Folder Structure

```
ALLLLM/
|- configs/            -> Model configs (e.g. phi-2.json)
|- data/               -> Raw .txt and tokenized .jsonl/.bin files
|- scripts/
|  |- tokenize_dataset.py  -> Script to tokenize raw text
|- train.py            -> Training entry script
|- checkpoints/        -> Checkpoints + seen_datasets.json
|- src/
|  |- model/           -> LLM components (attention, ffn, etc.)
|  |- tokenizer/       -> TokenizerManager for multiple backends
|  |- data/            -> TextDataset class
```

Step 1: Tokenization

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

- Reads raw text from path in phi-2.json
- Tokenizes using specified tokenizer
- Saves as: data/yourfile.cached.<seq_len>.<timestamp>.jsonl
- Optionally also saves .bin and .meta.json

Step 2: Training

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

- Scans data/*.jsonl
- Ignores already-seen files (via seen_datasets.json)
- Trains on only new files
- Saves checkpoint and updates seen_datasets.json

Dataset Tracking

Tracked in: checkpoints/phi-2/seen_datasets.json

- Keeps record of all .jsonl files already trained

- Ensures step-wise training without re-processing old data

Summary

- Tokenize with tokenize_dataset.py
- Train using train.py --only_new
- All configuration is centralized in configs/<model>.json
- Tracks training progress and dataset history automatically

Step-wise Training with .jsonl and .bin

You can incrementally train your model using tokenized datasets stored as either .jsonl or .bin files.

Step 1: Tokenize

- Run tokenize_dataset.py for new .txt files.
- This creates .jsonl and optionally .bin files in /data folder.
- Example: data/myfile.cached.2048.20240701_101512.jsonl

Step 2: Train

- Run train.py with:

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

- This automatically scans all .jsonl and .bin files in /data
- It filters out previously trained files using checkpoints/phi-2/seen_datasets.json

Supported Formats:

- .jsonl -> Text-based token ID chunks (one per line)
- .bin   -> Binary torch.save'd token tensors

Conditions for Valid Input:

- Must match seq_len in phi-2.json (e.g., 2048)
- Must be tokenized using the same tokenizer_path (e.g., microsoft/phi-2)
- Must not already appear in seen_datasets.json

This supports:

- Initial training on multiple files
- Later step-wise training on new data only
- Training resumption from latest checkpoint