УДК 004.051

FROM WORDS TO VECTORS: TEXT VECTORIZATION TECHNIQUES IN NATURAL LANGUAGE PROCESSING

Krez K.S.

*Belarusian State University of Informatics and Radioelectronics,
Minsk, Republic of Belarus*

Research Supervisor: Shneiderov E.N. – Cand. of Sci., associate professor, associate professor of the department of PICS

Annotation. This paper discusses the process of text vectorization, which is a key step in Natural Language Processing (NLP). The main vectorization methods are described, including One-Hot Encoding, Bag of Words, TF-IDF and Word Embeddings. The advantages and disadvantages of each method are analyzed, as well as their application in various NLP tasks. Text vectorization converts textual data into numerical vectors that can be processed by machine learning algorithms. This is necessary because computers cannot directly interpret text.

Keywords: encoding, vectorization, text, algorithm, analysis

Natural Language Processing (hereinafter – NLP) is a branch of artificial intelligence dealing with the development of algorithms for analyzing and generating human speech. Since computers are unable to interpret text in its original form, the data must be converted into a numerical format suitable for machine processing. This process of converting text into numerical vectors, known as vectorization, is a fundamental step in modern NLP systems [1].

Text vectorization is the process of transforming linguistic units (words, phrases or documents) into numerical vectors [2] structured sets of numerical values encoding semantic and syntactic features of text data for their subsequent processing by machine learning algorithms.

Since classical machine learning algorithms, including neural networks, work only with numbers, text needs to be converted into numerical form for processing.

In the process of vectorization, text data are transformed into multidimensional numerical representations encoding their linguistic characteristics - from superficial syntactic structures to deep semantic relationships. The resulting vector spaces serve as input to ML models that solve a wide range of NLP tasks, from text classification to natural language generation.

Word vectorization works analogously to a cataloging system in a library. Suppose a librarian has the goal of quickly selecting literature in a given area. Of course, it is possible to analyze each publication manually, but this approach would be extremely time inefficient. Therefore, he creates a system that systematizes the topics for him:

– analyzes the content of each text;

– assigns numerical labels to the texts that reflect the topics.

Text vectorization works in a similar way: each word is encoded with a numeric vector that reflects its meaning and features.

For example, if you represent subject labels as numbers, books on history will be highly weighted by the tag «history» and books on programming will be highly weighted by the tag «programming». Similarly, in the vector representation, words are described not by categories but by multidimensional numerical values that capture their semantic properties.

Main vectorization methods:

– One-Hot Encoding;

– Bag of Words;

– TF-IDF (Term Frequency-Inverse Document Frequency);

– Word Embeddings.

Direct encoding is the most basic approach to transforming text data. In this technology, each word in the dictionary is assigned a unique identifier (index). The words are then mapped in a vector space, where each vector has zero values of all elements except one, whose position corresponds to the index of the represented word.

For an example of direct encoding, consider a dictionary of three lexemes: [«giraffe», «elephant», «animal»]. Accordingly, the vector representation of the word «giraffe» will be [1, 0, 0], of the word «elephant» - [0, 1, 0], and of the word «animal» - [0, 0, 1].

The bag-of-words model represents a text as a multiset of its constituent tokens. The order of these lexemes is not taken into account, and the significant characteristic is their frequency. For each text document, a frequency vector is formed, where each component corresponds to the number of occurrences of a certain word from the dictionary. This method, being simple to implement, allows taking into account the statistics of word occurrence.

As an example of applying the bag-of-words model, consider a corpus of two sentences: «The elephant eats watermelon» and ‘The elephant swims in the river’. The lexicon of this corpus is: [«elephant», «eats», «watermelon», «bathes», «in», «river»]. The vector representation of this text in the «bag of words» model would be as follows: [2, 1, 1, 1, 1, 1, 1, 1, 1], where each number corresponds to the frequency of occurrence of a word from the dictionary.

TF-IDF is an improved approach compared to bag-of-words. Its peculiarity is that when determining the importance of a word, not only its frequency in a given document (TF), but also its rarity in the general corpus of texts (IDF) is taken into account. This approach reduces the influence of common but unimportant words (e.g., prepositions and conjunctions) and, on the contrary, increases the importance of rare and important terms.

This means that the word «elephant», although it may occur frequently in an individual document (high TF), would not be considered as important for distinguishing between documents as the word «watermelon». Since «watermelon» is rare, its high IDF value will outweigh its potentially low TF in those documents where it is present, making it more significant in terms of TF-IDF.

This method estimates the importance of words by considering their distribution throughout the document collection. However, it still ignores the order of words and their semantic relationship with each other. In addition, the accuracy of TF-IDF strongly depends on the size of the analyzed set of texts.

Embeddings represent the most recent and frequently used method in the field of NLP. They encode words as small vectors, with the semantic proximity between words reflected in the proximity of their vector representations. Embeddings are trained on extensive textual data using neural networks to obtain these representations. This approach demonstrates a number of significant advantages over alternative techniques:

– embeddings allow to take into account semantic relations between lexemes - the vector representation of the word «cat» will be close to the vector of the word «cat»;

– embedding vectors are characterized by low dimensionality, which is a significant advantage compared to high-dimensional vectors of direct encoding;

– over vector representations of words in embeddings it is possible to perform algebraic operations reflecting semantic analogies, for example: the vector «queen» minus the vector «woman» plus the vector «man» is approximately equal to the vector «king».

However, unlike other methods, embeddings are difficult to implement and computationally expensive to train a neural network. Their accuracy also strongly depends on the quality and size of training data.

Table 1 – Text vectorization methods

Characterization	One-Hot Encoding	Bag of Words (BoW)	TF-IDF	Word Embeddings
Presentation	A vector with 0 and 1, where 1 indicates the presence of a word in the text	Vector, where each value is the number of occurrences of the word in the text	Vector, where each value is the weighted frequency of a word given its rarity in the corpus	A dense vector representing the semantics of a word
Dimensionality	Equal to the size of the dictionary	Equal to the size of the dictionary	Equal to the size of the dictionary	Set manually (usually 100-300)
Application	Categorical data, simple NLP tasks	Simple NLP tasks, information retrieval	Information retrieval, text classification	Complex NLP tasks, machine translation, tone analysis
Example	[0, 1, 0, 0, 0, 0, 1] (if dictionary = [cat, dog, mouse, elephant, lion])	[1, 2, 0, 0, 0, 1] (if dictionary = [cat, dog, mouse, elephant, lion])	[0.1, 0.5, 0, 0, 0, 0.2] (weighted frequencies)	[0.2, -0.5, 0.8, ...] (vector of semantic representation of a word)

Thanks to embeddings, which are used in almost all advanced NLP models and large language models (LLMs) for text encoding, these models are able to understand the semantic content of text and generate logical and coherent sentences.

Among the various embedding technologies, Word2Vec and GloVe stand out for their popularity.

Word2Vec technology is based on two different architectures, each of which analyzes text data to extract meaning. However, their approaches to analysis differ:

– the Continuous Bag-of-Words (CBOW) architecture works on the principle of predicting the center word based on the surrounding words (context);

– in contrast, the skip-gram architecture uses the target word to predict its context, i.e., the words that surround it.

GloVe is a method that assumes that semantic relations between words can be determined by analyzing a co-occurrence matrix. This matrix reflects how often different words appear in the same context.

The training process of the GloVe model involves the factorization of this matrix, i.e., its decomposition into its components. The result is vector representations of words that contain information about how often these words occur together.

Converting text into numerical vectors, known as vectorization, is an integral part of NLP, allowing text to be analyzed using machine learning. Different vectorization methods offer different capabilities and are applied depending on the task at hand.

The choice of method, whether simple One-Hot Encoding or «bag-of-words» for basic classification tasks, or more advanced Word Embeddings for machine translation and tone analysis, is determined by the complexity of the task and the resources available.

List of references

1. Lee A, Auli M, Ranzato MA. Discriminative reranking for neural machine translation. In: ACL-IJCNLP 2021 – 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 2021. P. 7250–7264.

2. Bengfort B., Bilbro R., Okheda T. Prikladnoy analiz tekstovykh dannykh na Python. Mashinnoe obuchenie i sozdanie prilozheniy obrabotki estestvennogo yazyka [Applied analysis of text data in Python. Machine learning and building natural language processing applications]. Saint Petersburg: Piter, 2019, 368 p.