## **📘 1. One-Hot Encoding (OHE)**

### **✅ Concept:**

* Each **word** is represented as a **binary vector** the length of the vocabulary.
* Each sentence is represented as a list of **word vectors**.
* Position of 1 in the vector represents the index of the word in the vocabulary.

🧠 Think of it like:

* Vocabulary: ['A', 'man', 'eat', 'food', 'cat', 'fish', 'and', 'are', 'friends']
* Word 'cat' → One-hot vector: [0, 0, 0, 0, 1, 0, 0, 0, 0]

## **📘 1. Bag of Words (BoW)**

### **✅ Concept:**

* Build a vocabulary of all **unique words** in the corpus.
* Each sentence is represented as a **vector**, with counts of how many times each word from the vocabulary appears.

### **✏️ Step-by-Step:**

* Corpus: 3 sentences
* Unique words: 'A', 'man', 'eat', 'food', 'cat', 'fish', 'and', 'are', 'friends'
* Vocabulary size = 9

| **Sentence** | **A** | **man** | **eat** | **food** | **cat** | **fish** | **and** | **are** | **friends** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A man eat food | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| cat eat fish | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| man and cat are friends | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

| **Feature** | **One-Hot Encoding (OHE)** | **Bag of Words (BoW)** |
| --- | --- | --- |
| **Encodes** | Each **word** individually | Each **sentence/document** |
| **Vector type** | Binary vector with a single 1 | Count vector (word frequency) |
| **Use-case level** | Word-level representation | Sentence/document-level representation |
| **Ignores word order?** | No (used per word, in order) | Yes |
| **Captures frequency?** | ❌ No | ✅ Yes |
| **Shape example** | Word → Vector (1 in position of that word) | Sentence → Vector (counts of each word) |

## 

## 

## **🧠 What is an N-gram?**

An **n-gram** is a **sequence of "n" words** from a given sentence or text.

* **Unigram (n=1):** single word
* **Bigram (n=2):** pair of consecutive words
* **Trigram (n=3):** sequence of three consecutive words
* and so on…

## **❓ Why Use N-grams?**

Natural language isn't just about words — it's about **word combinations** and **context**. N-grams help capture this.

### **✅ Benefits:**

| **Purpose** | **Why N-grams Help** |
| --- | --- |
| **Context awareness** | Single words (unigrams) miss context. N-grams capture it. |
| **Phrase recognition** | Helps in identifying common expressions like “New York” or “not good”. |
| **Text classification** | Improves models by showing not just what words are used, but how they're used together. |
| **Autocomplete/search** | Predict next word based on previous 1–2 words (e.g., Google search suggestions). |

## **📌 Real-World Analogy (Computer-Related)**

Think of a **program log**:

* A **single log line** tells you *what happened* (unigram).
* But **two or three consecutive lines** together might tell you *why it happened* (bigram/trigram).

Likewise, in text, just the word “not” or “good” means less than “not good” together.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **📘 What is TF-IDF?**

**TF-IDF** stands for:

**Term Frequency – Inverse Document Frequency**

It is a numerical **statistic used to weight words** in documents.

Unlike BoW which counts words equally, TF-IDF **penalizes common words** and **rewards rare but important words**.

## **🎯 Purpose of TF-IDF**

| **Problem with BoW** | **How TF-IDF Fixes It** |
| --- | --- |
| Common words dominate | Downweights common words using IDF |
| Doesn’t measure importance | Gives higher score to rare, relevant words |
| All words treated equally | Assigns intelligent weights |

### **💡 Example:**

* The word "the" appears in almost **every document** → not important.
* The word "fraud" appears **rarely** but may indicate spam → important.

TF-IDF helps models focus on words that really **differentiate** documents.

## **🧠 TF-IDF = TF × IDF**

### **1. TF: Term Frequency**

Measures how often a word appears in a document.

TF(t,d)=count of t in dtotal words in d\text{TF}(t, d) = \frac{\text{count of } t \text{ in } d}{\text{total words in } d}TF(t,d)=total words in dcount of t in d​

### **2. IDF: Inverse Document Frequency**

Measures how **unique or rare** a word is across all documents.

IDF(t)=log⁡(N1+df(t))\text{IDF}(t) = \log\left(\frac{N}{1 + \text{df}(t)}\right)IDF(t)=log(1+df(t)N​)

* NNN: total number of documents
* df(t)\text{df}(t)df(t): number of documents containing the word ttt

### **3. TF-IDF Score**

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

