The AutoTokenizer used with LLaMA models typically relies on **SentencePiece** tokenization, specifically using **Byte-Pair Encoding (BPE)** or **Unigram Language Model** encoding depending on the version.

The AutoTokenizer in Hugging Face Transformers automatically loads the correct tokenizer configuration based on the model you specify (e.g., "meta-llama/Llama-2-7b-hf"), and under the hood, it uses the LlamaTokenizer or LlamaTokenizerFast, both based on SentencePiece.

Key Technique:

SentencePiece + BPE (for LLaMA 1/2) or **SentencePiece + Unigram (for LLaMA 3)**

Why SentencePiece?

- It allows subword-level tokenization.
- Works directly on raw Unicode text (no preprocessing like whitespace splitting required).
- Ideal for multilingual models or models trained on varied data.

complete dry run on a more complex sentence, step-by-step, using LLaMA's AutoTokenizer (e.g., from "meta-llama/Llama-2-7b-hf").

Sentence:

"The quick brown fox jumps over the lazy dog, effortlessly."

Step 1: Load Tokenizer

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b-hf")
```

Step 2: Tokenize Sentence

```
tokens = tokenizer.tokenize("The quick brown fox jumps over the lazy dog, effortlessly.")
print(tokens)
```

Output:

```
['▁The', '▁quick', '▁brown', '▁fox', '▁jumps', '▁over', '▁the', '▁lazy', '▁dog', ',', '▁effort',
'lessly', '.']
```

Step 3: Tokenization Logic (Dry Run):

Pre-tokenization (with special marker):

The sentence is first transformed into:

```
"▁The▁quick▁brown▁fox▁jumps▁over▁the▁lazy▁dog,▁effortlessly."
```

Notice:

- ▁ marks the beginning of a word.
- SentencePiece handles this, so words are not split by spaces directly.

Subword Matching (via BPE):

- Each part is matched greedily to longest vocab tokens:
  - '▁The' → one token
  - '▁quick' → one token
  - '▁effortlessly' → gets broken into: '▁effort', 'lessly' (since 'effortlessly' is not in vocab but parts of it are)

Step 4: Token IDs

```
ids = tokenizer.convert_tokens_to_ids(tokens)
print(ids)
```

Example Output:

```
[1332, 2398, 4149, 2317, 3664, 2934, 2781, 4009, 1443, 29892, 13204, 22765, 29889]
```

(*Note: IDs may vary slightly by tokenizer version.*)

Step 5: Decode Back

```
decoded = tokenizer.decode(ids)
print(decoded)
```

Output:

```
"The quick brown fox jumps over the lazy dog, effortlessly."
```

It reconstructs the sentence accurately.

Summary Table:

Word	Token(s)	Token ID(s)
"The"	'▁The'	1332
"quick"	'▁quick'	2398
"brown"	'▁brown'	4149
"fox"	'▁fox'	2317
"jumps"	'▁jumps'	3664
"over"	'▁over'	2934
"the"	'▁the'	2781
"lazy"	'▁lazy'	4009
"dog"	'▁dog'	1443
","	','	29892
"effortlessly"	'▁effort', 'lessly'	13204, 22765
"."	'.'	29889