POST 14 - GEN AI

UNDERSTANDING THE CONCEPT OF "TOKENS" AND "TOKENIZATION" IN GENERATIVE AI & LLM

Generative AI Deep Dives,
Key concepts for Transformers - Part 1

GENERATIVE AI FOR ALL

DINESHLAL

DINESH LAL
(DIRECTOR, DATA SCIENCE)

WHAT IS COVERED IN THIS DOCUMENT?

- This document explains one of the important basic term, essential for any Generative AI solutions, i.e. Tokens
- The document first defines Tokens and Tokenization
- Then it explains various types of Tokenization
- Next the document delves into various methods of Tokenization, and how it is done, with the help of examples.

DEFINITION

Tokens:

- **Tokens are the smallest units of text that the transformer-based model processes.**
- They act as the building blocks for the model to understand the meaning of a sentence.
- For example, in the sentence “This mountain is tall” the tokens are: [“This”, “Mountain”, “is”, “tall”].

DEFINITION

Tokenization:

- **Tokenization is the process of breaking down raw text into tokens.**
- Before feeding text data into a transformer based model, it needs to be broken down into smaller, machine-readable units. This process is called tokenization.
- It involves splitting sentences into individual words or subword units

TYPES OF TOKENIZATION

Word based tokenization:

- Description: Word-based tokenization splits the text into tokens based on word boundaries. Each token corresponds to a complete word in the text.

EXAMPLE

Input Text: "The quick brown fox jumps over the lazy dog."

Tokens: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

TYPES OF TOKENIZATION

Word based tokenization:

Advantages	Disadvantages
• Straightforward and intuitive. • Preserves the semantic meaning of individual words.	• May not handle out-of-vocabulary words well. • Can be challenging for languages with complex morphology.

DINESHLAL

TYPES OF TOKENIZATION

Sub-word based tokenization:

- Description: Subword-based tokenization splits the text into smaller subword units, which may or may not correspond to complete words.
- This approach is particularly useful for handling out-of-vocabulary words and morphologically rich languages.

EXAMPLE

INPUT TEXT: "UNHAPPINESS"
TOKENS: ["UN", "HAPPI", "NESS"]

TYPES OF TOKENIZATION

Sub-word based tokenization:

Advantages	Disadvantages
• Handles out-of-vocabulary words effectively. • Adaptable to languages with complex morphology.	• Increases vocabulary size, leading to longer training times. • May produce suboptimal tokenizations for certain words.

DINESHLAL

TYPES OF TOKENIZATION

Character-based tokenization:

- Character-based tokenization represents each character in the text as a separate token.
- This approach is useful when dealing with languages that do not have clearly defined word boundaries or when handling text data at the character level.

EXAMPLE

INPUT TEXT: "HELLO"

TOKENS: ["H", "E", "L", "L", "O"]

TYPES OF TOKENIZATION

Character-based tokenization:

Advantages	Disadvantages
• Preserves information at the character level. • Handles languages with complex scripts or character-based languages effectively.	• Can result in larger input sequences compared to word-based tokenization. • May not capture higher-level semantic information present in words.

DINESHLAL

TYPES OF TOKENIZATION

Sentence-based tokenization:

- Description: Sentence-based tokenization splits the text into tokens based on sentence boundaries.
- Each token corresponds to a complete sentence in the text.

EXAMPLE

INPUT TEXT: "HELLO, HOW ARE YOU? I'M DOING WELL."

TOKENS: ["HELLO, HOW ARE YOU?", "I'M DOING WELL."]

TYPES OF TOKENIZATION

Sentence-based tokenization:

Advantages	Disadvantages
• Useful for tasks that operate at the sentence level. • Preserves sentence boundaries, which can be important for certain applications.	• Requires a reliable sentence segmentation algorithm. • May not be suitable for languages with flexible sentence structures.

TOKENIZATION METHODS

1

WordPiece Tokenization:

- WordPiece tokenization merges the most frequent word pieces or subword units to create a vocabulary of tokens.

Token Generation:

- Initialize the vocabulary with individual characters and a special end-of-word marker.
- Compute the frequency of each word piece or subword unit in the training data.
- Merge the most frequent word pieces iteratively until a predefined vocabulary size is reached.
- Tokenize input text using the generated vocabulary.

WordPiece Tokenization (Explained with an example):

- **Example Sentence:**
  "The quick brown fox jumps over the lazy dog."

- **Step 1:Initialize Vocabulary:** The initial vocabulary consists of individual characters and a special end-of-word marker.

Initial Vocabulary:
["T", "h", "e", " ", "q", "u", "i", "c", "k", "b", "r", "o", "w", "n", "f", "x", "j", "m", "p", "s", "v", "t", "l", "a", "z", "y", "d", "g", "."]

WordPiece Tokenization (Explained with an example):

- **Step 2: Merge Word Pieces:** Merge the most frequent word pieces iteratively to create subword units. Here, we'll illustrate a simplified example with a few merge operations:

1. Merge "e" + "r" → "er"
2. Merge "e" + "n" → "en"
3. Merge "o" + "v" → "ov"
4. Merge "l" + "a" → "la"

These merge operations are performed iteratively based on the frequency of word pieces until a predefined vocabulary size is reached.

TOKENIZATION METHODS

WordPiece Tokenization (Explained with an example):

- **Step 3: Tokenization:** Tokenize the input text using the generated subword vocabulary. Each token corresponds to a subword unit.

Subword Tokens:
["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

In the subword tokens, each token represents a merged subword unit generated by WordPiece tokenization. These subword tokens capture meaningful linguistic units in the input text while allowing for flexibility and adaptability to different word formations and languages.

TOKENIZATION METHODS

Byte-level Byte Pair Encoding (BBPE):

- BBPE Byte-level Byte Pair Encoding (BBPE) operates at the byte level and preserves byte-level information, making it suitable for handling multilingual text and preserving byte-level information such as special characters or byte-level encoding schemes.

Token Generation:

- Represent each character in the input text using its byte-level encoding.
- Merge the most frequent byte pairs iteratively to create subword units.
- Tokenize input text using the generated subword vocabulary.

Byte-level Byte Pair Encoding (BBPE)(Explained with an example):

- **Example Sentence:**
"The quick brown fox jumps over the lazy dog."

- **Step 1:Byte-level Representation:** Each character in the input sentence is represented using its byte-level encoding. For simplicity, let's consider ASCII encoding for English characters.

Byte-level Representation:["T", "h", "e", " ", "q", "u", "i", "c", "k", " ", "b", "r", "o", "w", "n", " ", "f", "o", "x", " ", "j", "u", "m", "p", "s", " ", "o", "v", "e", "r", " ", "t", "h", "e", " ", "l", "a", "z", "y", " ", "d", "o", "g", "."]

Byte-level Byte Pair Encoding (BBPE)(Explained with an example):

- **Step 3: Tokenize** the input text using the generated subword vocabulary. Each token corresponds to a subword unit.

Subword Tokens:
["The", "▁quick", "▁brown", "▁fox", "▁jumps", "▁over", "▁the", "▁lazy", "▁dog", "."]

In the subword tokens, "▁" represents the space character, and the remaining tokens are the merged subword units generated by BBPE. These subword tokens capture meaningful linguistic units in the input text while preserving byte-level information and supporting multilingual text processing.

TOKENIZATION METHODS

Unigram Language Model Tokenization:

- Unigram Language Model tokenization treats each token as a separate word and computes the likelihood of different subword units using a unigram language model.

Token Generation:

- Initialize the vocabulary with individual words.
- Compute the likelihood of different subword units using a unigram language model trained on the input text data.
- Tokenize input text using the generated vocabulary.

Unigram Language Model Tokenization (Explained with an example):

- **Example Sentence:**
  "The quick brown fox jumps over the lazy dog."

- **Step 1:Initialize Vocabulary:** The initial vocabulary consists of individual words in the input text.

Initial Vocabulary:
["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

Unigram Language Model Tokenization (Explained with an example):

- **Step 2: Compute the likelihood of different subword units using a unigram** language model trained on the input text data. In this simplified example, we'll illustrate the likelihoods of a few subword units:

1. "qu" - likelihood: 0.4
2. "ick" - likelihood: 0.6
3. "br" - likelihood: 0.3
4. "own" - likelihood: 0.5
5. "fox" - likelihood: 0.7

These likelihoods are based on the frequency of subword units observed in the training data.

Unigram Language Model Tokenization (Explained with an example):

Step 3: Tokenization: Tokenize the input text using the generated subword vocabulary. Each token corresponds to a subword unit with a likelihood score.

Subword Tokens:
["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

In the subword tokens, each token represents a subword unit selected based on its likelihood score computed from the unigram language model. These subword tokens capture meaningful linguistic units in the input text while providing flexibility and adaptability to different word formations and languages.

Thank You

SPECIAL THANKS TO CHATGPT, OPEN AI, COPILOT FOR THE SUPPORT ON CONTENT