
IRT
SAINT
EXUPÉRY

Text vectorization

Master ValDom – 10/01/2025

Summary

Traditional Vectorization Techniques

Bag of Words (BoW)

Bag of Words is a simple and widely-used text representation technique. It represents a text (such as a sentence or document) as a set of words (a "bag"), ignoring grammar, word order, and context.

Key Features

1. **Vocabulary Creation**: Extracts all unique words from the text corpus to form a vocabulary.
2. **Vector Representation**: Each document is represented as a vector where:
    - Each dimension corresponds to a word in the vocabulary.
    - Values indicate word frequency (or binary presence).

Example

Corpus: ["I like NLP", "I love NLP and AI"]

Vocabulary: [I, like, NLP, love, and, AI]

Vectors: Doc 1: [1, 1, 1, 0, 0, 0]
Doc 2: [1, 0, 1, 1, 1, 1]

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection (or corpus) of documents.

Key Features

1. **Term Frequency (TF):** Measures how often a term appears in a document. $$\mathrm{TF}(t) = \frac{\text{Count of term } t \text{ in the document}}{\text{Total terms in the document}}$$
2. **Inverse Document Frequency (IDF)**: Reduces the weight of common words appearing in many documents. $$\mathrm{IDF}(t) = \log\left(\frac{\text{Number of documents}}{\text{Number of documents containing } t}\right)$$
3. **TF-IDF Weight**: Final weight of a term in a document is the product $$\text{TF-IDF}(t) = \mathrm{TF}(t) \times \mathrm{IDF}(t)$$

Example

Corpus: ["I like NLP", "I love NLP and AI"]

TF ('NLP', Doc 1): 1/3
IDF ('NLP'): log(2/2) = log(1) = 0
TF-IDF ('NLP', Doc 1): 1/3×0=0

N-Grams

n-Grams are contiguous sequences of n words (or characters) from a given text.
An n-gram model captures word sequences and their probabilities, providing insight into the structure and context of text.

Key Features

1. What is n? n=1: Unigram (single words).
   n=2: Bigram (pairs of words).
   n=3: Trigram (triplets of words).
   Larger n: Captures longer context but increases sparsity.
2. **How it works:** Extract all n-word sequences from text.

Example

Text: "I love NLP models."
Bigrams: ["I love", "love NLP", "NLP models"].

Comparaison

	BoW	TF-IDF	N-Grams
Advantages	• Easy to understand and implement. • Effective for small datasets or simple models.	• Highlights **important terms** specific to a document. • Reduces the impact of **common words** like "the" or "is."	• Adds **context** compared to unigrams. • Flexible for text analysis tasks.
Limitations	• **No context**: Ignores word order and semantics. • **High dimensionality**: Vocabulary size grows with corpus size. • **Sparse representation**: Many zeros in the vector.	• **No context**: Similar to Bag of Words, it ignores word order and semantics. • **High dimensionality**: Large vocabularies result in sparse vectors.	• **Dimensionality grows**: More unique n-grams with higher n. • **Data sparsity**: Larger n values need more data for meaningful results. • **No long-range dependencies**: Limited context beyond n.

Word Embeddings

Word2Vec

Word2Vec is a neural network-based model that learns vector representations (embeddings) of words from a text corpus. It maps words to a continuous vector space where similar words are closer together based on their context.

Two Training Architectures: CBOW (Continuous Bag of Words) or Skip-gram.

Word Representations:

- Each word is represented as a fixed-size vector (e.g., 100-300 dimensions).
- Captures **semantic similarity** and **relationships** (e.g., "king" - "man" + "woman" ≈ "queen").

Continuous Bag of Words (CBOW)

CBOW is a training architecture in Word2Vec that predicts a target word based on the context words surrounding it. It uses a shallow neural network to generate word embeddings.

Key Features

- **Input Representation**: One-hot encoding of context words.
- **Hidden Layer**: Projects words into a lower-dimensional vector space.
- **Output Layer**: Predicts the target word using a softmax function.

How CBOW Works

Input:

Context words surrounding a target word.

Example: "I __ NLP models" → Input = ["I", "NLP"]

Output:

Predicts the **missing word** ("love" in this case).

Training:

- The model learns embeddings by maximizing the probability of the target word given its context.
- Objective function:

$$\max \prod_{w \in \text{Vocabulary}} P(w|\text{context})$$

Skip-gram

Skip-gram is a training architecture in Word2Vec that predicts context words given a target word.
It uses a shallow neural network to generate word embeddings.

Key Features

- **Input Representation**: One-hot encoding of the target word.
- **Hidden Layer**: Projects the target word into a lower-dimensional vector space.
- **Output Layer**: Predicts multiple context words using a softmax function.

How Skip-gram Works

Input:

A single target word.

Example: Input = "love."

Output:

Predicts surrounding **context words** (e.g., "I" and "NLP").

Training:

- The model learns embeddings by maximizing the probability of context words given the target word.
- Objective function: $\max \prod_{c \in \text{Context}} P(c|\text{target})$

Word2Vec – Architecture Training Choice

Criteria	CBOW	Skip-gram
Objective	Predict a target word from its surrounding context words.	Predict context words from a target word.
Efficiency	Faster to train because it aggregates multiple contexts.	Slower as it generates multiple predictions per target word.
Performance on Rare Words	Less effective, as it gives more importance to frequent words.	Better at learning representations for rare words.
Data Size Requirement	Requires a larger corpus for good performance.	Works well even with a smaller corpus.

GloVe (Global Vectors)

GloVe is a word embedding model that learns vector representations by analyzing the global word co-occurrence statistics from a text corpus.
Unlike Word2Vec, GloVe focuses on both local and global context for richer semantic embeddings.

Word Representations:

- Each word is represented as a fixed-size vector (e.g., 100-300 dimensions).
- Captures **semantic similarity** and **relationships** (e.g., "king" - "man" + "woman" ≈ "queen").

Global Context:

Captures word meaning by considering overall word relationships across the corpus.

How GloVe Works ?

Word Co-occurrence Matrix:

Builds a matrix where each cell represents how often two words co-occur in the corpus.

Optimization Objective:

Learns word embeddings by factorizing the co-occurrence matrix into lower-dimensional vectors.

Objective function minimizes the reconstruction error:

$$J = \sum_{i,j} f(X_{ij})(\vec{w}_i^\top \vec{w}_j + b_i + b_j - \log(X_{ij}))^2$$

X_{ij}: Co-occurrence count between words i and j.

$f(X_{ij})$: Weighting function to control the influence of rare vs. frequent pairs.

Contextual Representations

Contextual Techniques

Overview:

Embeddings based on Transformers (e.g. BERT).
Each word’s representation depends on its context.

Pretrained Models:

Use large corpora for training, enabling rich contextual understanding.

Comparison Example:

Sentence: "The bank is on the river."
Word2Vec: One vector for "bank."
BERT: Different vectors for "bank" depending on context.

How It Works?

Covered in CM5 Transformers

Questions