TEXT REPRESENTATION

- Corpus: A corpus is a large collection of text used for training or testing NLP models
- Vocabulary: Vocabulary is the set of unique words present in the entire corpus.
- Document: A document is a single piece of text inside the corpus.

One Hot Encoding: One-Hot Encoding (OHE) is a way to represent categorical data (like words, labels, or categories) as binary vectors.
Example: Suppose the Vocabulary = {cat, dog, apple}

- "cat" → [1, 0, 0]
- "dog" → [0, 1, 0]
- "apple" → [0, 0, 1]

Advantages of One-Hot Encoding

- Very straightforward to implement.
- Many ML algorithms (like Logistic Regression, Naïve Bayes, Neural Networks) work well with OHE data.
- Each category has a unique representation (no ambiguity).

Disadvantages of One-Hot Encoding

1. High Dimensionality:
   - If vocabulary/categories are huge (say 50,000 words), the vector becomes very long and sparse
   - This leads to high memory usage and slower training.

2. No semantic meaning:
   - "cat" = [1,0,0], "dog" = [0,1,0] → both vectors are equally distant.
   - It does not capture similarity

Bag Of Words: Bag of Words is a text representation technique where we build a vocabulary (unique words) from the corpus and each document is represented as a vector of word counts (or frequencies).

Example: Corpus = {"The cat sat on the mat", "The dog barked at the cat"}

- Doc1 ("The cat sat on the mat"):
  [The=2, cat=1, sat=1, on=1, mat=1, dog=0, barked=0, at=0] → [2,1,1,1,1,0,0,0]
- Doc2 ("The dog barked at the cat"):
  [The=2, cat=1, sat=0, on=0, mat=0, dog=1, barked=1, at=1] → [2,1,0,0,0,1,1,1]
-

Advantages of Bag Of Words

- Simple & Intuitive – Easy to understand and implement.
- Works with Many ML Algorithms – Can be directly fed into classifiers like Naïve Bayes, Logistic Regression, or SVM.
- Good for Small Datasets

Disadvantages of Bag Of Words

1. High Dimensionality:

   - Vocabulary can be very large (thousands or millions of words).
   - Leads to sparse vectors.

2. No semantic meaning:

   - Treats all words as independent; cannot capture similarity (e.g., "good" and "excellent" are treated as unrelated).
   - It does not capture similarity

3. No Word Order :

   - "dog bites man" vs. "man bites dog" → same BoW representation, though meaning is completely different.

One-Hot Encoding → represents words individually, no frequency info, lots of redundancy.
Bag of Words → represents the whole document in one vector, includes word frequency, and is more practical for text classification tasks.

N Gram: In NLP, an N-gram is a sequence of N consecutive words from a given text. Instead of treating each word independently (like in Bag of Words), N-grams consider word order and context.

Example: Sentence: "I love natural language processing"

Unigram: ["I", "love", "natural", "language", "processing"]
Bigram: ["I love", "love natural", "natural language", "language processing"]
Trigram: ["I love natural", "love natural language", "natural language processing"]

Advantages of N-gram

1. Captures Local Context: Unlike Bag of Words, N-grams consider word order (e.g., "New York" ≠ "York New").

2. Improves Meaning Representation: Phrases like "not good" vs. "good" → Bigrams distinguish them, while BoW might miss the difference.

3.Simple and Interpretable: Easy to understand and implement for text classification, sentiment analysis, etc.

Disadvantages of N-gram

1.High Dimensionality:

- As N increases, the vocabulary grows very fast
- Example: With 10,000 unique words → possible bigrams = $10{,}000^2$ = 100M.

2.Data Sparsity:

- Many N-grams may never appear in training data, making it hard to generalize.

3.Missing Long Dependencies and Computationally Expensive:

- Even trigrams only look at 2–3 words in a row, so long-distance dependencies are missed
- Larger N-grams → more storage, more processing time.

- BoW = simple but ignores order.
- N-grams = adds local context, but at the cost of more dimensionality & sparsity.

TF-IDF Vectorizer

TF-IDF (Term Frequency – Inverse Document Frequency) is a text representation technique. It improves on BoW by reducing the weight of common words (like “the”, “is”) and increasing the weight of important words that appear rarely but carry more meaning. Used in search engines, document similarity, and NLP tasks.

How it Works: It assigns a weight to each word in a document using two factors

1.Term Frequency:

- How often a word appears in a document compared to total words.
- Formula:

IF(t, d) = (Number of times term t appears in document d) / (Total number of terms in document d)

2.Inverse Document Frequency:

$$IDF(t) = \log(N / (1 + df(t)))$$

where:
N = total number of documents
df(t) = number of documents containing term t

3. TF-IDF Score: *Final weight = TF X IDF*

TF – IDF(t,d) = TF(t,d) X IDF(t)

Example: Corpus: 2 document = {“ I love NLP”, “I love machine learning”}
Vocabulary = {I, love, NLP, machine, learning}
"love" appears in both documents → IDF is low.
"NLP" appears only in doc1 → IDF is high.
So, TF-IDF will give higher weight to "NLP" than "love".

Advantages of TFIDF

1. Better than BoW – Reduces importance of common words.

2. Simple & Efficient – Easy to compute, widely used.

3. Captures Importance – Highlights rare but meaningful words.

4. Good for Document Similarity – Helps in search engines, information retrieval.

Disadvantages of TFIDF

1. Still Ignores Word Order – Just like BoW, it doesn't consider context or sequence.

2. High Dimensionality – Vectors are still sparse when vocabulary is large.

3.No Semantic Meaning – Synonyms ("car" vs "automobile") are treated as different.

4. Not Good for Deep NLP – Modern tasks (translation, chatbots) need embeddings (Word2Vec, BERT, etc.).