Data Collection & Preprocessing Report

(Submitted by: Ahsan Zafar H. Syed, NUID: 002801441)

1. Dataset Sources and Total Size

For this project, three publicly available text datasets from the HuggingFace Hub were used. To meet the requirement of preparing a dataset of at least 1GB while preserving storage efficiency, all large datasets were loaded in streaming mode, preventing full downloads (which would otherwise exceed 40GB+).

The final dataset contains approximately 1GB of cleaned raw text sampled from:

Datasets Used

1. Wikimedia/Wikipedia (English Dump)
   Provides factual, encyclopedic, well-structured text covering diverse topics.

2. HuggingFaceFW/FineWeb-Edu (sample-10BT subset)
   A curated subset of large-scale web crawls with general web-style content, including articles, blog posts, and discussions.

3. SetFit/AG News
   A smaller news dataset used to introduce journalistic writing styles for additional domain variety.

Why These Datasets?

- Combines encyclopedic, web, and news domains
- Provides stylistic and linguistic diversity
- Supported by streaming, avoiding multi-GB downloads
- Stable, widely used datasets suitable for language model pretraining
- Sampling continued until the global target (~1.5GB raw cleaned text) was reached.

2. Cleaning Strategies and Reasoning

High-quality preprocessing is essential for stable and efficient language model pretraining. The following pipeline was applied.

Cleaning Steps

1. Lowercasing
   Ensures vocabulary consistency and prevents BPE token explosions caused by inconsistent casing.

2. HTML/Markup Removal
   FineWeb data often contains HTML fragments. A lightweight regex was used to remove <tags> and inline markup.

3. Whitespace Normalization
   All repeated whitespace was collapsed into a single space using \s+ regex.

4. Length Filtering (≥50 words)
   Removes extremely short or low-information content (titles, ads, broken lines), keeping only meaningful text.

5. Deduplication (MD5 Hashing)
   To ensure uniqueness, MD5 hashes were computed from the first 2000 characters of each cleaned document. Documents with identical hashes were discarded.

Benefits of Cleaning

- Reduces noise and junk text
- Prevents the model from memorizing duplicates
- Improves the representativeness of token distributions
- Standardizes input format across heterogeneous sources

3. Tokenization Choices

Tokenization prepares raw text into tokenIDs for model consumption. The following choices were made:

Tokenizer
- GPT-2 Byte Pair Encoding (BPE)
  Loaded using AutoTokenizer from Hugging Face.

Vocabulary
- ~50,000 subword units
  (GPT-2 standard vocabulary size)

Special Token Decisions
- GPT-2 has no built-in PAD token
- EOS token was reused as the padding token to allow batch padding

Chunking Strategy (Block Size)
- Block size: 512 tokens
- All tokenized sequences were split into fixed-size blocks of max 512 tokens.

Why Block Size = 512?
- Memory-efficient during training
- Commonly used for mid-size transformer training
- Allows good context range while keeping batch sizes manageable

4. DataLoader & Implementation Details

A fully custom PyTorch dataloader was implemented to handle batching, padding, and attention mask generation.

Components

TokenBlockDataset

- Stores all token blocks as Python lists of token IDs.
- Implements standard __len__ and __getitem__.

Custom collate_fn

Handles:

- Dynamic padding to match the longest sequence in the batch
- Creation of the attention_mask
    1. 1 for real tokens
    2. 0 for padding tokens

PyTorch DataLoader

- batch_size = 8
- shuffle = True
- collate_fn injected via lambda
- Efficient batch assembly for training

Saved Output

A sample batch containing:

- input_ids
- attention_mask

was saved to sample_dataset.pt as required.

5. Challenges Encountered

1. Deprecated Dataset Scripts

The older "wikipedia" dataset is no longer supported in Hugging Face.
Solution: use wikimedia/wikipedia (Parquet-based, streaming-compatible).

2. Memory and Disk Constraints

Loading Wikipedia and FineWeb normally would require tens of gigabytes.
Solution: streaming mode so only small portions are kept in memory.

3. Deduplication Efficiency

Deduping millions of documents requires storing hashes.
Solution: hash only first 2000 characters → drastically reduces memory footprint.

4. Inconsistent Column Names
Different datasets use different field names:
Wikipedia: "text"

5. FineWeb: "text"
AG News: "text" or "description"
Solution: dynamic field detection.

6. Variable Sequence Length
Batches contained variable-length sequences.
Solution: custom collate function to handle padding and attention mask generation.

6. Reflections on Impact to Pretraining Quality

Preprocessing quality has an outsized impact on the performance, stability, and sample-efficiency of language model pretraining.

Key Reflections

1. Cleaner input = more stable training
   Noise, HTML, duplicates, and broken text harm convergence. Cleaning improves token distribution and reduces gradient noise.

2. Deduplication prevents memorization
   Language models easily memorize repeated documents. Deduplication ensures more efficient learning and better generalization.

3. Domain-balanced sampling improves robustness
   Mixing encyclopedic, web, and news text exposes the model to diverse syntax and styles.

4. Chunking at 512 tokens balances memory and context
   Larger blocks improve context understanding but reduce batch size; 512 is a good compromise.

5. Tokenizer compatibility is essential
   Choosing GPT-2's tokenizer ensures a stable, well-tested subword vocabulary.

Overall Conclusion

High-quality preprocessing is arguably as important as model architecture.
Clean, deduplicated, diverse data directly improves:

- Convergence speed
- Representation quality
- Downstream fine-tuning performance
- Generalization & robustness