Folder Structure

```
ALLLLM/
|- configs/              -> Model configs (e.g. phi-2.json, my-llm.json)
|- data/                 -> Raw .txt files and generated .jsonl/.bin tokenized datasets
|- scripts/
|  |- train_tokenizer.py   -> Trains custom tokenizer (BPE)
|  |- tokenize_dataset.py  -> Tokenizes corpus using existing tokenizer
|- train.py              -> Trains the LLM using tokenized files
|- checkpoints/          -> Stores model checkpoints and seen_datasets.json
|- src/
|  |- model/             -> Model components (attention, FFN, etc.)
|  |- tokenizer/         -> TokenizerManager abstraction (HF, tiktoken, custom)
|  |- data/              -> TextDataset class for .jsonl/.bin loading
```

Step 1: Tokenization

```
$ python scripts/tokenize_dataset.py --model phi-2
```

- Reads raw text from path in phi-2.json (or my-llm.json)
- Tokenizes using the specified tokenizer (Huggingface or custom BPE)
- Saves output as: data/<basename>.cached.<seq_len>.<timestamp>.jsonl
- Optionally also saves .bin and .meta.json

Step 2: Training

```
$ python train.py --model phi-2 --only_new
```

- Scans all .jsonl and .bin files in the data/ directory
- Ignores files already listed in checkpoints/phi-2/seen_datasets.json
- Trains on only new files (incremental training)
- Saves model checkpoints (e.g. model_epoch0.pt)
- Updates seen_datasets.json with newly trained files

Dataset Tracking

checkpoints/phi-2/seen_datasets.json

- Stores a list of .jsonl files that have already been used for training
- Ensures files are not used more than once
- Keeps training safe and incremental
- Automatically updated after every training run

Tokenizer vs Dataset Tokenization (Custom LLM)

Tokenizer Training (tokenizer.json):

- Should be done only once, ideally on the largest dataset.
- Output: tokenizer.json (fixed vocabulary and merge rules).
- Overwriting tokenizer.json will invalidate previous .jsonl files.

Dataset Tokenization (jsonl/bin):

- Can be done repeatedly for new data using tokenize_dataset.py.
- Generates new .jsonl files with timestamps.
- Always uses the fixed tokenizer.json (never modifies vocabulary).

Correct Workflow for Custom LLM

1. Train tokenizer on large dataset:

```
$ python scripts/train_tokenizer.py --model my-llm
```

2. Tokenize raw text anytime:

```
$ python scripts/tokenize_dataset.py --model my-llm
```

3. Train model incrementally:

```
$ python train.py --model my-llm --only_new
```

Reminder:

Do not retrain tokenizer on small corpora later always reuse the original.