

Lecture 8: The GPT Tokenizer (Byte Pair Encoding)

1. The Three Types of Tokenization

To understand why BPE is necessary, Dr. Dander first contrasts it with the two other primary methods of tokenization:

- **Word-based Tokenization:**
 - **Mechanism:** Every individual word is treated as a unique token (e.g., "The", "fox", "chased").
 - **Pros:** Simple to implement.
 - **Cons:**
 1. **Huge Vocabulary:** English has 170k–200k words; accommodating all of them requires a massive vocabulary size.
 2. **The OOV Problem:** It struggles with "Out of Vocabulary" words. If a user inputs a word not seen during training (e.g., "football" when the model only knows "cricket"), the model errors out or uses a generic "unknown" token.
 3. **Loss of Meaning:** It treats related words like "boy" and "boys" as completely unrelated tokens, missing the shared root.
- **Character-based Tokenization:**
 - **Mechanism:** Every character (a, b, c...) is a token.
 - **Pros:** Extremely small vocabulary (approx. 256 for English) and eliminates the OOV problem entirely (any word can be spelled out).
 - **Cons:**
 1. **Loss of Meaning:** Individual characters carry no semantic meaning.
 2. **Sequence Length:** A single word like "dinosaur" becomes 8 separate tokens, making the input sequence for the LLM inefficiently long.

2. Subword Tokenization (The Solution)

Subword tokenization is described as the "Best of Both Worlds" because it bridges the gap between word and character tokenization. It follows two main rules:

1. **Rule 1:** Do not split frequently used words (keep them as whole tokens).
 2. **Rule 2:** Split rare words into meaningful smaller subwords or characters.
- *Example:* "Boy" (frequent) remains "Boy". "Boys" (rare) might become "Boy" + "s".
 - *Benefit:* This allows the model to recognize that "Token," "Tokens," and "Tokenization" all share the same root meaning.

3. Byte Pair Encoding (BPE) Algorithm

BPE is a specific type of subword tokenization. Historically, it was introduced in **1994** as a data compression algorithm.

How the Algorithm Works (Conceptual):

1. Scan the data to find the **most frequent pair** of consecutive bytes.
2. **Merge** that pair and replace it with a new byte/token that does not exist in the data.
3. Repeat this process iteratively.

Applying BPE to LLMs (Detailed Example): The lecture demonstrates BPE using a dataset of four words: **Old, Older, Finest, Lowest**.

- **Initialization:** Append a delimiter like `</w>` to mark the end of words.
- **Frequency Count:**
 - `old`: 7 times
 - `older`: 3 times
 - `finest`: 9 times
 - `lowest`: 4 times.
- **Iteration 1:** The algorithm finds that the characters "e" and "s" appear together most frequently (13 times: 9 in *finest*, 4 in *lowest*). It merges them into a new token `es`.
- **Iteration 2:** It finds that `es` and `t` appear together frequently. It merges them into `est`.
- **Iteration 3:** It merges `est` with the end token `</w>` to create `est</w>`.
 - *Significance:* This creates a token specifically for the *suffix* "est" at the end of a word. This allows the model to distinguish between "estimate" (start of word) and "highest" (end of word).
- **Result:** The algorithm learns tokens like `old` (root) and `est` (suffix), capturing grammatical structures automatically.

4. Advantages of BPE for GPT Models

Why do GPT-2 and GPT-3 use BPE?

1. **Vocabulary Management:** It reduces the vocabulary size significantly compared to word-based methods. GPT-2 uses a vocabulary of **50,257 tokens**, which is roughly 1/3 the size of a standard English word vocabulary.
2. **Solves OOV Errors:** It can handle *any* unknown word by breaking it down.
 - *Example:* The gibberish string "akwirwier" does not crash the model; BPE breaks it down into subwords/characters like `ak`, `w`, `ir`, etc..
3. **Semantic Capture:** It retains root meanings (like `old`) while handling variations (like `older`) efficiently.

5. Practical Implementation: Tiktoken

While one can code BPE from scratch, in practice, developers use OpenAI's open-source library called `tiktoken`.

- **Efficiency:** It is highly optimized for performance.
- **Usage:**
 - The lecture demonstrates using `tiktoken.get_encoding("gpt2")`.

- **End of Text Token:** The vocabulary includes a special token `<|endoftext|>` (ID **50256**) used to separate unrelated documents during training.
- **Handling Unknowns:** The lecture demonstrates passing the phrase "some unknown place." The tokenizer successfully encodes it without error by breaking "unknown" and "place" into smaller known subwords.