Assignment 1

Data Collection and Preprocessing for Foundation Model Pre-Training

Environment: Python 3.x | PyTorch | Hugging Face Datasets & Transformers | TQDM | Regex

---

1. Dataset Sources and Coverage

Two public, large-scale English text datasets were used: **Wikipedia (20220301.en)** and **OpenWebText**.

Wikipedia provides factual and encyclopedic writing, while OpenWebText contributes informal, conversational, and news-style content.

Together, they exceed **1 GB of raw text** and span multiple domains, satisfying the assignment's requirements for scale, diversity, and public availability.

Both were accessed in **streaming mode** through the Hugging Face `datasets` API to avoid local storage overflow and ensure scalability.

---

2. Cleaning and Normalization

A lightweight but comprehensive cleaning pipeline was implemented to ensure text quality and consistency.

Each document was Unicode-normalized (NFKC), lowercased, and stripped of HTML tags and redundant whitespace.

Sequences with fewer than 50 words were removed as low-quality samples.

Long repeated characters (e.g., "!!!!!!") were compressed, and exact duplicates were removed using a hash set.

This process improved overall data consistency and reduced noise typical of web text, directly supporting higher-quality pretraining data.

---

3. Tokenization and Chunking

Text was tokenized using the **GPT-2 Byte-Pair Encoding (BPE)** tokenizer for full compatibility with transformer-based language models.

Because GPT-2 lacks a predefined padding token, its end-of-sequence (EOS) token was reused for padding during batching.

Each tokenized document was split into **fixed-length blocks of 128 tokens**, ensuring that all samples fit within the model's context window and enabling efficient parallel processing.

This single tokenization and chunking step produced clean, uniform training segments while retaining coverage of long-form content.

4. Custom Data Loader (PyTorch)

A **custom PyTorch Dataset and DataLoader** were implemented to handle variable-length token sequences.

The dataset returns both *input IDs* and *labels* for autoregressive next-token prediction.

A custom collate function dynamically pads shorter sequences within each batch to match the longest sample, ensuring equal tensor shapes and stable batch training.

This satisfies the "custom data loader" requirement for efficient batching, shuffling, and padding of tokenized data.

5. Requirement Compliance Summary

Requirement	Implementation Summary
≥ 1 GB raw text	Wikipedia + OpenWebText
Public sources	Both open Hugging Face datasets
Domain diversity	Encyclopedic + Web content
Cleaning pipeline	Normalization, deduplication, filtering
Tokenizer	GPT-2 BPE

Requirement	Implementation Summary
Handling long sequences	128-token chunking
Custom DataLoader	PyTorch Dataset + dynamic padding
Reproducibility	Standard Python environment

6. Key Design Decisions

- **Streaming I/O:** Avoided local disk errors while maintaining full-dataset scalability.
- **Pad = EOS:** Simplified GPT-2 batching without changing vocabulary.
- **Block size = 128:** Chosen for fast demonstration yet easily extendable.
- **Short-text threshold = 50 words:** Balanced data retention vs quality.
- **Exact deduplication:** Used set-based lookup for O(1) efficiency.

Each design choice reflects trade-offs between computational efficiency and representational quality.

7. Challenges and Mitigations

Challenge	Mitigation
Disk capacity during download	Used `streaming=True` to avoid caching
Package version conflicts	Reinstalled `datasets` / `pyarrow` compatibly
Variable-length sequences	Implemented dynamic padding in collate function
Dataset domain imbalance	Combined Wikipedia and OpenWebText for coverage

8. Reflections

This preprocessing pipeline demonstrates a complete, reproducible foundation-model data workflow.

Cleaning and deduplication enhanced signal-to-noise ratio.

BPE tokenization balanced vocabulary size and generalization.

Chunking and dynamic padding ensured efficient mini-batch training.

The result is a scalable, high-quality dataset suitable for pretraining transformer language models.

Deliverables

- `data_collection_preprocessing.ipynb` — fully annotated notebook
- `sample_dataset.pt` — example tokenized and padded batch
- `Assignment1_Report.pdf` — this report
