NLP and Vectorization

NLP

Natural Language Processing (NLP) is a field of **Artificial Intelligence (AI)** that focuses on enabling **computers to understand, interpret, and generate human language** — like English, Hindi, or any spoken/written language.

Key Goals of NLP:

- 📖 **Understand** human language (text or speech)
- 💬 **Generate** human-like responses
- 🗃️ **Extract insights** from unstructured text
- 🧹 **Clean and preprocess** raw text data

Techniques Used in NLP:

- Rule-based methods (older)
- Machine Learning (e.g., Naive Bayes, SVM)
- Deep Learning (e.g., RNNs, LSTMs, Transformers like BERT/GPT)

Popular NLP Libraries:

- `nltk` – for basic NLP tasks
- `spaCy` – fast NLP processing
- `transformers` (by HuggingFace) – modern deep learning models
- `TextBlob` , `gensim` , `Scikit-learn` , etc.

Count Vectorization

Count Vectorization (also called **Frequency Vectorization**) is a simple and commonly used technique in Natural Language Processing (NLP) to **convert text into numerical features** that machine learning models can understand. Count Vectorization turns a collection of text documents into a

matrix of token counts — basically, it **counts how many times each word appears** in each document.

How It Works:

1. Build a **vocabulary** of all unique words in your dataset.
2. For each document, count how many times each word appears.
3. Store the result in a matrix where:
    - Rows = documents
    - Columns = words in the vocabulary
    - Values = word counts

Advantages:

- Simple and fast
- Works well with algorithms that expect numerical input (like Naive Bayes)

Limitations:

- Doesn’t capture the **meaning or context** of words
- Produces **sparse** matrices for large vocabularies
- Ignores word order (bag of words assumption)

TF-IDF (Term Frequency–Inverse Document Frequency)

TF-IDF (Term Frequency–Inverse Document Frequency) is an advanced text vectorization technique used in NLP to convert text into numerical values that reflect **how important a word is** in a document **relative to the entire collection (corpus)**.

Formula

1. TF (Term Frequency)

How often a word appears in a document.

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total terms in document } d}$$

2. IDF (Inverse Document Frequency)

How **rare** a word is across **all documents**.

$$IDF(t) = \log\left(\frac{N}{1 + df(t)}\right)$$

Where:

- N = Total number of documents
- df(t) = Number of documents containing the term t
- The "+1" avoids division by zero.

3. TF-IDF Score

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

Word2Vec

Word2Vec is a word **embedding technique** used in Natural Language Processing (NLP) to convert words into **dense vector representations** that capture **semantic meaning** — i.e., how words relate to each other in context.

The Idea Behind Word2Vec

Unlike Count Vectorizer or TF-IDF (which are based on word frequency and produce sparse vectors), Word2Vec tries to **understand the meaning** of a word based on its **context** — the words that appear around it.

How Word2Vec Works

Word2Vec comes in **two model architectures**:

Architecture	What it does
CBOW (Continuous Bag of Words)	Predicts the **current word** based on surrounding context words
Skip-gram	Predicts the **context words** from the current word

Output: Vector Representations

After training, Word2Vec gives each word a **vector of real numbers** (e.g., 100 or 300 dimensions).

Words with **similar meanings** will have **similar vectors** (i.e., they will be close in vector space).

Example:

- vector("king") - vector("man") + vector("woman") ≈ vector("queen")

Euclidean distancing

What is Euclidean Distance?

Euclidean distance is the straight-line distance between two points in Euclidean space. It's the most common way to measure the distance between two vectors or coordinates.

Formula (2D space):

If you have two points:

- $A = (x_1, y_1)$
- $B = (x_2, y_2)$

Then the Euclidean distance between them is:

$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

For n-Dimensional Vectors:

If $\mathbf{a} = (a_1, a_2, ..., a_n)$ and $\mathbf{b} = (b_1, b_2, ..., b_n)$, then:

$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$

Use Cases:

- **K-Means Clustering** (to measure closeness between a point and a cluster center)
- **K-Nearest Neighbors (KNN)**
- Measuring similarity in recommendation systems

- Image comparison and classification

Example:

Let A = (3, 4), B = (7, 1)

$$d(A, B) = \sqrt{(7-3)^2 + (1-4)^2} = \sqrt{16+9} = \sqrt{25} = 5$$

Cosine distancing

What is Cosine Distance?

Cosine distance (or cosine dissimilarity) is a metric used to measure how **different** two vectors are **in terms of direction**, rather than magnitude. It is commonly used in **text mining** and **high-dimensional data** where **Euclidean distance is not effective**.

Intuition:

- Cosine distance looks at the **angle** between two vectors, not their length.
- If two vectors **point in the same direction**, they have a **cosine distance of 0**.
- If they are **orthogonal (90°)**, the cosine distance is **1** (most dissimilar).

Formula for Cosine Similarity:

For vectors **A** and **B**:

$$\text{Cosine Similarity} = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| \cdot ||\vec{B}||}$$

Where:

- $\vec{A} \cdot \vec{B}$ is the **dot product** of the vectors
- $||\vec{A}||$ is the **magnitude (norm)** of vector A

Cosine Distance:

$$\text{Cosine Distance} = 1 - \cos(\theta)$$

Use Cases:

- **Text Similarity** (e.g., comparing documents using TF-IDF vectors)
- **Recommendation Systems**
- **Clustering** high-dimensional data (e.g., in NLP, image tagging)

Example:

Let:

- A = [1, 2, 3]
- B = [4, 5, 6]

Then:

1. **Dot Product**: 1*4+2*5+3*6=32
2. **Magnitudes**:
   $||A|| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$
   $||B|| = \sqrt{4^2 + 5^2 + 6^2} = \sqrt{77}$
3. **Cosine Similarity**:
   $\frac{32}{\sqrt{14} \cdot \sqrt{77}} \approx 0.9746$
4. **Cosine Distance**:

   1 - 0.9746 = 0.0254

This means vectors A and B are **very similar** in direction.
