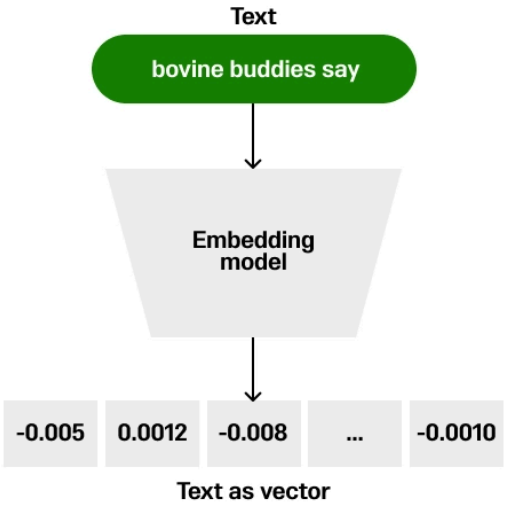
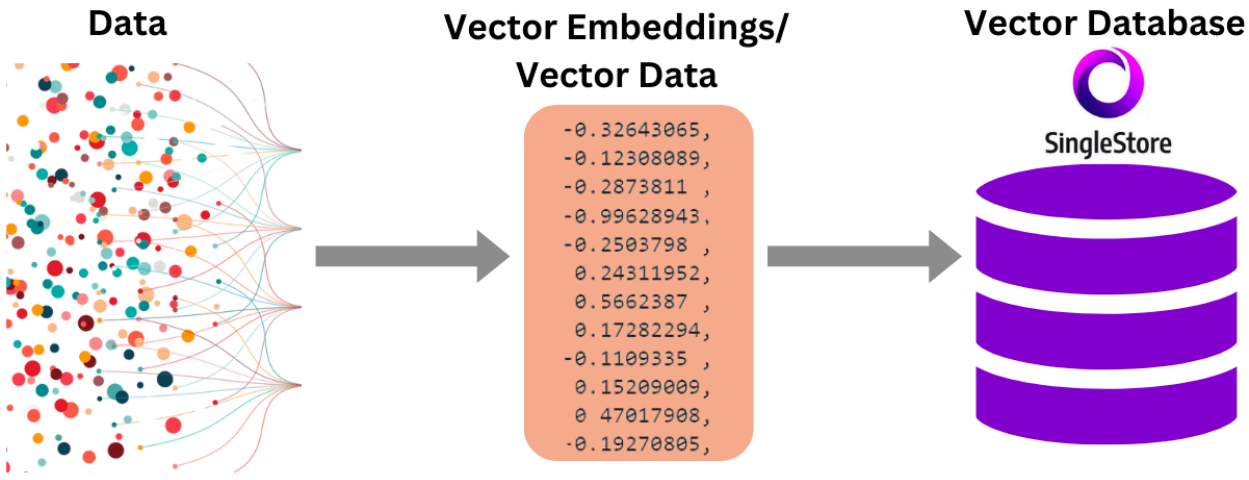
Embeddings



Embeddings in **Transformers** are crucial components that convert input data (like words, sub-words, or even images and other modalities) into **dense vector representations** that the model can process. They serve as the initial layer of representation, capturing semantic and contextual information.

Process of Input Tokenization & Embedding



Tokenization: Initially, a sentence is transformed into tokens using a tokenizer. Different tokenization methods exist, such as Byte Pair Encoding (BPE), WordPiece, and SentencePiece. The choice of tokenizer can influence the model's performance due to variations in how words are broken down into tokens.

For instance, consider the sentence:" Dolphins leap gracefully over waves."
A BPE tokenizer might tokenize this as: ['Dolphins', 'leap', 'grace', '##fully', 'over', 'waves', '.']

Embedding: Each token is then mapped to a **high-dimensional space (vector)** using learned embeddings. The embedding process involves converting tokens into a dense vector of floats (the embedding vector) that the model can process. Each vector essentially captures semantic and syntactic information about the token.

Example of Embedding:
Suppose we are embedding the word "waves" from the sentence above. The embedding layer will convert "waves" into a dense vector of size 512. Here is a hypothetical representation of the embedding:

waves = [0.032, -0.024, 0.118, …, 0.021] # A 512-dimensional vector

These vectors encapsulate the essence of each word, making it digestible for AI transformers. Some embeddings even capture the order of words or differentiate content types, such as questions from answers. In practice, an LLM would have more extensive values, enabling it to grasp the nuances and associations between words (for example, apple and banana are both a type of fruit, car and truck are both a type of automobile). With these vectors, transformers can process language by focusing on these numerical representations and their interrelations.

Positional Embeddings

- **Addressing Sequential Order**: Since Transformers process inputs in parallel, they don't inherently understand the order of words in a sequence.
- **Positional Encoding**: Positional embeddings are added to the input embeddings to provide this crucial positional information.
- **Sinusoidal Functions**: A common method is using a fixed function of sine and cosine waves, which provides a deterministic way to represent token positions and allows the model to generalize to different sequence lengths.

How They Work Together

- **Combined Representation**: The token embedding and its corresponding positional embedding are combined (often by addition).
- **Input to Transformer Layers**: This combined embedding forms the initial input to the Transformer's encoder or decoder layers.

- **Contextual Learning**: The self-attention mechanism within the Transformer then uses these embeddings to analyze the contextual relationships between tokens, allowing the model to understand the sentence's meaning effectively.