# Feature Extraction and Word Embedding

### **Bag of Words (BoW)**

from sklearn.feature\_extraction.text import CountVectorizer

text\_data = ["I love NLP", "NLP is great", "I love learning NLP"]

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(text\_data)

print(vectorizer.get\_feature\_names\_out())  # Output: ['great', 'learning', 'love', 'nlp']

print(X.toarray())  # Output: [[0 0 1 1] [1 0 0 1] [0 1 1 1]]

### **Term Frequency-Inverse Document Frequency (TF-IDF)**

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf\_vectorizer = TfidfVectorizer()

X\_tfidf = tfidf\_vectorizer.fit\_transform(text\_data)

print(tfidf\_vectorizer.get\_feature\_names\_out())  # Output: ['great', 'learning', 'love', 'nlp']

print(X\_tfidf.toarray())  # Output: [[0.         0.         0.63009934 0.77651462] [0.77651462 0.         0.         0.63009934] [0.         0.79596054 0.45291081 0.4012868 ]]

**Word Embedding**

import gensim.downloader as api

# Load pre-trained Word2Vec model

model = api.load('word2vec-google-news-300')

# Find similar words

similar\_words = model.most\_similar('king')

print(similar\_words)  # Output: [('queen', 0.7261044979095459), ('prince', 0.6777598857879639), ...]