Assignment 1 – Data Collection & Preprocessing Report

1. Dataset and Raw Data Collection

Source dataset: `dylanebert/openwebtext` on Hugging Face.

Motivation: This is a parquet-based mirror of the OpenWebText corpus, which is widely used for language-model pretraining.

Loading mode:

- - Used the FULL mode (dev_mode=False) for the final run.
- - Log messages:

- [INFO] Running in FULL mode (approx >1GB raw text).

- [FULL MODE] Loading dylanebert/openwebtext with at most 250000 examples.

Number of documents loaded:

- - Loaded 250000 raw documents from dylanebert/openwebtext.

2. Approximate Data Size

Public statistics for OpenWebText indicate approximately 8M documents and about 38GB total text.

This implies an average of roughly 4–5 KB of text per document.

Our subset uses 250,000 documents, so the total raw text size is approximately:

- - 250k × 4–5 KB ≈ 1.0–1.2 GB of raw text.

This satisfies the assignment requirement that the raw text size exceeds 1 GB.

3. Cleaning and Deduplication

Cleaning operations applied to each document:

- - Convert all text to lowercase.
- - Collapse multiple spaces/newlines into a single space (\s+ → " ").
- - Remove documents with fewer than 50 words.

Deduplication strategy:

- - Maintain a set of already-seen cleaned texts.
- - Skip any document whose cleaned text string has appeared before (exact duplicate removal).

Result after cleaning and deduplication:

- - Log: After cleaning & deduplication: 250000 documents remain.
- - For this subset, the number of documents remained the same (no extremely short or exactly duplicate documents in the 250k slice).

4. Tokenization and Chunking

Tokenizer:

- - AutoTokenizer.from_pretrained("gpt2").
- - pad_token is set to eos_token, giving pad_token_id = 50256.
- - Log: Using tokenizer: gpt2 (pad_token_id=50256).

Tokenization procedure:

- - For each cleaned document, call tokenizer.encode(text, add_special_tokens=True).
- - This may produce long sequences (e.g., a warning about sequence length > 1024 for the GPT-2 model), but since we only use the token IDs for chunking, it does not break the pipeline.

Chunking strategy:

- - Block size: 512 tokens per chunk.
- - Minimum chunk length: 32 tokens; shorter chunks are discarded.
- - Iterate over each sequence in steps of 512 tokens and keep all chunks with length ≥ 32.
- - Cap the total number of chunks with max_chunks = 100000 in FULL mode to control memory/time.

Final number of tokenized chunks:

- - Log: Total tokenized chunks: 100000.

5. DataLoader and Saved Sample Dataset

Custom PyTorch Dataset:

- - Each item is a torch.LongTensor representing one token ID sequence (one chunk).

Collate function (collate_fn):

- - Find the max sequence length in the current batch.
- - Pad all sequences to this length using pad_token_id.
- - Create corresponding attention masks (1 for real tokens, 0 for padding).
- - Return a batch dictionary with:
- - "input_ids": tensor of shape [batch_size, seq_len].
- - "attention_mask": tensor of shape [batch_size, seq_len].

DataLoader configuration:

- - batch_size = 8.
- - shuffle = True.
- - Use the custom collate_fn.
- - Log: DataLoader is ready.

Saving sample batches:

- - Take the first 5 batches from the DataLoader.
- - Store them as a list[dict] (each dict contains input_ids and attention_mask).
- - Save to disk via torch.save(..., "sample_dataset.pt").
- - Log: Saved sample batches to sample_dataset.pt.

Purpose of sample_dataset.pt:

- - Acts as a small representative snapshot of the pretraining dataset.
- - Can be used to quickly test model code without running the full pipeline again.

6. Limitations and Potential Improvements

Current limitations:

- - Disk space and compute constraints limit the maximum number of documents and chunks processed in a single run.
- - Advanced filtering such as language detection or quality scoring is not yet implemented.