Data Collection and Preprocessing for Foundation Model Pre-Training

1. Dataset Sources and Total Size

To meet the assignment requirement of at least 1GB of raw text, I collected data from three diverse public sources using the Hugging Face datasets library. The selection prioritizes domain diversity, covering encyclopedic knowledge, general web content, and news.

Data Sources:

- **WikiText-103 (wikitext-103-raw-v1):** A large-scale collection of Good and Featured articles from Wikipedia.
- **OpenWebText:** An open-source recreation of the WebText dataset, representing general web content.
- **AG News:** A collection of news articles used to add a specific journalistic domain to the corpus.

Dataset Statistics:

The total raw text collected exceeded the 1GB requirement. The breakdown calculated during the collection phase is as follows:

Dataset	Raw Size (MB)	Document Count
WikiText-103	117.35 MB	250,000
OpenWebText	957.90 MB	200,000
AG News	12.94 MB	50,000
Total	**1,088.19 MB (1.063 GB)**	**500,000**

2. Cleaning Strategies and Reasoning

Raw text data often contains noise, duplicates, and formatting issues that can negatively impact model learning. I implemented a three-stage preprocessing pipeline to ensure data quality.

2.1 Deduplication

Strategy: Exact deduplication using hash-based matching. **Result:** Removed **20,003 duplicate documents** (primarily from WikiText). **Reasoning:** Duplicate documents artificially inflate training metrics and can cause the model to memorize specific phrases rather than learning generalized language patterns.

2.2 Text Normalization

Strategy: A normalization pipeline using Python's `re` library to:

- Remove HTML tags and Markdown (e.g., links, headers).
- Convert text to lowercase.
- Remove extra whitespace and irrelevant symbols. **Reasoning:** Removing artifacts ensures the model learns natural language rather than markup syntax. Lowercasing helps in standardizing the inputs and reducing the vocabulary size required.

2.3 Filtering Low-Quality Documents

Strategy: Filtering out "short" documents containing fewer than 50 words. **Result:** Removed **95,202 documents** (**24.3% of the deduplicated corpus**).

- *Before Cleaning:* 411,633 documents (deduplicated).
- *After Cleaning:* 296,428 documents. **Reasoning:** Extremely short texts often lack sufficient context for learning long-range dependencies. Removing them improves the average information density of a training batch.

3. Tokenization Choices

I utilized the **GPT-2 tokenizer** via Hugging Face's `AutoTokenizer`.

- **Tokenizer Type:** Byte-Pair Encoding (BPE).
    - *Reasoning:* BPE tokenization was chosen because it effectively handles out-of-vocabulary words through subword units, balancing vocabulary size and representation capability.
- **Block Size:** 512 tokens.
    - *Reasoning:* The 512-token block size balances context length with memory efficiency, fitting within typical GPU memory constraints while providing sufficient context for language modeling compared to smaller windows like 128 or 256.
- **Chunking Strategy:** I implemented a custom `tokenize_and_chunk` function to split long documents into 512-token blocks. Documents shorter than 512 tokens, or

incomplete chunks at the end of longer documents, were discarded to maintain uniform block sizes and eliminate the need for padding.

Tokenization Results by Source: While the raw document count was high, the final yield of usable training blocks revealed significant disparity:

- **WikiText-103:** 189 blocks (0.06%)
- **OpenWebText:** 312,366 blocks (99.94%)
- **AG News:** 0 blocks (0%)
- **Total:** 312,555 blocks

4. Custom Data Loader Implementation

I implemented a custom data loader using **PyTorch** (`torch.utils.data`).

- **Dataset Class:** The `TextDataset` class flattens the dictionary of tokenized blocks into a single list for seamless indexing. It returns both `input_ids` and `labels` in each sample. For causal language modeling, labels are set equal to `input_ids`, enabling the model to learn next-token prediction in a self-supervised manner.
- **DataLoader:** Configured with `batch_size=8`, `shuffle=True`, and `num_workers=2` to ensure efficient, parallel data loading during training.

5. Challenges Encountered

5.1 Dataset Size Estimation

- **Challenge:** Initial sample sizes were underestimated, resulting in a total raw size of <1GB.
- **Solution:** Iteratively increased sample sizes for OpenWebText and WikiText and recalculated the total size until reaching the target of **1.063 GB**.

5.2 Empty Documents in WikiText

- **Challenge:** The WikiText-103 dataset contained approximately **35% empty or whitespace-only documents**.
- **Impact:** I collected only 161,633 non-empty documents despite requesting 250,000 samples.
- **Solution:** Adjusted expectations for document yield and documented the natural data loss during the collection phase.

5.3 Processing Time

- **Challenge:** Tokenizing 200,000 OpenWebText documents was computationally expensive, taking **approximately 30 minutes on CPU** to complete.
- **Solution:** Implemented checkpointing (saving `.pkl` files) to save intermediate results. This prevented data loss in case of disconnection and avoided re-running the tokenization for the entire corpus.

5.4 AG News Filtering

- **Challenge: 90.7%** of AG News documents (45,337 out of 50,000) were removed due to the <50 word threshold.
- **Analysis:** The dataset primarily contains brief headlines or summaries rather than full-length articles.
- **Impact:** This resulted in **minimal contribution to final tokenized blocks (0 blocks)** from the AG News domain, as the remaining documents were too short to form full 512-token sequences.

5.5 Memory Management

- **Challenge:** Loading over 1GB of raw text data and processing it simultaneously creates significant memory pressure.
- **Solution:** Utilized **streaming mode** (`streaming=True`) when loading the large WikiText and OpenWebText datasets. This allowed the script to iterate through samples one by one without loading the entire dataset into RAM, preventing environment crashes.

6. Reflections on Preprocessing Impact

The most critical insight from this assignment is that **domain diversity in raw documents does not guarantee domain diversity in tokenized blocks**.

Despite collecting data from three distinct domains, **99.94% of the final training blocks came from OpenWebText alone**. This demonstrates that uniform preprocessing decisions (e.g., a fixed 512 block size and global 50-word filtering threshold) can inadvertently erase entire domains if their characteristics differ from the norm.

Future work must address this imbalance by either:

1. **Adjusting filtering thresholds per domain** (e.g., allowing shorter texts for news).
2. **Oversampling** underrepresented domains like WikiText.
3. **Concatenating multiple short documents** (like AG News headlines) into single 512-token blocks to maintain domain balance in the final training data.