BERT, short for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, is a language model based on the transformer architecture and excels in dense embedding and retrieval models. Its bidirectional nature reads text in both directions, which makes it different from earlier models that processed text in only one direction. Because of this, BERT achieves state-of-the-art results in various NLP tasks. Unlike traditional sequential natural language processing methods that move from left to right of a sentence or vice versa, BERT grasps word context by analyzing the entire word sequence simultaneously, thereby generating dense embeddings.

Foundation for choosing BERT for creation of vector embeddings:

1. Pre-training Process:

   - BERT's pre-training consists of two main tasks:

     - Masked Language Modeling (MLM):

       - Randomly masks portions of input tokens.

       - Trains the model to predict these masked tokens by considering context from both sides (left and right).

       - Enhances understanding of unidirectional context.

     - Next Sentence Prediction (NSP):

       - Predicts whether one sentence logically follows another.

       - Crucial for understanding relationships between sentences, aiding in paragraph structure comprehension.

2. Self-Attention Mechanism:

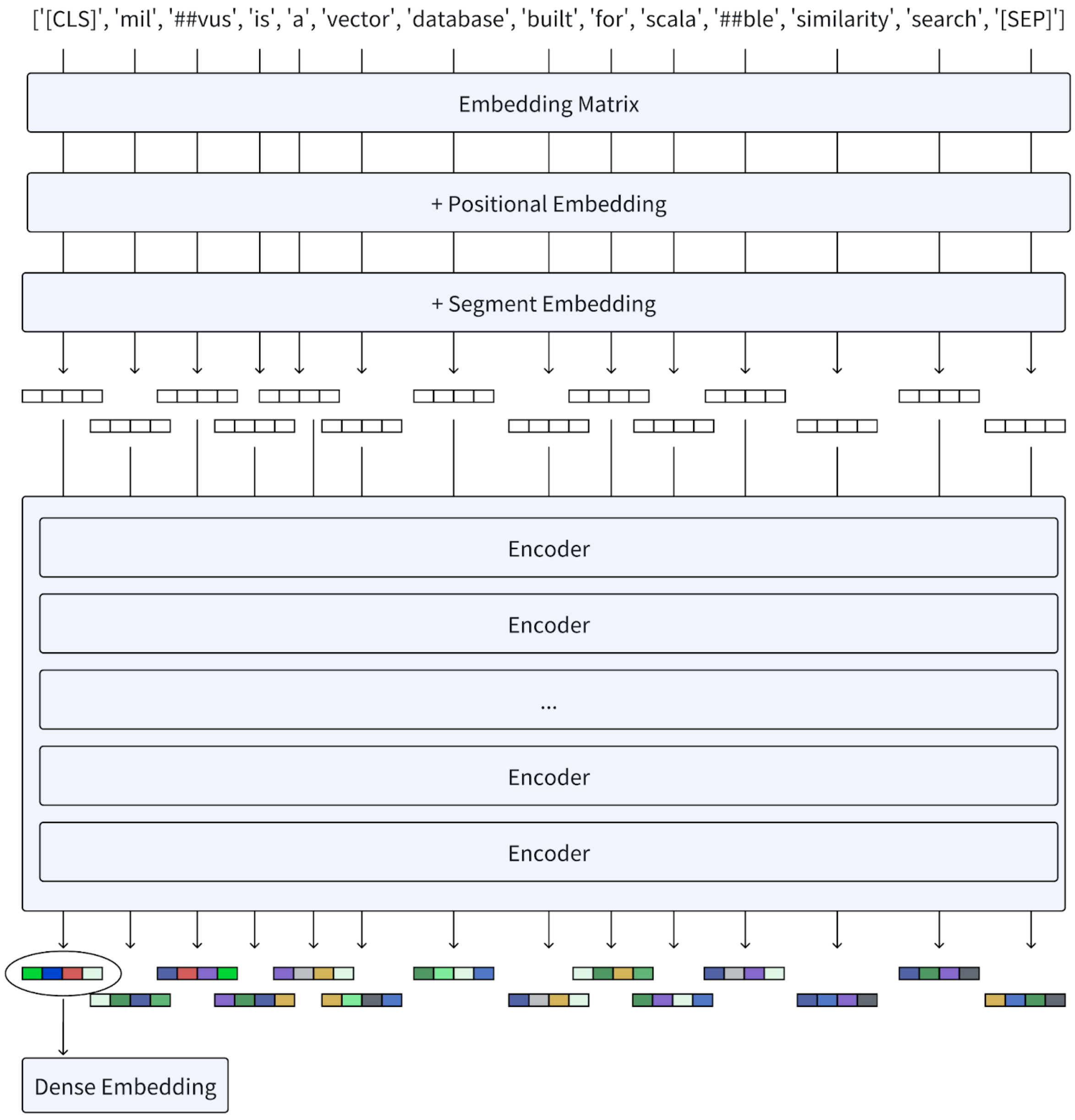
   - Each encoder layer employs self-attention to evaluate the significance of other words when interpreting a specific word.

   - Enables a nuanced understanding of word meanings based on context.

3. Positional Encoding:

   - Provides a method to understand the order of words.

- Integrates sequence information into the otherwise position-agnostic self-attention process.



['[CLS]', 'mil', '##vus', 'is', 'a', 'vector', 'database', 'built', 'for', 'scala', '##ble', 'similarity', 'search', '[SEP]']
Embedding Matrix
+ Positional Embedding
+ Segment Embedding
Encoder
Encoder
...
Encoder
Encoder
Dense Embedding

From tokens to BERT dense embeddings

When a query is input into BERT, the following steps occur:

A. Tokenization:

- The text is tokenized into a sequence of word pieces.

- The [CLS] token is added at the beginning for sentence-level tasks.

- [SEP] tokens are used to separate sentences and indicate their end.

B. Embedding:

- Each token is transformed into vectors via an embedding matrix, akin to Word2Vec.
- Positional embeddings are included to maintain the order of words.
- Segment embeddings differentiate between various sentences.

C. Encoders:

- Vectors pass through multiple encoder layers.
- Each layer contains self-attention mechanisms and feed-forward neural networks.
- This process iteratively refines each token's representation based on the context of other tokens in the sequence.

D. Output:

- The final layer produces a sequence of embeddings.
- For sentence-level tasks, the [CLS] token's embedding serves as the overall representation.
- Individual token embeddings can be used for detailed tasks or combined through methods like max or sum pooling to create a single dense representation.

Overview:

Firstly, BERT tokenizes the sentence into word pieces known as tokens. Then, it adds a special token [CLS] at the beginning of the sequence of generated tokens and a token [SEP] at the end of these tokens to separate sentences and indicate the end.

Next comes embedding and encoding. BERT transforms each token into a vector through an embedding matrix and multiple layers of encoders. These layers refine the representation of each token based on the contextual information provided by all other tokens in the sequence.

Finally, all token vectors are combined using pooling operations to form a unified dense representation.

However, one major disadvantage of BERT is its resource-intensive nature. Its base model has 110 million parameters, while the larger version has 340 million. Training and deploying BERT on edge devices require a lot of memory and computational resources.