Mid Evaluation Report

Name: `Ansh Chablani`
Roll No.: `2022111031`

Information

Language 1: `English`
Language 2: `Hindi`
Language 3: `Sanskrit`

Approach

Getting the Data

The data collection strategy was designed to meet the project's requirement of a 3 billion token corpus with a specified language distribution.

- **English and Hindi:** Given the availability of large web corpora, the `FineWeb` dataset was chosen as the primary source. Its focus on quality filtering makes it an excellent starting point for building a high-quality pretraining dataset.
- **Sanskrit:** To acquire a substantial and verified Sanskrit corpus, data was sourced from the **ai4bharat/sangraha** dataset available on the Hugging Face Hub. The verified texts from `https://huggingface.co/datasets/ai4bharat/sangraha/tree/main/verified/san` provided a diverse collection of sources, satisfying the requirement for the Indian language component of the dataset.

This collection strategy ensures a robust foundation for the model, balancing a large, high-quality web corpus with a specialized, verified corpus for the less-resourced language.

Preprocessing the Data

The raw text data was processed using a comprehensive pipeline (`tasks/01_preprocess_data.py`) to ensure quality and consistency. The key steps were:

1. **Text Cleaning:** Each line of text from all language files was individually cleaned. This involved:
    - Using the `ftfy` library to fix potential Unicode errors and inconsistencies (mojibake).
    - Normalizing all whitespace (tabs, newlines, multiple spaces) into a single space to create uniform sentences.
2. **Parallel Processing:** To handle the large volume of data efficiently, the cleaning process was parallelized to utilize all available CPU cores, significantly reducing the processing time.
3. **Global Deduplication:** To prevent the model from overfitting on repeated sentences, a two-step deduplication was performed. First, duplicates were removed within each language file. Then, all unique lines were combined into a single master set, which removed any duplicates that existed *between* the language files.

4. **Data Splitting:** The final, clean, and globally unique dataset was shuffled thoroughly to ensure a random distribution of languages. It was then split into three sets for the model lifecycle:

   - **Training Set:** 98%
   - **Validation Set:** 1%
   - **Test Set:** 1%

This process resulted in clean, non-redundant, and well-structured datasets saved in the `data/processed/` directory, ready for model training.

Tokeniser

A single, unified tokenizer was trained on the combined raw text of all three languages.

- **Choice of Tool: SentencePiece** was selected due to its language-agnostic nature. It operates directly on Unicode characters, making it ideal for handling datasets with different scripts (Latin for English, Devanagari for Hindi and Sanskrit) without requiring complex, language-specific pre-tokenization rules.

- **Algorithm:** The **Byte-Pair Encoding (BPE)** algorithm was used. BPE is effective at creating a subword vocabulary that can handle complex morphology and out-of-vocabulary words by breaking them down into smaller, known pieces.

- **Training Parameters:** The training was optimized for a large, multilingual corpus:

  - `vocab_size`: **32,000**, a standard size that provides a good balance between vocabulary richness and model complexity.
  - `character_coverage`: **1.0**, ensuring all characters from all three languages were included in the base vocabulary.
  - `input_sentence_size`: **10,000,000**, a crucial memory-efficiency parameter. Instead of loading all ~47 million sentences into RAM (which causes a crash), the trainer samples a large, random subset, which is sufficient to build a high-quality vocabulary.
  - `num_threads`: Dynamically set to use all available CPU cores to maximize training speed.

The resulting tokenizer files (`multilingual_spm.model` and `multilingual_spm.vocab`) are saved in the `model/tokenizer/` directory.

Evaluating the Tokeniser

The tokenizer was evaluated to ensure it supports all three languages adequately, a critical requirement for a fair multilingual model. The evaluation was two-fold (`evaluate/02_evaluate_tokenizer.py`):

1. **Quantitative Analysis (Token Coverage):** The primary evaluation metric was token coverage, calculated as `100% - (percentage of unknown '<unk>' tokens)`. This was measured **independently for each language** by tokenizing a large sample (500,000 lines) of each raw text file.

   Results:

   - **English Coverage:** `99.9993`%
   - **Hindi Coverage:** `99.9998`%
   - **Sanskrit Coverage:** `100.0`%

The high and comparable coverage rates across all three languages confirm that the vocabulary is well-balanced and does not favor one language over the others.

2. **Qualitative Analysis (Vocabulary Inspection):** A manual inspection of the `multilingual_spm.vocab` file was performed. Samples from the beginning, middle, and end of the vocabulary list showed a healthy mix of English subwords (e.g., " and", "ation"), Devanagari characters and subwords (e.g., " ार", " कि"), and transliterated tokens. This provides strong anecdotal evidence that the tokenizer successfully learned a shared representation for the multilingual corpus.
