Python

Training Module

1. Bag of Words (BoW)

Definition:

The Bag of Words model is one of the simplest and most intuitive techniques for converting text into a numerical format that can be used by machine learning models. It represents text data as a collection of words, disregarding grammar and word order but keeping track of word frequency.

How BoW Works:

- Step 1: Tokenization: Text is broken down into individual words or tokens. For example, the sentence "The cat sat on the mat" is split into the tokens: ["The", "cat", "sat", "on", "the", "mat"].
- Step 2: Vocabulary Creation: A vocabulary (a list of all unique words) is created from the entire corpus (collection of documents). For the sentence above, the vocabulary would be: ["The", "cat", "sat", "on", "mat"].
- Step 3: Representation as a Vector: Each document is represented as a vector, where each word in the vocabulary corresponds to a specific feature (dimension), and the value represents the frequency of that word in the document. For example, for the sentence "The cat sat on the mat", the vector representation could be:

Word	The	cat	sat	on	mat
Count	2	1	1	1	1

- Step 4: Vectorization: Each document is now a fixed-length vector. A document with the same vocabulary but different word frequencies would have a different vector.

Pros and Cons of BoW:

- Advantages:
    - Simplicity: Easy to understand and implement.
    - Straightforward feature extraction for text classification tasks.
- Disadvantages:
    - Sparsity: The representation can be very sparse (many zeros), especially for large vocabularies.
    - Ignores context: Word order and grammar are lost.
    - High dimensionality: The vocabulary grows large with the size of the corpus, leading to high-dimensional vectors.

Use Cases:

- Sentiment analysis
- Document classification
- Spam detection

---

2. TF-IDF (Term Frequency-Inverse Document Frequency)

Definition:

TF-IDF is a statistical measure used to evaluate how important a word is to a document within a collection of documents (corpus). It adjusts the frequency of words based on how common or rare they are across the entire corpus, giving more weight to words that are unique to specific documents.

Formula:

TF-IDF is the product of two components:

- Term Frequency (TF): Measures how frequently a word appears in a document.

TF(t,d)=Number of times word t appears in document dTotal number of words in docum ent d\text{TF}(t, d) = \frac{\text{Number of times word } t \text{ appears in document } d}{\text{Total number of words in document } d}TF(t,d)=Total number of words in document dNumber of times word t appears in docu ment d

- Inverse Document Frequency (IDF): Measures how important a word is across the entire corpus. Words that appear in many documents are less informative.

  IDF(t,D)=log⁡(Total number of documents in the corpus Number of documents containi ng word t)\text{IDF}(t, D) = \log \left(\frac{\text{Total number of documents in the corpus }}{\text{Number of documents containing word } t} \right)IDF(t,D)=log(Number of documents containing word tTotal number of documents in the corpus)

- TF-IDF:

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

Example:

Consider a document corpus with three documents:

- Document 1: "The cat sat on the mat."
- Document 2: "The dog sat on the mat."
- Document 3: "The dog ran away."

For the word "dog":

- TF for Document 2: 1 (because "dog" appears once in Document 2)
- IDF for "dog" across the three documents: log(3/2) = 0.176
- TF-IDF: TF × IDF = 1 × 0.176 = 0.176.

Pros and Cons of TF-IDF:

- Advantages:
    - More informative than BoW because it accounts for word importance.
    - Helps to reduce the weight of common words like "the", "is", "on", etc., which are not useful for classification.
- Disadvantages:
    - Still ignores context and semantics of words.
    - High dimensionality like BoW.

Use Cases:

- Information retrieval (search engines)
- Document similarity
- Text classification

---

3. Word Embeddings

Definition:

Word embeddings are a type of word representation that allows words to be represented as vectors in a continuous vector space. Unlike BoW and TF-IDF, word embeddings capture semantic relationships between words based on their meanings and contexts, leading to dense vectors with more useful information.

How Word Embeddings Work:

- Dense Vectors: Each word is mapped to a low-dimensional vector (e.g., 50, 100, or 300 dimensions), where similar words are represented by vectors that are close together in the vector space.
- Context-based Learning: Word embeddings are learned from large corpora using models like Word2Vec, GloVe, or FastText. These models learn the relationships between words by analyzing the context in which words appear. For example:

- "king" and "queen" will have similar embeddings because they appear in similar contexts.
- "king" and "man" will have a closer relationship than "king" and "car".

Popular Word Embedding Models:

1. Word2Vec: A shallow neural network-based model developed by Google that uses either the Skip-Gram or CBOW (Continuous Bag of Words) approach to predict a word's context or predict context words from a target word.
    - Skip-Gram: Given a target word, predict its surrounding context.
    - CBOW: Given the context, predict the target word.
2. GloVe (Global Vectors for Word Representation): Developed by Stanford, GloVe uses global statistical information to create embeddings that represent word similarity.
3. FastText: Developed by Facebook, it improves on Word2Vec by considering subword information, which helps in dealing with rare and out-of-vocabulary words.

Example of Word Embeddings:

In Word2Vec, the word "king" might be represented as a vector like:

- king: [0.2, -0.1, 0.4, ..., 0.8]

This vector will be close to the vectors of semantically similar words, like:

- queen: [0.3, -0.2, 0.4, ..., 0.75]
- man: [0.1, -0.3, 0.5, ..., 0.7]

Pros and Cons of Word Embeddings:

- Advantages:
    - Captures semantic meaning and relationships between words.

    - Dense representation (fewer dimensions compared to BoW and TF-IDF).
    - Can handle synonyms and related words better.
- Disadvantages:
    - Requires a large corpus for training.
    - Not always interpretable, as embeddings are abstract numerical representations.

Use Cases:

- Semantic similarity tasks (e.g., finding similar words or sentences).
- Machine translation.
- Sentiment analysis.
- Named entity recognition (NER).

Conclusion:

- BoW and TF-IDF are traditional methods that work well for many NLP tasks, especially when dealing with structured and labeled data. However, they are limited by their inability to capture semantic meaning and context.
- Word embeddings, on the other hand, provide a more advanced and effective approach by capturing the semantic relationships between words, making them the preferred choice for more complex NLP tasks, such as machine translation, named entity recognition, and sentiment analysis.