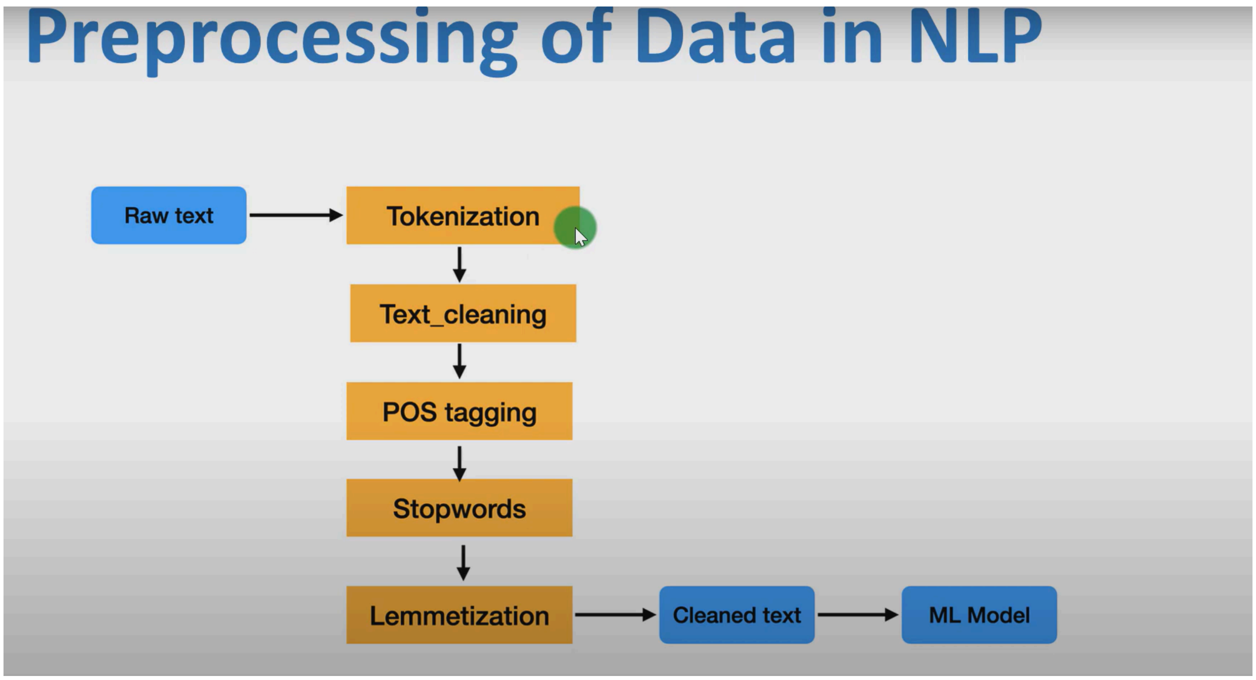


Tokenization

Splits raw text into smaller units called tokens (words or subwords).
This helps in analyzing the text at the word level. —Hugging Face

Text Cleaning

Removes unwanted characters like punctuation, numbers, and special symbols.
Ensures uniform formatting for consistent processing. — Beautiful Soap

POS Tagging

Assigns part-of-speech labels (noun, verb, etc.) to each token
Helps understand the grammatical structure of the sentence. –SpaCu

Stopwords Removal

Filters out common words (like "the", "is", "and") that carry little meaning.
Reduces noise and dimensionality in the text. –SpaCy

Lemmatization

Converts words to their base or dictionary form (e.g., "running" → "run").
Ensures consistent representation of similar words. –SpaCy , WordNetLemmitizer

Step Order	Pros (✅)	Cons (❌)
1. Stopwords Removal → POS Tagging	• Faster processing • Fewer tokens to tag	• Loss of context • Reduced tagging accuracy
2. POS Tagging → Stopwords Removal	• Accurate POS tagging • Smarter stopword filtering	• Slightly slower • Higher memory usage
🔷 Best Practice	• Tag first, filter later for better accuracy and flexibility	–

Step Order	Pros (✅)	Cons (❌)
1. Stopwords Removal → Lemmatization	• Faster lemmatization • Fewer tokens processed	• Important words may be lost • Lemma matching may fail
2. Lemmatization → Stopwords Removal	• Better stopword match • Retains semantic richness	• Slightly slower • Processes more tokens initially
🔷 Best Practice	• Lemmatize first, then remove for better results	–

Stemming cuts words to their root form using simple rules, often producing non-words.
Lemmatization reduces words to their meaningful base form (lemma) using vocabulary and context.

1. **Stemming** is faster but less accurate, while **Lemmatization** is slower but more precise.
2. **Stemming** ignores the word's part of speech, whereas **Lemmatization** uses it to find the correct base form.

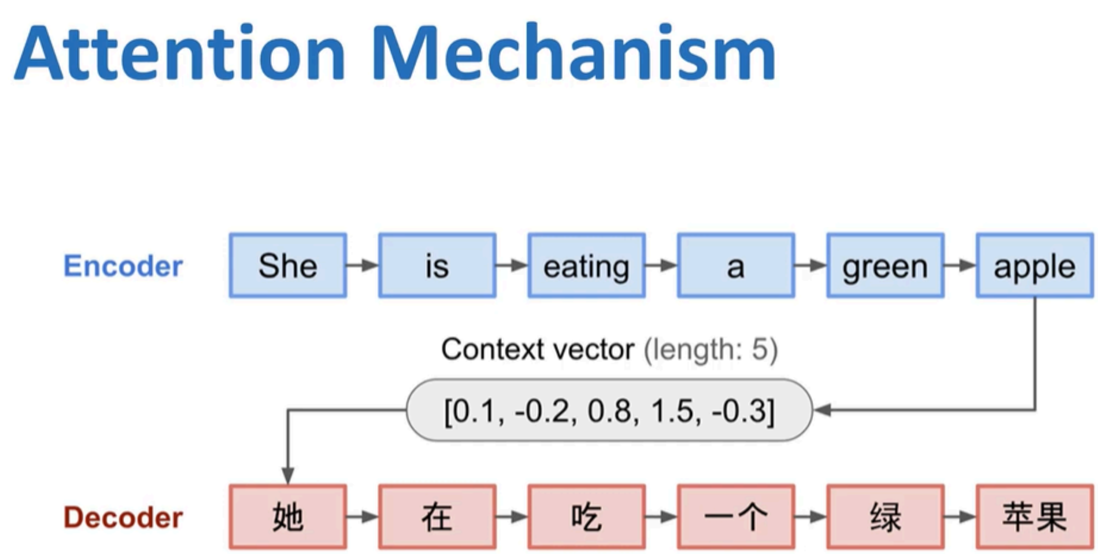
Feature	Stemming	Lemmatization
Output	May not be a valid word (e.g., "happi")	Always a valid word (e.g., "happy")
Speed	Faster	Slower (due to POS analysis)
Context Awareness	No	Yes (requires POS tags)
Use Case	Search engines, quick preprocessing	Chatbots, sentiment analysis

Generally we apply stemming after removing stop words. More efficient —Tool : Porter Scanner

Applications:

- **Sentiment Analysis:** Detecting emotions in text.
- **Question Answering:** Extracting answers from context (e.g., chatbots).
- **Topic Modeling:** Identifying themes in documents (e.g., LDA).

WORD EMBEDINGS: A word embedding is **a learned representation for text where words that have the same meaning have a similar representation**.



Deep Learning in NLP - Lil Details

Word Embeddings:

- Word2Vec & GloVe turn words into numbers (vectors) so computers understand meaning by position in space.

Attention Mechanisms:

- Help models focus on important words in a sentence (like “Transformers” do).

Encoder-Decoder Models:

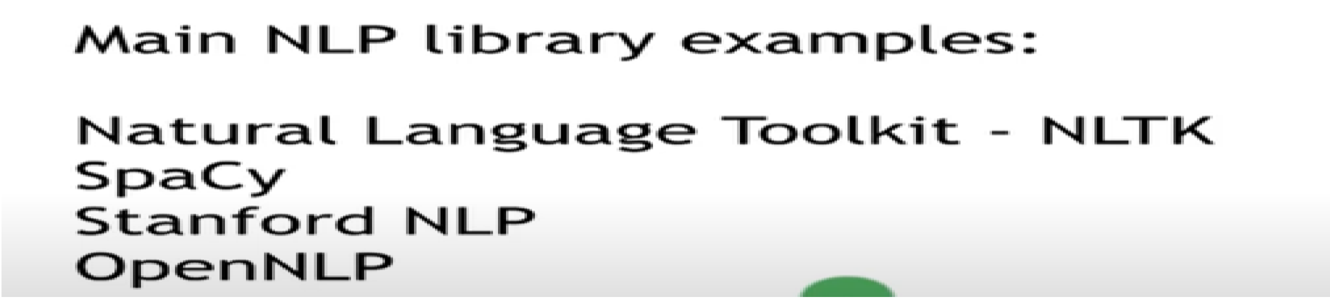
- Two-part models: one reads input, the other generates output (used in translation).

Large Language Models (LLMs):

- Big pretrained models (GPT-3/4, BERT) that can do many language tasks.

Prompt Engineering:

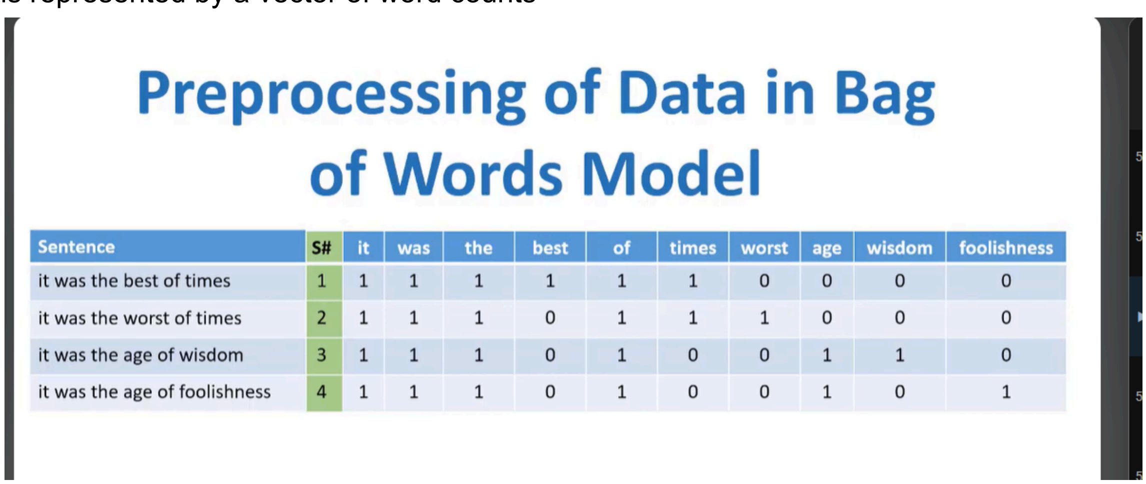
- Crafting specific inputs to get desired answers from LLMs (like asking “Translate this: …”).



Bag of Words (BoW)

- **What is it?**
  A simple way to represent text by counting how many times each word appears, ignoring grammar and order.

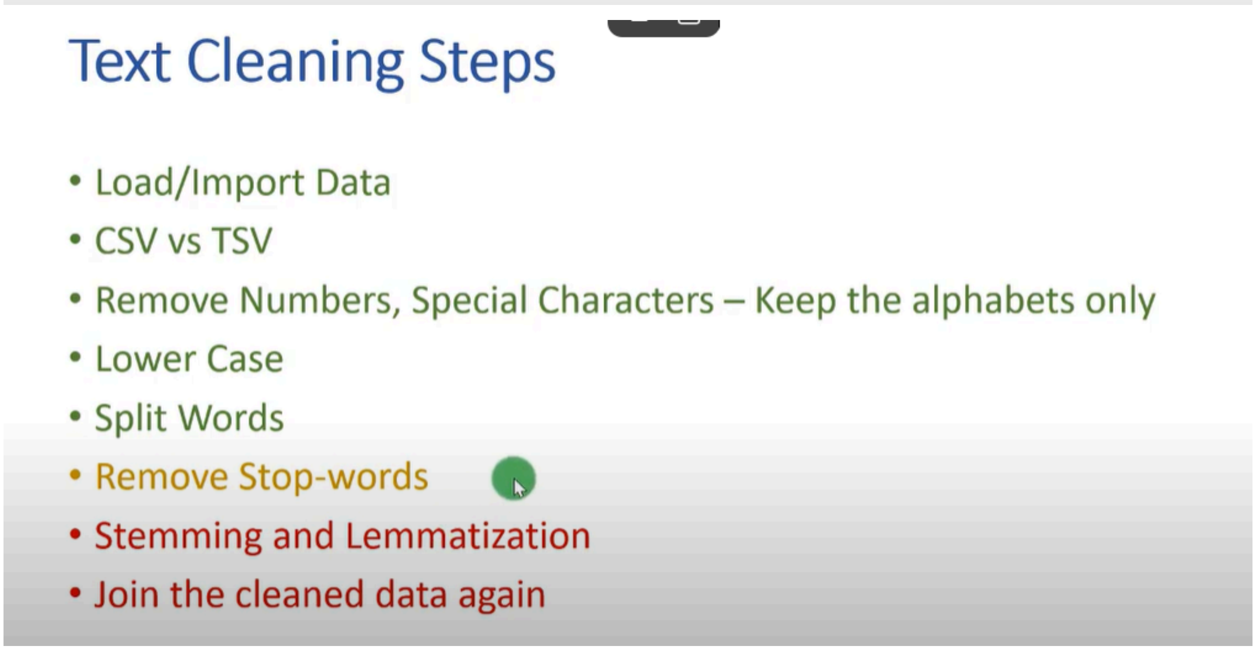
- **How it works:**
  It creates a “bag” (collection) of all unique words in the text corpus, then each document is represented by a vector of word counts



Preprocessing of Data in Bag of Words Model

Sentence	S#	it	was	the	best	of	times	worst	age	wisdom	foolishness
it was the best of times	1	1	1	1	1	1	1	0	0	0	0
it was the worst of times	2	1	1	1	0	1	1	1	0	0	0
it was the age of wisdom	3	1	1	1	0	1	0	0	1	1	0
it was the age of foolishness	4	1	1	1	0	1	0	0	1	0	1

1. Clean texts to prepare them for the Machine Learning models,
2. Create a Bag of Words model,
3. Apply Machine Learning models onto this Bag of Worlds model.

- 

🧮 CountVectorizer and Sparse Matrix

- CountVectorizer converts a set of text documents into a **sparse matrix**.
- The **dimensions** of the sparse matrix depend on the number of **distinct words** in the corpus.
- This results in **very large dimensions**, especially when the corpus contains a lot of unique words, leading to **computational challenges**.

🔻 Need for Dimensionality Reduction

- Example: If a word like **"Tim"** appears only in one sentence and not in the remaining 10,000, a whole column is dedicated to it with just a single '1' and the rest '0's.
- Words like **"Tim"**, **"Rick"**, and **"Steve"** (proper nouns) add no value in determining the **sentiment or polarity** of the sentence.
- These rare and irrelevant words become **computational load**.

🛠️ Solution: Using `max_features` in CountVectorizer

- Set a **maximum number of features** (e.g., `max_features = 1500`) to limit the number of words (columns) retained.
- Example: Original sparse matrix had **1565 columns** (i.e., 1565 unique words).
  - With `max_features=1500`, the **least frequent 65 words** are removed.
  - These often include uncommon proper nouns and rare terms.
- This is a **soft method** of dimensionality reduction.

⚙️ Why Reduce Columns, Not Rows?

- Each **row** represents a document or review (i.e., a data point).
- Removing rows would mean **losing actual data**.
- Reducing **columns** (words/features) helps in reducing noise while keeping important data intact.

🧪 Other Dimensionality Reduction Techniques

- More **systematic and advanced** methods:
  - **Principal Component Analysis (PCA)**
  - **Singular Value Decomposition (SVD)**
- These methods go beyond frequency and analyze the **contribution of each word** to the overall data structure.

1. **How does removing stopwords affect model performance? Can it sometimes remove important context?**
   It reduces noise and dimensionality, but may remove contextually important words in some tasks like sentiment analysis.

2. **Why is lemmatization preferred over stemming? What changes does it make?**
   Lemmatization returns the proper base form (e.g., "running" → "run"), making text more semantically accurate than stemming.

3. **What effect can incorrect POS tagging have on NLP tasks?**
   It can lead to incorrect parsing, entity recognition, or translation errors due to misunderstood grammatical roles.

4. **How does text cleaning impact data quality? Can it go too far?**
   It standardizes input and reduces noise, but over-cleaning might remove meaningful

symbols or characters.

5. **What are the limitations of Bag of Words (BoW)?**
   It ignores grammar and word order, and creates sparse high-dimensional vectors that lack semantic meaning.

6. **How do word embeddings improve over BoW?**
   They capture semantic relationships by placing similar words closer in vector space, improving model understanding.
7. ***Why are attention mechanisms important in NLP?***
   They help the model focus on key parts of input text, improving performance in tasks like translation and summarization.

8. ***What changes do encoder-decoder models introduce?***
   They split processing into understanding (encoder) and generating (decoder), enabling complex tasks like translation.