Assignment 1 Report

1. Dataset Sources and Total Size

1.1 Dataset Source

In this assignment, we use the English Wikipedia dump as the primary data source for foundation model pretraining.

The dataset is accessed via the Hugging Face wikimedia/wikipedia repository, using the 20231101.en configuration.

Wikipedia is a publicly available, large-scale, and high-quality textual corpus that covers a wide range of domains, including politics, science, history, geography, and culture. Its encyclopedic nature makes it a commonly used dataset for pretraining large language models.

1.2 Dataset Size and Access Method

The full English Wikipedia corpus contains tens of millions of articles and has a total raw text size well exceeding 1 GB, satisfying the assignment requirement for large-scale data.

To efficiently handle the dataset without downloading it entirely to local storage, we utilize streaming mode provided by the Hugging Face datasets library. This approach enables scalable preprocessing while avoiding memory and disk bottlenecks.

2. Cleaning Strategies and Reasoning

2.1 Text Normalization

Raw text data often contains inconsistent formatting that can negatively affect model training. We apply the following normalization steps:

Whitespace normalization: Multiple consecutive whitespace characters are collapsed into a single space.

Lowercasing: All text is converted to lowercase to reduce vocabulary size and improve token consistency.

These steps standardize the input text while preserving its semantic content.

2.2 Low-Quality Document Filtering

Very short documents usually provide limited training value and may introduce noise.

We remove documents containing fewer than 50 words, which helps eliminate low-information samples such as stubs or incomplete entries.

2.3 Deduplication

Large-scale web-based corpora often contain duplicated or near-duplicated content. To address this issue, we implement hash-based deduplication:

Each cleaned document is hashed using the MD5 algorithm.

Documents with previously seen hashes are discarded.

This approach is lightweight, streaming-friendly, and effective at removing exact duplicates without requiring large memory overhead.

2.4 Design Rationale

The cleaning strategy is intentionally conservative. Instead of aggressively removing content, we focus on:

Preserving semantic information

Improving consistency

Removing obvious low-quality or redundant data

This balance helps maintain dataset diversity while improving overall data quality.

3. Tokenization Choices

3.1 Tokenizer Selection

We use a GPT-style Byte Pair Encoding (BPE) tokenizer, specifically the GPT-2 tokenizer provided by Hugging Face.

This tokenizer is well-suited for autoregressive language modeling and is widely used in foundation model pretraining.

Tokenizer type: GPT-style BPE

Vocabulary size: ~50,000 tokens

3.2 Tokenization Strategy

Tokenization is performed without truncation at the tokenizer stage. Instead, entire documents are first tokenized into token ID sequences.

To accommodate transformer input length constraints, long token sequences are divided into fixed-length blocks.

3.3 Block Size

We choose a block size of 512 tokens, which is commonly used in transformer-based pretraining and provides a balance between context length and computational efficiency.

Each block is treated as an independent training sample.

4. Data Loader Implementation

4.1 Dataset Design

To support large-scale streaming data, we implement a custom PyTorch IterableDataset.

This design allows token blocks to be generated on-the-fly without storing the entire dataset in memory.

4.2 Batching and Padding

A PyTorch DataLoader is used to batch token blocks:

Batch size: 8

Padding: Sequences within a batch are padded using the tokenizer's padding token

Output shape: (batch_size, block_size)

This setup ensures efficient batching while maintaining flexibility for variable-length sequences.

4.3 Memory Efficiency

The combination of streaming datasets, iterable loading, and on-the-fly tokenization allows the pipeline to scale to very large corpora without exceeding memory constraints.

5. Challenges Encountered

Several challenges were encountered during preprocessing:

Memory limitations: Loading large datasets entirely into memory is infeasible; streaming was required.

Deduplication trade-offs: Hash-based deduplication efficiently removes exact duplicates but does not detect near-duplicates.

Long documents: Wikipedia articles can exceed typical transformer input lengths, requiring explicit chunking.

Batching efficiency: Ensuring consistent batch shapes while maintaining streaming compatibility required careful dataset design.

6. Reflections on Preprocessing Impact

Effective preprocessing plays a crucial role in foundation model pretraining.

Cleaning and normalization improve token consistency, while deduplication reduces redundancy and prevents overfitting to repeated content.

Chunking long documents into fixed-length blocks ensures compatibility with transformer architectures and enables efficient training.

Overall, the preprocessing pipeline improves data quality, training stability, and computational efficiency, which are critical factors for large-scale language model pretraining.

7. Code Detail

For more detail about code and data sample, check url:

https://github.com/CodecNao/CSYE7374-Infrastructure-DevOps-w-LLMs/tree/main/Assignment1