Vectorization in Natural Language Processing: From Encoding to Contextual Representations

Abstract

Natural Language Processing (NLP) requires the transformation of raw textual data into numerical representations that machine learning algorithms can process effectively. This paper presents a comprehensive examination of various text vectorization techniques, ranging from traditional statistical methods to advanced neural network-based approaches. We analyze the evolution from simple frequency-based methods like Bag of Words to sophisticated contextual embeddings using neural architectures. Each technique is evaluated based on its methodology, applications, advantages, and limitations, providing insights into their appropriate use cases in modern NLP applications.

Introduction

Raw text cannot be directly processed by machine learning models, which require numerical input. The task of vectorization bridges this gap by transforming text into numeric vectors while preserving linguistic meaning. Over time, techniques have evolved from simple categorical encodings to sophisticated contextual embeddings, significantly improving the performance of NLP applications such as sentiment analysis, information retrieval, and machine translation. This paper surveys the major methods used for text vectorization, their applications, and their limitations.

I] Foundational Encoding Techniques

I.1] Label Encoding

Label encoding represents one of the most fundamental approaches to converting categorical text data into numerical format. This technique assigns unique integer values to distinct categories, creating a direct mapping between textual labels and numerical representations. Categorical variables in Python can be transformed into numerical labels using the label encoding technique. It gives each category in a variable a distinct numerical value, enabling machine learning algorithms to interpret and analyse the data effectively.

Process:

1. **Vocabulary Creation:** Compilation of all unique categorical values present in the dataset
2. **Unique Integer Assignment:** Systematic allocation of distinct integer values to each category
3. **Numerical Representation:** Replacement of original categorical values with corresponding integers

Application:

Label encoding finds primary application in:

1. **Text Categorization**: Product classification and sentiment analysis

2. **Feature Engineering**: Creation of derived features from existing categorical data
3. **Recommendation Systems**: Numerical representation of categorical user preferences

Limitation:

Despite its simplicity, label encoding presents several critical limitations:

1. **Implied Ordinal Relationships**: Introduces artificial ordering among categories
2. **Inadequacy for Nominal Data**: Unsuitable for non-ordinal categorical variables
3. **Limited Semantic Representation**: Fails to capture meaningful word relationships
4. **High Cardinality Issues**: Computational challenges with numerous unique categories
5. **Semantic Information Loss**: Discards contextual meaning inherent in text
6. **Out-of-Vocabulary Problem**: Cannot handle new categories without retraining
7. **Inconsistent Dimensionality**: Lacks uniform input dimensions across datasets

I.2] One-Hot Encoding

One-hot encoding addresses several limitations of label encoding by creating binary vector representations where each category corresponds to a unique binary vector with exactly one element set to 1.

Advantages:

1. Eliminates artificial ordinal relationships
2. Provides consistent binary representation
3. Maintains categorical independence

Limitations:

1. Dimensional explosion with high-cardinality variables
2. Memory inefficiency due to sparse matrices
3. Computational overhead from increased feature dimensions

II] Frequency-Based Vectorization Methods

II.1] Bag Of Words

Beginner level text processing and featurisation method. It is also an information retrieval algorithm that converts text inputs into vectored numeric output on which ml algorithms can be implemented. It focuses on unstructured assortment of all the known words in a text document defined according to frequency.

Process

BoW creates a vocabulary of all unique words across the corpus and represents each document as a vector where each dimension corresponds to a word's frequency in that document.

Applications

1. Information retrieval systems
2. Document classification
3. Spam detection
4. Basic sentiment analysis

Limitations

1. **Sparse Representation**: Majority of vector elements are zero
2. **Semantic Inadequacy**: Fails to capture word meaning and context
3. **Word Order Insensitivity**: Ignores sequential relationships
4. **Polysemous Word Handling**: Cannot differentiate multiple word meanings

II.2] Bag Of N-Grams

N-gram models extend BoW by considering sequences of n consecutive words, partially addressing word order limitations.

Process

Instead of individual words, n-grams capture phrase-level information by creating features from word sequences of length n.

Advantages

1. Better handling of word correlations
2. Improved representation of polysemous words
3. Preservation of local word order

Limitations

1. **Increased Dimensionality**: Exponential growth in feature space
2. **Enhanced Sparsity**: More pronounced sparse representations
3. **Out-of-Vocabulary Persistence**: Still faces unseen n-gram challenges

II.3] TF-IDF

Statistical method used in natural language processing and information retrieval sectors to evaluate the significance of a word in a document in relation to a larger collection of documents

Mathematical Calculation

Term Frequency:

Measures how often a word appears in a document. A higher frequency suggests greater importance. If a term appears frequently in a document, it is likely relevant to the document's content.

$$\text{Tf (t,d)} = \frac{Number\ of\ times\ t\ appears\ in\ document\ d}{total\ terms\ in\ document\ d}$$

Inverse Document Frequency:

Reduces the weight of common words across multiple documents while increasing the weight of rare words. If a term appears in fewer documents, it is more likely to be meaningful and specific. That is it tries to eliminate the stop words to filter only meaningful content without it loosing the essence of the document

$$\text{IDF}(t,D) = \log\left(\frac{Total\ number\ of\ documents\ in\ corpus\ d}{Number\ of\ documents\ containing\ t}\right)$$

This balance allows TF-IDF to highlight terms that are both frequent within a specific document and distinctive across the text document, making it a useful tool for tasks like search ranking, text classification and keyword extraction.

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

Process

1. **Term Frequency Calculation**: Measure word occurrence within documents
2. **Inverse Document Frequency Computation**: Assess word rarity across corpus
3. **TF-IDF Score Generation**: Combine both metrics for final representation

Applications

1. Document Similarity and Clustering**: Grouping related texts**
2. Text Classification**: Pattern identification for categorization**
3. Keyword Extraction**: Automatic identification of important terms**
4. Recommendation Systems**: Content-based filtering through textual descriptions**

Advantages

1. Reduces impact of common words (stop words)
2. Highlights document-specific important terms
3. Computationally efficient and interpretable
4. Effective for many traditional NLP tasks

Limitations

1. **Sparsity and Dimensionality**: Increased feature space complexity
2. **Semantic Blindness**: Cannot capture word relationships
3. **Out-of-Vocabulary Problem**: Struggles with unseen terms
4. **Context Insensitivity**: Ignores word position and context

III] Text Representation Using Word Embeddings

Neural word embeddings address fundamental limitations of traditional methods by creating dense, continuous vector representations that capture semantic relationships between words. These techniques offer several key advantages:

- **Semantic Similarity**: Similar words receive similar vector representations
- **Dimensionality Reduction**: Dense vectors reduce sparsity issues
- **Relationship Modeling**: Capture complex linguistic relationships
- **Contextual Understanding**: Better handling of polysemous words

A. Continuous Bag of Words and Skip Gram

Word2Vec: Continuous Bag of Words:

Predicts the target word from the context. It is a neural network model that is used to generate the word embeddings by predicting a target word based on its surrounding context words within a specified window

Working:

1. **Initial Encoding**: Words encoded as one-hot vectors
2. **Context Window Definition**: Specified window size for context consideration
3. **Context Vector Creation**: Average probability vectors of context words
4. **Target Prediction**: Predict central word from averaged context vector
5. **Parameter Optimization**: Backpropagation refines word embeddings
6. **Embedding Extraction**: Final embeddings extracted from learned weight matrix

Word2Vec: Skip Gram:

Predicts the context words based on the target. It utilises a shallow neural network with an input layer, a single hidden layer that shows the word embeddings and an output layer

Working:

1. **Target Word Input**: Single word as input to network
2. **Context Prediction**: Predict surrounding words within window
3. **Probability Maximization**: Maximize likelihood of correct context prediction
4. **Weight Optimization**: Adjust embedding weights through gradient descent
5. **Efficiency Techniques**: Employ negative sampling and hierarchical softmax

GloVe (Global Vectors):

Global Vectors derives a semantic relation between the words using word to word co-occurrence matrix (gives us the relational occurrences of 2 words in a corpus). Its an unsupervised learning algorithm that also created word embeddings. It combines 2 major approaches to effectively generate the feature vectors:

1. LSA: latent Semantic Analysis – that uses global statistical information
2. Context-based Models – like Word2Vec that focus on local word context

It constructs a word co-occurrence matrix where each element reflects how often a pair of words appears together within a given context window. It then optimizes the word vectors such that the dot product between any two word vectors approximates the pointwise mutual information (PMI) of the corresponding word pair. This optimization allows GloVe to produce embeddings that effectively encode both syntactic and semantic relationships across the vocabulary.

PMI – pointwise mutual information is the measure to calculate the association in 2 events or here words based on their join probability compared to their individual one.

$$\text{PMI}(x,y) = \log\left(\frac{P(x,y)}{P(x).P(y)}\right)$$

Interpretation of PMI :

1. High PMI: Strong word association beyond chance
2. Zero PMI: Random co-occurrence
3. Negative PMI: Less frequent co-occurrence than expected

Working:

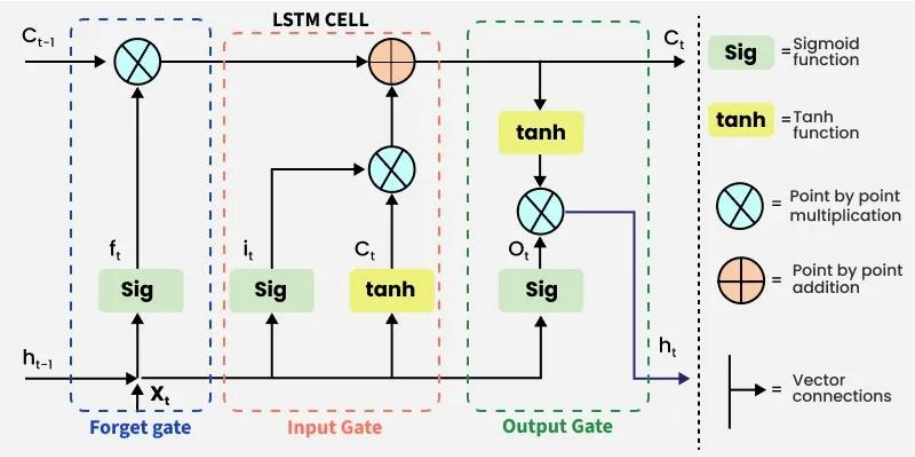
1. **Co-occurrence Matrix Construction:** Build word-word co-occurrence statistics
2. **Pointwise Mutual Information (PMI):** Calculate word association measures
3. **Vector Optimization:** Minimize difference between dot products and PMI scores
4. **Embedding Generation:** Extract dense vectors capturing semantic relationships

IV] Long Short-Term Memory (LSTM)

RNN Limitations:

Traditional Recurrent Neural Networks (RNNs) struggle with long-term dependencies due to vanishing gradient problems. LSTM networks address these limitations through sophisticated gating mechanisms that control information flow.

LSTM Architecture



Forget Gate

Controls information removal from cell state:

$$f_t = \sigma(Wf \cdot [h_{t-1}, x_t] + bf)$$

where:

- Wf: forget gate weight matrix
- $[h_{t-1}, x_t]$: concatenated previous hidden state and current input
- bf: forget gate bias
- σ: sigmoid activation function

Input Gate

Regulates new information addition to cell state:

$i_t = \sigma(Wi \cdot [h_{t-1}, x_t] + bi)$

$\tilde{C}_t = \tanh(Wc \cdot [h_{t-1}, x_t] + bc)$

Cell State Update

Combines forget and input operations:

$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$

5.2.4 Output Gate

Determines hidden state output:

$o_t = \sigma(Wo \cdot [h_{t-1}, x_t] + bo)$

$h_t = o_t \odot \tanh(C_t)$

LSTM Advantages

1. **Long-term Dependency Capture**: Maintains information across extended sequences
2. **Selective Memory**: Intelligent retention and forgetting of information
3. **Gradient Stability**: Mitigates vanishing gradient problems
4. **Versatile Applications**: Effective for translation, speech recognition, and time series

V] Use Case Recommendations

- **Simple Classification**: Label encoding, One-hot encoding
- **Information Retrieval**: TF-IDF, Bag of Words
- **Semantic Similarity**: Word2Vec, GloVe
- **Sequential Modeling**: LSTM, advanced RNN variants
- **Large-scale Applications**: Pre-trained embeddings (Word2Vec, GloVe)

Conclusion

The choice of vectorization technique should be guided by specific application requirements, computational constraints, and data characteristics. Simple methods suffice for basic classification tasks, while complex semantic understanding requires advanced neural approaches.