**Training Custom Tokenizer**

**What does "Training Our Own Tokenizer" mean?**

Normally, tokenizers like BertTokenizer, GPT2Tokenizer, or T5Tokenizer are pre-trained on large general datasets like Wikipedia, CommonCrawl, etc.  
They split the text into tokens (pieces) that the model understands.

👉 Training your own tokenizer means building a tokenizer from scratch, using your own domain-specific data instead of relying on prebuilt tokenizers.

✅ You teach the tokenizer how to split words best for your data.

*from tokenizers import ByteLevelBPETokenizer*

*# Initialize a tokenizer*

*tokenizer = ByteLevelBPETokenizer()*

*# Train the tokenizer on your files*

*tokenizer.train(*

*files=["./tech\_resumes.txt"],*

*vocab\_size=30\_000,*

*min\_frequency=2,*

*special\_tokens=["<s>", "<pad>", "</s>", "<unk>", "<mask>"]*

*)*

*# Save the tokenizer*

*tokenizer.save\_model("./my\_tokenizer")*

**1. Why Use BPE (Byte-Pair Encoding) to Train a Custom Tokenizer?**

**Answer:**

* **BPE (Byte-Pair Encoding)** is the most commonly used method today because:
  + It balances between splitting words too much (character-level) and splitting too little (word-level).
  + It handles rare and out-of-vocabulary (OOV) words efficiently.
  + It is fast, produces a compact vocabulary, and ensures good coverage of common and uncommon words.
  + It is widely adopted by famous models like **GPT-2**, **Roberta**, and **DistilBERT**.

**Example:**

* In resume data, a word like "TensorFlowDeveloper" would split into meaningful subwords like "Tensor", "Flow", and "Developer" rather than individual characters, maintaining useful semantic units.

**2. What Are the Other Algorithms for Training a Custom Tokenizer?**

**Answer:**

Here are the main algorithms you can use to train a tokenizer:

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Description** | **Used By** | **Good For** |
| Word-Level Tokenizer | Splits by spaces into whole words | Early models (word2vec) | Simple tasks, but bad with unknown words |
| Character-Level Tokenizer | Splits into individual characters | Some LSTM models | Languages like Chinese or Japanese, or character-level tasks |
| BPE (Byte-Pair Encoding) | Merges most frequent token pairs into larger tokens | GPT-2, Roberta | General-purpose, fast, efficient |
| Unigram Language Model | Probabilistic model selecting tokenizations with highest likelihood | T5, ALBERT | Noisy/multilingual data, complex datasets |
| WordPiece | Variant of BPE, pre-tokenizes rare words into known subwords | BERT, DistilBERT | Excellent for text classification tasks |

**Quick Comparison Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Simple | Handles Rare Words | Training Speed | Best For |
| Word-Level | ✅ | ❌ | 🚀 Fast | Basic tasks |
| Char-Level | ✅ | ✅ | 🚀🚀 Super Fast | Asian languages, code |
| BPE | ✅ | ✅ | 🚀 | General NLP tasks |
| WordPiece | ✅ | ✅ | 🚀 (slightly slower) | Classification tasks |
| Unigram | ✅✅ | ✅✅ | 🐿 Slow | Noisy, multilingual text |