancaster Stemmer particularly effective in search systems and text retrieval applications where an approximate matching of words is sufficient

by reducing words to their root forms, stemming enhances the efficiency of these systems while managing large volumes of text

Lemmatization

Reduces words to their base or dictionary form (lemma) with correct context.

often provides better results

Input: "am", "are", "is"

Output: "be"

Tool: WordNet Lemmatizer

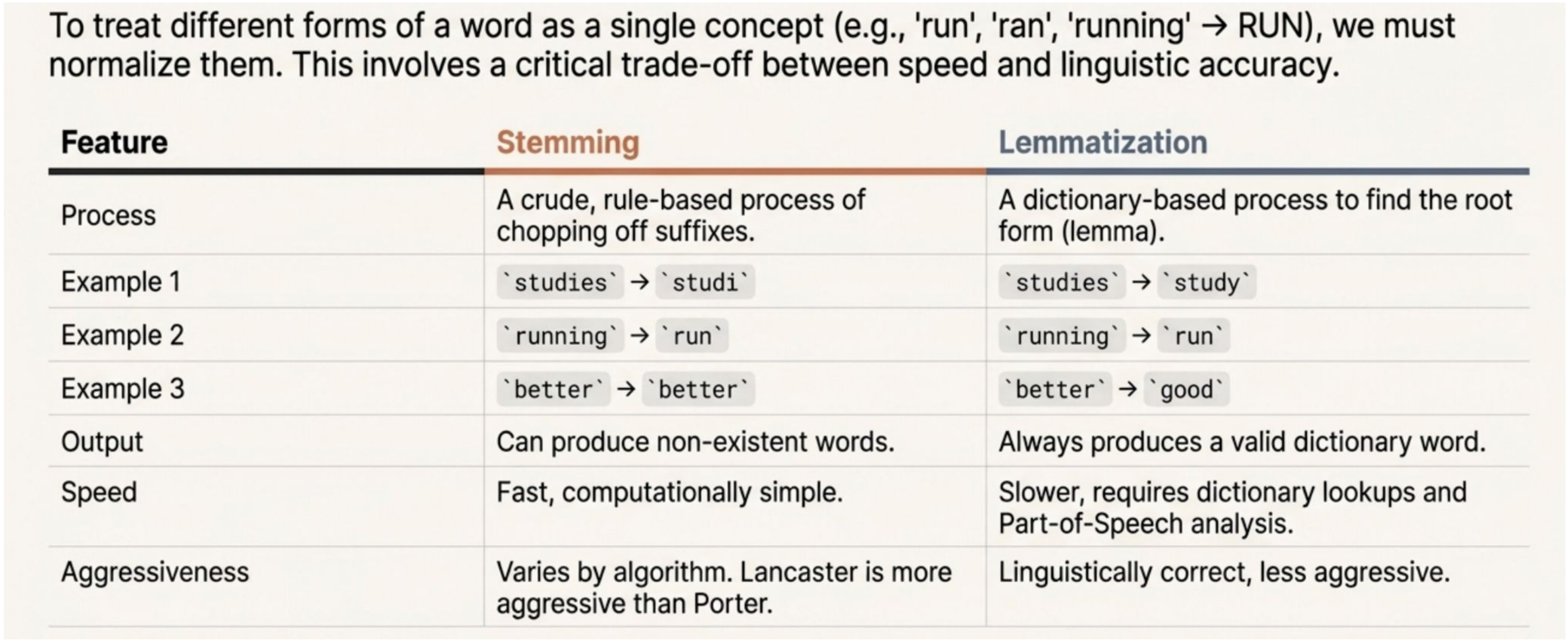
Lemmatization

Lemmatization involves several steps:

Part-of-Speech (POS) Tagging: Identifying the grammatical category of each word (e.g., noun, verb, adjective).

Morphological Analysis: Analyzing the structure of the word to understand its root form.

Dictionary Lookup: Using a predefined vocabulary to find the lemmaof the word.



To treat different forms of a word as a single concept (e.g., 'run', 'ran', 'running' → RUN), we must normalize them. This involves a critical trade-off between speed and linguistic accuracy.

Feature	Stemming	Lemmatization
Process	A crude, rule-based process of chopping off suffixes.	A dictionary-based process to find the root form (lemma).
Example 1	`studies` → `studi`	`studies` → `study`
Example 2	`running` → `run`	`running` → `run`
Example 3	`better` → `better`	`better` → `good`
Output	Can produce non-existent words.	Always produces a valid dictionary word.
Speed	Fast, computationally simple.	Slower, requires dictionary lookups and Part-of-Speech analysis.
Aggressiveness	Varies by algorithm. Lancaster is more aggressive than Porter.	Linguistically correct, less aggressive.

POS Tagging

Identifies the grammatical category of each word using:

Dictionary lookup for known words

Suffix rules for unknown words (e.g., -ing → VERB, -tion → NOUN)

Capitalization detection for proper nouns

POS Tagging

Tag	Category	Examples
Noun	Person, place, thing	Dog, city, happiness
Verb	Action or state	Run, is, become
Adj	Describes a noun	Beautiful, quick, large
Adv	Describes a verb	Quickly, very, well
Determiner	determines/specifies nouns	The , a, this, that
Pron	Replaces a noun	He, she, it
prep	Show relationship	in , on, at, with
conj	Connects words	And, but, or

Morphological Analysis

Breaks words into structural components

- Prefix (un-, re-, dis-, pre-, anti-)
- Root/Base (core meaning)
- Suffix (-ing, -ed, -tion, -ness, -able)

Example: unhappiness → un- (not) + happy (root) + -ness (state of) = NOUN

Dictionary Lookup Lemmatization

Uses a predefined vocabulary to find base forms:

Irregular verbs: went → go, saw → see, thought → think

Irregular plurals: children → child, phenomena → phenomenon

Comparatives: better → good, worst → bad

Rule-based fallback for unknown words

Lemmatization

It is essential for tasks that demand a deeper understanding of language, such as question answering, text summarization, and detailed text analysis

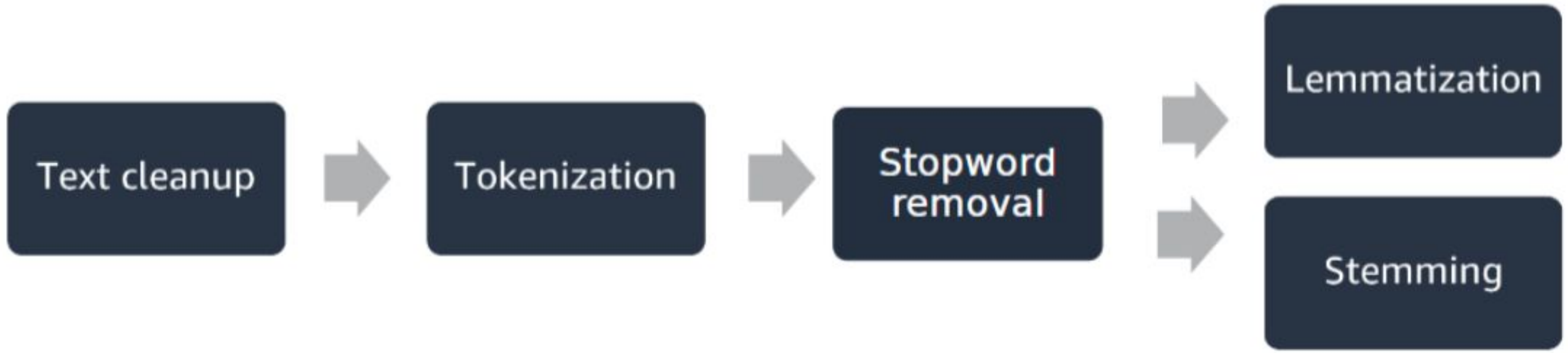
ensures that words are reduced to their accurate base forms, preserving their meaning and grammatical context

Original: There is nothing either good or bad but thinking makes it so.

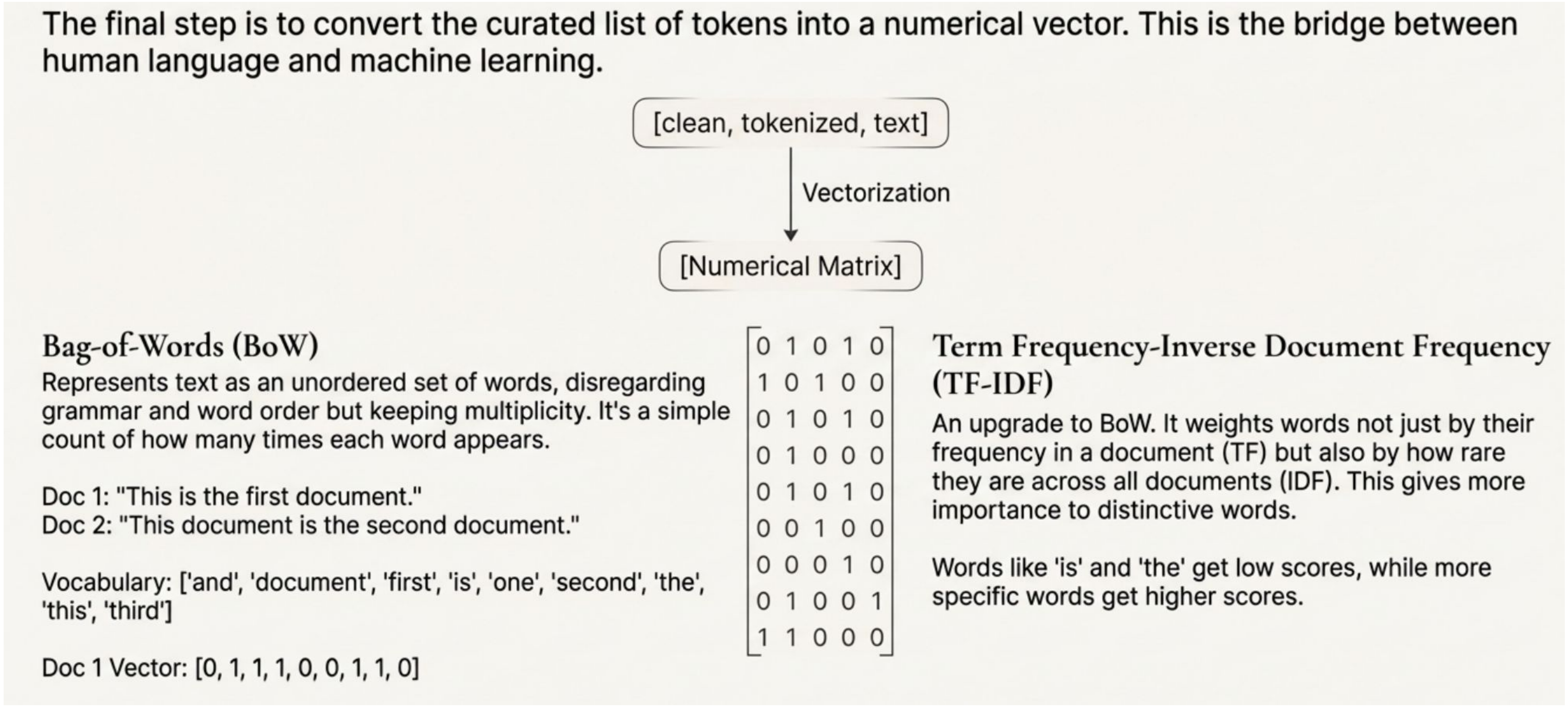
Tokenized: [’There’, ’is’, ’nothing’, ’either’, ’good’, ’or’, ’bad’, ’but’, ’thinking’, ’makes’, ’it’, ’so’, ’.’]

Lemmatized: There be nothing either good or bad but think make it so .

Basic Data Preprocessing tasks



Turning text into numbers



The final step is to convert the curated list of tokens into a numerical vector. This is the bridge between human language and machine learning.

$$\begin{tikzcd}
\text{[clean, tokenized, text]} \arrow[d, "\text{Vectorization}"] \\
\text{[Numerical Matrix]}
\end{tikzcd}$$

Bag-of-Words (BoW)

Represents text as an unordered set of words, disregarding grammar and word order but keeping multiplicity. It's a simple count of how many times each word appears.

Doc 1: "This is the first document."
Doc 2: "This document is the second document."

Vocabulary: ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'this', 'third']

Doc 1 Vector: [0, 1, 1, 1, 0, 0, 1, 1, 0]

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Term Frequency-Inverse Document Frequency (TF-IDF)

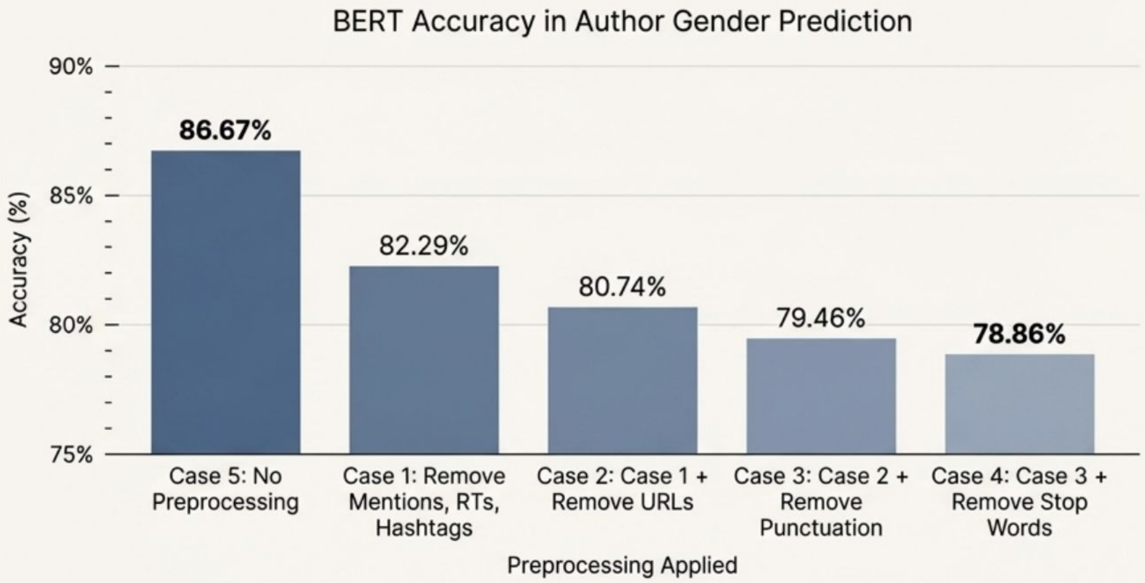
An upgrade to BoW. It weights words not just by their frequency in a document (TF) but also by how rare they are across all documents (IDF). This gives more importance to distinctive words.

Words like 'is' and 'the' get low scores, while more specific words get higher scores.

For transformer model we donot use preprocessing

The Surprising Truth: When Preprocessing Hurts Performance

An experimental study on author gender profiling using the BERT model revealed a counter-intuitive finding: more preprocessing led to worse results.



For BERT, aggressive preprocessing destroyed valuable signal, reducing accuracy by nearly 8 percentage points.