Tokenization (Beginner-Friendly Explanation) :-

Tokenization is the process of **breaking text into smaller pieces called tokens** so an AI model can understand and process it.

In modern NLP systems like GPT-3 and GPT-4, tokenization is the **first step before training or inference**.

---

1 What is a Token?

A **token** is a small unit of text.

It could be:

- a **word**
- part of a word
- a **character**
- punctuation

Example sentence:

```
I love learning AI
```

Tokens could be:

```
["I", "love", "learning", "AI"]
```

Each token is then converted to a **number (token ID)** so a neural network (e.g., in PyTorch) can process it.

Example:

```
["I", "love", "learning", "AI"]
→
[10, 245, 678, 92]
```

Computers **only understand numbers**, not text.

2 Why Tokenization is Needed

AI models cannot directly read text.

They need text converted into **numerical representations**.

Pipeline:

```
Text
↓
Tokenization
↓
Token IDs
↓
Neural Network
```

Example:

```
Hello world
↓
["Hello", "world"]
↓
[15496, 995]
```

---

3 Types of Tokenization

1. Word Tokenization

Splits text by spaces.

Example:

```
I love AI
```

Tokens:

```
["I", "love", "AI"]
```

Traditional libraries like NLTK use this.

Problem:

- Vocabulary becomes **huge**.

---

2. Character Tokenization

Breaks text into characters.

Example:

```
Hello
```

Tokens:

```
["H","e","l","l","o"]
```

Problem:

- Sequences become **very long**.

---

3. Subword Tokenization (Used in LLMs) □

Modern models use **subwords**.

Example:

```
unbelievable
```

Tokens:

```
["un", "believe", "able"]
```

This keeps the vocabulary **small and efficient**.

Algorithms used include:

- Byte Pair Encoding
- WordPiece
- SentencePiece

---

4 Example Tokenization in Python

Hugging Face Tokenizers Install it :-

C:\Users\Big Data>d:

D:\>env\scripts\activate

(env) D:\>pip install tokenizers

Using Hugging Face Tokenizers encode-text.py :

```
from tokenizers import Tokenizer

tokenizer = Tokenizer.from_pretrained("bert-base-uncased")

output = tokenizer.encode("Hello I love AI")

print(output.tokens)
print(output.ids)
```

Output example:

```
['[CLS]', 'hello', 'i', 'love', 'ai', '[SEP]']
[101, 7592, 1045, 2293, 9932, 102]
```

Here is an example of what the **`data.txt`** file should look like when training a tokenizer with Hugging Face Tokenizers.

You just need **plain text**, usually **one sentence per line**.

Example `data.txt`

```
Machine learning is amazing
I love artificial intelligence
Transformers are powerful models
Deep learning is changing the world
Natural language processing helps computers understand text
Large language models can generate human like responses
AI is used in healthcare finance and robotics
```

Folder structure example

```
project/
│
├── train_tokenizer.py
└── data.txt
```

Where:

- `train_tokenizer.py` → your Python script
- `data.txt` → training text for tokenizer

train_tokenizer.py file code:-

```
from tokenizers import Tokenizer

from tokenizers.models import BPE

from tokenizers.trainers import BpeTrainer

from tokenizers.pre_tokenizers import Whitespace
```

```
# Step 1: Create tokenizer
tokenizer = Tokenizer(BPE())

# Step 2: Pre-tokenization (split by spaces)
tokenizer.pre_tokenizer = Whitespace()

# Step 3: Trainer configuration
trainer = BpeTrainer(vocab_size=100)

# Step 4: Training data
with open("data.txt", "w") as f:
    f.write("Machine learning is amazing\n")
    f.write("I love artificial intelligence\n")
    f.write("Transformers are powerful models\n")

# Step 5: Train tokenizer
tokenizer.train(["data.txt"], trainer)

# Step 6: Encode text
encoding = tokenizer.encode("Machine learning is amazing")

print("Tokens:", encoding.tokens)
print("Token IDs:", encoding.ids)
```

```
# Step 7: Decode tokens back to text

decoded = tokenizer.decode(encoding.ids)


print("Decoded text:", decoded)
```

Example Output

```
Tokens: ['Machine', 'learning', 'is', 'amazing']
Token IDs: [23, 45, 12, 67]
Decoded text: Machine learning is amazing
```