4.Bag of Words - Detailed Implementation

Step-by-Step Process with Example

Initial Dataset

```
Sentence 1 (S₁): "He is a good boy"        → Output: 1 (Positive)
Sentence 2 (S₂): "She is a good girl"      → Output: 1 (Positive)
Sentence 3 (S₃): "Boy and girl are good"   → Output: 1 (Positive)
```

Step 1: Text Preprocessing

1.1 Lowercase Conversion

```
S₁: "he is a good boy"
S₂: "she is a good girl"
S₃: "boy and girl are good"
```

1.2 Remove Stop Words

Stop words removed: {he, she, is, a, and, are}

After preprocessing:

```
S₁: "good boy"
S₂: "good girl"
S₃: "boy girl good"
```

Step 2: Build Vocabulary

Extract all unique words and count their frequency across all sentences:

Word	Frequency	Rank
good	3	1
boy	2	2
girl	2	3

Vocabulary = {good, boy, girl}

Vocabulary Size = 3

Important Notes:

- Words are ordered by **descending frequency**
- In larger datasets, you can select **top N features** (e.g., top 10, top 100)
- Words appearing only once can be excluded to reduce noise

Step 3: Create Feature Vectors

Using vocabulary as features: **[good, boy, girl]**

Sentence 1: "good boy"

- good present → 1
- boy present → 1
- girl absent → 0

Vector: [1, 1, 0]

Sentence 2: "good girl"

- good present → 1
- boy absent → 0
- girl present → 1

Vector: [1, 0, 1]

Sentence 3: "boy girl good"

- good present → 1
- boy present → 1
- girl present → 1

Vector: [1, 1, 1]

Final Representation

Sentence	good	boy	girl	Output
S_1	1	1	0	1
S_2	1	0	1	1
S_3	1	1	1	1

These vectors can now be fed into any machine learning algorithm for classification tasks.

Binary vs Regular Bag of Words

Example: "good good girl"

Regular BoW (Count-based):

- Counts word frequency
- good appears 2 times → value = 2
- **Vector: [2, 0, 1]**

Binary BoW:

- Only checks presence/absence
- good appears 2 times → value = 1 (forced)
- **Vector: [1, 0, 1]**

Type	Formula	Example Vector
Regular BoW	Count frequency	[2, 0, 1]
Binary BoW	Presence = 1, Absence = 0	[1, 0, 1]

Advantages of Bag of Words

1. Simple and Intuitive

- Easy to understand and implement
- Straightforward vectorization process

2. Fixed-Size Input Vectors

- **All sentences → same dimension vectors**
- Vocabulary size = 3 → Every sentence becomes 3-dimensional
- Essential for machine learning algorithms
- Solves the variable-length problem of One-Hot Encoding

3. Better than One-Hot for ML

- One vector per sentence (vs. multiple vectors per sentence)
- Ready for direct ML algorithm input

Disadvantages of Bag of Words

1. Sparse Matrix Problem

- **Still produces mostly zeros**
- Example with 50,000 vocabulary:
  - Each sentence → 50,000-dimensional vector
  - Only a few positions are non-zero
  - Leads to high memory usage and overfitting

2. Loss of Word Order

Critical Issue: Word order changes meaning!

Example:

```
Original: "Boy girl good" → [1, 1, 1]
Reordered: "Good boy girl" → [1, 1, 1]
```

Both produce the same vector, but:

- Ordering in vocabulary: [good, boy, girl] (by frequency)
- Original sentence order is completely lost
- **Semantic meaning gets distorted**

3. Out of Vocabulary (OOV) Problem

Training Data Vocabulary: {good, boy, girl}

Test Sentence: "boy girl good school"

- "school" is NOT in vocabulary
- Gets **ignored completely**
- Vector becomes: [1, 1, 1] (school information lost)
- May lose critical information

4. No Semantic Meaning Captured

Problem 4a: Equal Importance

All present words get value = 1

- Cannot distinguish which words are more important
- "good" and "boy" treated equally
- No contextual understanding

Problem 4b: Opposite Meanings Look Similar

Example:

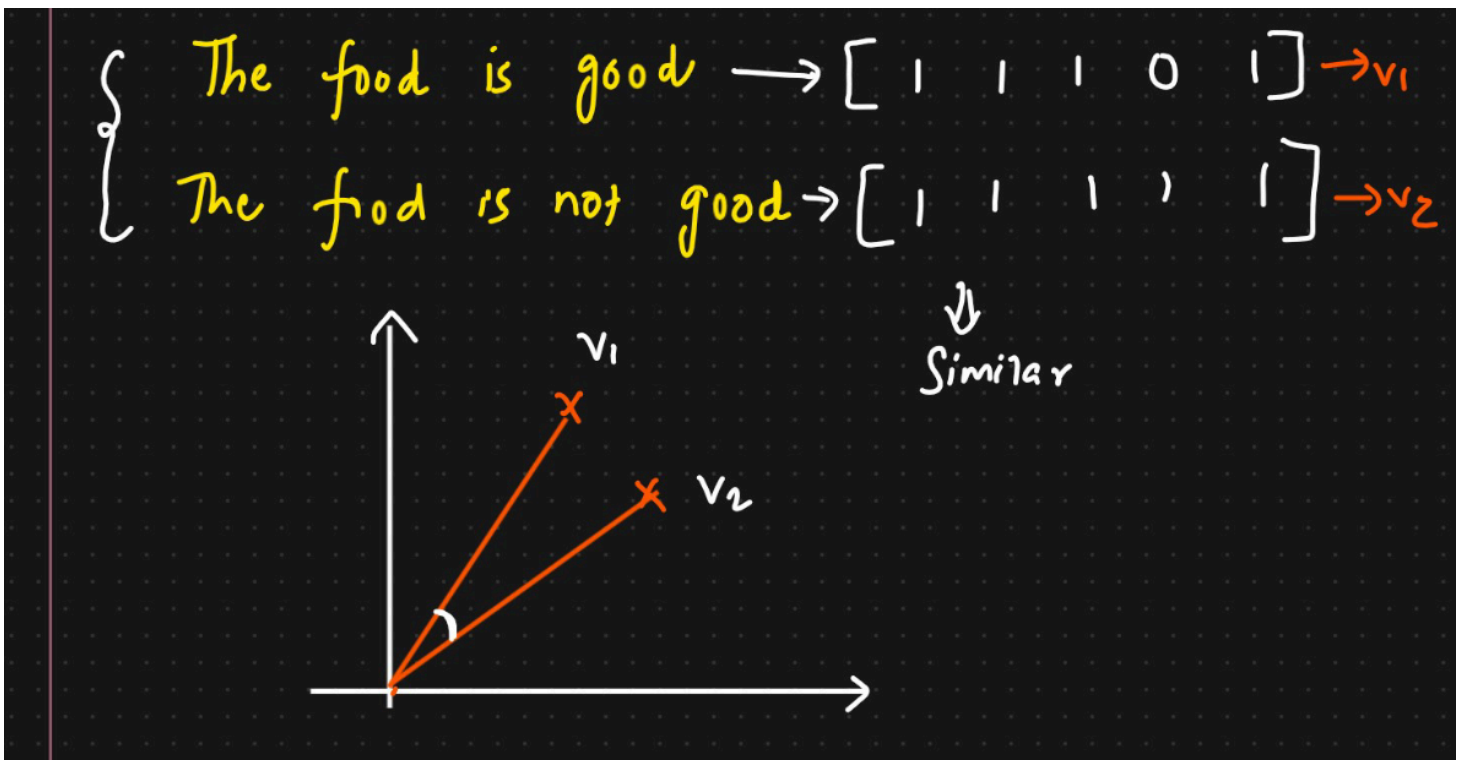
```
S₁: "the food is good"     → [1, 1, 1, 0, 1]
                                ↓  ↓  ↓  ↓  ↓
                               the food is not good

S₂: "the food is not good" → [1, 1, 1, 1, 1]
```

Vocabulary: [the, food, is, not, good]

Cosine Similarity Calculation:

- Only 1 position differs (not: 0 vs 1)
- Vectors appear **very similar**
- Angle between vectors is small
- Algorithm thinks sentences are similar

Reality: They are **complete opposites!**



This is a **major flaw** that gets addressed in advanced techniques like Word2Vec.

Summary Comparison: One-Hot vs Bag of Words

Feature	One-Hot Encoding	Bag of Words
Vectors per Document	Multiple (one per word)	Single
Fixed-Size Input	❌ No	✅ Yes
Captures Frequency	❌ No	✅ Yes (Regular BoW)
Word Order	❌ Lost	❌ Lost
Semantic Meaning	❌ No	❌ Limited
Sparsity	Very High	High
OOV Problem	✅ Exists	✅ Exists
ML Ready	❌ Difficult	✅ Easy

Both techniques are foundational but have significant limitations that are addressed by more advanced methods like TF-IDF and Word2Vec.