Final Report

Language, Model and Agents Mini-Project

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

Specific Information

Languages:

1. `English`
2. `Hindi`
3. `Sanskrit`

Model: `Qwen3-Dense`

Tasks:

1. Fact Decontextualization (Knowledge & Commonsense Reasoning)
2. Identify and replace nouns with verbs from given sentence (Sentence Structure & Syntax)

Dataset Collection

I had to extract the data for the 3 languages English, Hindi and Sanskrit

Fortunately, I was able to get the data for these three languages because of their popularity.

Sources:

1. English -> FineWeb
2. Hindi -> Fineweb
3. Sanskrit -> AI4Bharat-Sangraha

These three data sources were enough to provide sufficient sources.

The token counts for the above languages are:

English: `1697.12M`
Hindi: `540.52M`
Sanskrit: `216.50M`

The file sizes of all the languages are:

1. English: `6589.23 MB`
2. Hindi: `4561.78 MB`
3. Sanskrit: `1520.55 MB`

Number of Sentences/Lines:

1. English: `35309702`

2. Hindi: `9386954`
3. Sanskrit: `1951518`

The file for dataset gathering is: tasks/00_get_data.py.

Preprocessing

I applied several preprocessing steps to enhance the quality of my 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%
5. **Script and Language Filtering:** Each line underwent an additional layer of filtering to ensure that the text matched the intended script and language. This involved:
    - **Regex-based script detection:** Lines were quickly pre-filtered using Unicode ranges — Latin characters for English, Devanagari for Hindi and Sanskrit.
    - **Language identification:** The `langdetect` library was applied to verify that the cleaned line belonged to the expected language (e.g., `en` for English, `hi` for Hindi, `sa` or `hi` for Sanskrit, accounting for common misclassifications).
    - **Noise reduction:** Lines with too many symbols or mixed scripts were discarded.

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

Tokenisation

I trained a **SentencePiece** tokenizer optimized for handling English, Hindi, and Sanskrit text.

The key steps were:

1. **Multilingual Training Data:** The raw text files for English, Hindi, and Sanskrit were used together to train a shared tokenizer. This ensured that all three languages could be encoded consistently with a single vocabulary.

2. **SentencePiece with BPE:** I used the SentencePiece library with the **Byte Pair Encoding (BPE)** algorithm.

    - Vocabulary size: **320,003 tokens**
    - Character coverage: **100%**, ensuring that all Unicode characters from the training set were represented.
    - Special tokens were explicitly reserved:
        - `<pad>` → ID 0
        - `<unk>` → ID 1
        - `<bos>` → ID 2
        - `<eos>` → ID 3
    - **Custom user-defined symbols were added for each language tag: `<en>`, `<hi>`, `<sa>`.**

3. **Efficient Training:**

    - Leveraged all available CPU cores (`num_threads = os.cpu_count()`) for parallel training.
    - Sampled up to **10 million sentences** with shuffling to ensure balanced representation of each language.

4. **Integration with Hugging Face:**

    - After training, the `.model` and `.vocab` files from SentencePiece were converted into a **Hugging Face–compatible tokenizer** (`LlamaTokenizer`).
    - The tokenizer was saved in `model/tokenizer/`, producing the standard Hugging Face files:
        - `tokenizer.json`
        - `tokenizer_config.json`
        - `special_tokens_map.json`

5. **Testing:**
   The trained tokenizer was tested on sample sentences in all three languages to verify correct tokenization:

    - English: `"This is a test sentence in English."`
    - Hindi: `"यह हिंदी में एक परीक्षण वाक्य है।"`
    - Sanskrit: `"एषः संस्कृतस्य परीक्षणवाक्यम् अस्ति।"`

Why `<en>`, `<hi>`, `<sa>`?

Instead of simply mixing English, Hindi, and Sanskrit lines into the dataset, I introduced **explicit language tags** (`<en>`, `<hi>`, `<sa>`) at the tokenizer level.

- **Disambiguation:** These tags give the model a clear signal about the language of the current input. Without tags, the model might confuse closely related scripts (e.g., Hindi vs Sanskrit, which are often misclassified).

- **Better Control:** During inference, I can *force* the model to generate text in a specific language by prefixing the input with `<en>`, `<hi>`, or `<sa>`.
- **Multilingual Generalization:** Language tags have been shown in multilingual models (e.g., mBART, XLM-R) to improve cross-lingual alignment and reduce code-switching errors.
- **Innovation vs Naive Mixing:** A naive approach would just concatenate all multilingual lines and let the model figure out the language implicitly. By explicitly marking the language at the tokenizer level, I give the model structured guidance, which improves both training efficiency and generation quality.

This design choice is an **innovation** in my pipeline: instead of treating multilingual data as an unstructured mixture, I explicitly encode language identity as a token. This not only improves training stability but also provides controllability during downstream tasks (translation, classification, or generation).

Pretraining

For pretraining my multilingual model, I explored **two different strategies**:

---

1. Simple Training (Baseline)

The first approach was a **straightforward pretraining pipeline** using the Hugging Face `Trainer` API:

- **Architecture:** A custom Qwen3-based model (~125M parameters) configured from scratch.
- **Tokenizer:** The multilingual SentencePiece tokenizer (`<en>`, `<hi>`, `<sa>` support) trained earlier.
- **Dataset:** The cleaned and deduplicated text corpus across English, Hindi, and Sanskrit.
- **Training setup:**
  - Maximum sequence length: 2048 tokens
  - Training epochs: 3
  - Cosine learning rate schedule with warmup
  - Gradient checkpointing for memory efficiency
  - Mixed precision (`fp16` or `bf16`) to speed up training

This method directly optimized the model on the raw dataset using cross-entropy loss. It produced a **baseline multilingual model** that was consistent but relatively large and resource-intensive.

---

2. Knowledge Distillation (Student–Teacher Training)

To make the model lighter while retaining performance, I used **knowledge distillation**:

- **Teacher Model:** A larger pretrained checkpoint (producing logits for each token).
- **Student Model:** A smaller model initialized with the same architecture family, but with fewer parameters.
- **Distillation Loss:** A weighted combination of:
  - **Hard targets** → Cross-entropy with ground-truth labels.
  - **Soft targets** → KL-divergence between teacher and student probability distributions, scaled by a temperature parameter.
- **Chunked Training:** Teacher logits were precomputed and stored in chunks (`distillation_teacher_logits_chunk_*.pt`). The student model iterated over these chunks efficiently without re-running the teacher forward pass.

- **Checkpointing:** Student checkpoints were saved regularly to support resumption and avoid loss of progress.

Results from Distillation

The **distilled student model** achieved:

- **Smaller size:** Significantly fewer parameters compared to the baseline model, reducing memory footprint and inference cost.
- **Faster inference:** Improved latency, making the model more practical for deployment on resource-constrained devices.
- **Comparable quality:** Retained most of the language understanding and generation capabilities of the baseline model, thanks to the teacher's soft supervision.
- **Better stability:** The combination of hard labels and softened teacher distributions improved convergence and reduced overfitting.

In summary:
I first pretrained a baseline multilingual Qwen3 model from scratch for performance benchmarking. Then, I trained a distilled student model that was smaller and faster but still competitive in quality. This two-step process provided both a **high-quality reference model** (teacher) and a **lightweight deployable model** (student).

Finetuning

Dataset Creation - Task 1

To fine-tune the model on a **fact decontextualization** task, I created a custom dataset using a large language model (LLM).
The goal was to train the model to convert a **contextualized sentence** into a **standalone atomic fact**.

Dataset Generation Pipeline

1. **Sentence Generation:**
   Using Gemini (`gemini-2.5-flash`), I prompted the model to generate diverse, contextualized sentences containing explicit or implicit factual content.

   - Example prompt: *"Generate 10 diverse, complex, and contextualized English sentences that contain explicit or implicit facts..."*
   - This was repeated separately for **English** and **Hindi** datasets.

2. **Fact Extraction:**
   For each sentence, I asked Gemini to extract a single **decontextualized fact** that:

   - Resolves pronouns.
   - Can stand alone without additional context.
   - Is concise and factual.

   Few-shot examples were included in the prompt to guide Gemini's outputs.

3. **Human-in-the-loop Review:**
   Each Gemini-suggested fact was reviewed. The script allowed three options:
   - **Accept** → keep Gemini's suggestion.
   - **Edit** → modify Gemini's fact.
   - **Reject** → manually enter a corrected fact.
     For the initial dataset, I auto-accepted all suggestions (`review_choice = 'Y'`), but the script supports manual review for higher quality.

4. **JSONL Output:**
   The final dataset was saved in `.jsonl` format, where each entry consists of:

```
{
  "input": "CEO John Smith announced record profits at the meeting.",
  "output": "John Smith is a CEO."
}
```

Why This Approach?

Synthetic but diverse: Leveraging an LLM allowed me to generate diverse, domain-rich sentences without needing a large annotated dataset.

Decontextualization Task-Specific: Instead of generic QA pairs, the dataset directly targeted the skill I wanted my model to learn: rewriting context-rich sentences into standalone factual statements.

Bilingual Extension: The same pipeline was applied to Hindi, ensuring that the fine-tuned model could handle decontextualization across multiple scripts and languages.

To enhance the model's capabilities, I fine-tuned it on two specific tasks.

Finetuning Process - Task 1

For the first task, fact decontextualization, I employed Parameter-Efficient Fine-Tuning (PEFT) using Low-Rank Adaptation (LoRA). This approach is highly efficient as it avoids retraining the entire model. Instead, it freezes the pretrained model weights and injects trainable "adapter" layers.

The key steps in this process were:

1. **Model Loading:** The pretrained multilingual model was loaded, along with its corresponding tokenizer.
2. **LoRA Configuration:** I configured LoRA to target the query and value projection layers (`q_proj`, `v_proj`) of the model's attention mechanism. This is a common practice as these layers are crucial for how the model weighs and interprets input tokens. The rank (`r`) of the LoRA matrices was set to 16, providing a good balance between expressiveness and parameter efficiency.
3. **Prompt Templating:** A standardized prompt template was created to structure the input for the model. This template clearly separates the instruction, the input sentence, and the expected response, which helps the model learn the task more effectively.

4. **Dataset Preparation:** The previously generated `.jsonl` files for English and Hindi were loaded into a custom PyTorch Dataset. A crucial step here was to mask the input part of the prompt in the loss calculation. This ensures that the model is only trained to predict the `output` (the decontextualized fact) and not the instruction or input sentence, leading to more efficient learning.
5. **Training with PEFT:** The model was trained for 20 epochs using the AdamW optimizer. The PEFT library seamlessly integrated the LoRA adapters into the model, and I was able to train it using a standard PyTorch training loop. The number of trainable parameters was a small fraction of the total model size (around 0.08%), which significantly reduced the computational resources required.
6. **Saving Adapters:** After training, only the LoRA adapter weights were saved, not the entire model. This is a key advantage of PEFT, as these adapters are small and can be easily shared and loaded on top of the base pretrained model for inference.

This PEFT-based finetuning process allowed for efficient and effective adaptation of the pretrained model to the specific task of fact decontextualization.

Dataset Creation - Task 2

For the second task, "Identify and replace nouns with verbs from a given sentence," I generated two separate datasets for English and Hindi.

English Dataset:

1. **Corpus:** I used the classic "Sense and Sensibility" text from the NLTK Gutenberg corpus as the source of grammatically correct English sentences.
2. **Sentence Tokenization:** The raw text was first cleaned to normalize whitespace and then split into individual sentences using `nltk.sent_tokenize`.
3. **POS Tagging:** The `spaCy` library was used to perform Part-of-Speech (POS) tagging on each sentence to identify all nouns (`NOUN`, `PROPN`) and verbs (`VERB`).
4. **Noun-Verb Swapping:** For each sentence containing both nouns and verbs, a random number of noun-verb pairs were selected for swapping. The selected nouns and verbs were then shuffled and swapped to create a syntactically altered but grammatically plausible new sentence.
5. **Data Cleaning:** A post-processing step was applied to clean up any artifacts from the automated swapping, such as extra spaces before punctuation.
6. **Output Format:** The final dataset, consisting of 5000 samples, was saved in a JSON file. Each entry included a standard instruction, the original sentence (`input`), and the modified sentence (`output`).

Hindi Dataset:

A similar process was followed for Hindi, with adjustments for the language:

1. **Corpus:** A large raw text file of Hindi sentences (`lang_hindi.txt`) was used as the source. To handle the large file size without excessive memory usage, the script was designed to read the file line by line.
2. **NLP Pipeline:** The `Stanza` library, which offers robust support for Hindi, was used for sentence segmentation and POS tagging.
3. **Swapping Logic:** The core noun-verb swapping logic remained the same. Stanza's word objects were used to identify nouns and verbs, and a new sentence was constructed by swapping their text.
4. **Instruction:** The instruction was provided in Hindi ("दिए गए वाक्य में संज्ञा को क्रिया से और क्रिया को संज्ञा से पहचानें और बदलें।").

5. **Output:** 5000 samples were generated and saved in a separate JSON file, following the same structure as the English dataset.

This dual-language dataset creation ensures that the model is trained to perform the noun-verb swap task in both English and Hindi.

Finetuning Process - Task 2

The finetuning for the second task followed a similar PEFT with LoRA approach as the first task, building on the same pretrained model.

1. **Model and Tokenizer:** The same base pretrained model and tokenizer were used.
2. **Dataset Loading:** A new PyTorch Dataset class was created to load the JSON files for the noun-verb swap task. This class was designed to handle the list-of-dictionaries format of the new dataset.
3. **Prompt Template:** A flexible prompt template was used, which could accommodate both the English and Hindi instructions directly from the data files.
4. **LoRA Configuration:** The LoRA configuration (rank, alpha, target modules) was kept consistent with the first finetuning task to maintain a standardized approach.
5. **Training:** The model was trained for 20 epochs with a batch size of 24. As before, the prompt portion of the input was masked from the loss calculation, focusing the training on generating the correct modified sentence.
6. **Saving Adapters:** The newly trained LoRA adapters specific to the noun-verb swap task were saved to a separate directory. This modular approach allows for keeping different sets of task-specific adapters that can be loaded onto the base model as needed.

By using the same PEFT methodology, I was able to efficiently train the model on a second, distinct task, further enhancing its versatility and demonstrating the power of using trainable adapters for multi-task learning.

Results (See full results in `outputs folder`)

Pretraining

Example 1

```
--- Generating for English ---
Prompt: <en>Once upon a time

Cleaned Generated Text:
Once upon a time to you think the world. This is to be on the home, and it
was there's just we want to get. But's have about the best of the same
time. I will have an only for the most other week, but

--- Generating for Hindi ---
Prompt: <hi>एक समय की बात है

Cleaned Generated Text:
एक समय की बात है, मैं इस मुझे नकर में और तो यह को ही नहीं हो। आप अपने से भी कुछ है! एक के
```

```
हर कम था कि इससे फिर अपने दो दी जाता था। |2||

--- Generating for Sanskrit ---
Prompt: <sa>कदाचित्कालः आसीत्

Cleaned Generated Text:
कदाचित्कालः आसीत् ।
```

Example 2

```
--- Generating for English ---
Prompt: <en>Once upon a time

Cleaned Generated Text:
Once upon a time, and the same of the first is so more like in me. If
you're to be able to go for your good to have been only just

--- Generating for Hindi ---
Prompt: <hi>एक समय की बात है

Cleaned Generated Text:
एक समय की बात है। (५) तयंत् । 'अत्रं लति' इति 'अमस्य' इत्ययमिति । नापि - 'तथम्' न चः म्यामा,
तद्धिपिकः ।

--- Generating for Sanskrit ---
Prompt: <sa>कदाचित्कालः आसीत्

Cleaned Generated Text:
कदाचित्कालः आसीत् । अथ चैव प्राहना हि न गन्तत इति शङ्कति, तदेणिकानि, ते ससन्गवय एव वयाम्, अचे
मते समल्वस्य विहति ।
```

Example 3

```
--- Generating for English ---
Prompt: <en>Once upon a time

Cleaned Generated Text:
Once upon a time of the most is not you or was have to go? For no way, it
can be do like out. It was the people were as I can't tell me that the good
with the number’s in an other-the

--- Generating for Hindi ---
Prompt: <hi>एक समय की बात है

Cleaned Generated Text:
एक समय की बात है कि पर 'अधुजस्य' ।

--- Generating for Sanskrit ---
Prompt: <sa>कदाचित्कालः आसीत्
```

```
Cleaned Generated Text:
कदाचित्कालः आसीत् । ननु तस्य स्वस्य वा साऽर्थे तु यथा न तक्षौ गावादकस्तां पूर्वं जवागते । यन्ताहयापताति समन्
पिपि भुते 'मदती
```

Finetuning

Task 1

Example 1

```
--------------------
Original English Sentence:
She enjoys painting landscapes in her free time

Extracted Fact:
The first employees are a best of United disease.
--------------------
--------------------
Original Hindi Sentence:
ताजमहल, जो भारत के आगरा शहर में स्थित एक हाथीदांत-सफेद संगमरमर का मकबरा है, को मुगल सम्राट
शाहजहाँ ने अपनी पसंदीदा पत्नी मुमताज महल की याद में बनवाया था।

Extracted Fact:
चोजना मनिपा का एक वविमला एक साथ ही दीता है।
--------------------
--------------------
Original Custom Sentence:
The mall, which opened in 2005, is the largest shopping center in the city.

Extracted Fact:
The book is a new way of the world.
--------------------
```

Example 2

```
--------------------
Original English Sentence:
She enjoys painting landscapes in her free time

Extracted Fact:
The first lot of first than king product.
--------------------
--------------------
Original Hindi Sentence:
ताजमहल, जो भारत के आगरा शहर में स्थित एक हाथीदांत-सफेद संगमरमर का मकबरा है, को मुगल सम्राट
शाहजहाँ ने अपनी पसंदीदा पत्नी मुमताज महल की याद में बनवाया था।
```

```
Extracted Fact:
कोसिसिया ने एक बबलेस में देश में भी खँगरे के लिए एक वशारकह के आति-निसर के साथ पिनक हैं।
--------------------
--------------------
Original Custom Sentence:
The mall, which opened in 2005, is the largest shopping center in the city.

Extracted Fact:
The park has a new best book.
--------------------
```

Example 3

```
--------------------
Original English Sentence:
She enjoys painting landscapes in her free time

Extracted Fact:
The first love need the first employees.
--------------------
--------------------
Original Hindi Sentence:
ताजमहल, जो भारत के आगरा शहर में स्थित एक हाथीदांत-सफेद संगमरमर का मकबरा है, को मुगल सम्राट शाहजहाँ ने अपनी पसंदीदा पत्नी मुमताज महल की याद में बनवाया था।

Extracted Fact:
जविलल विचत के खिलाफ एक ही देश है।
--------------------
--------------------
Original Custom Sentence:
The mall, which opened in 2005, is the largest shopping center in the city.

Extracted Fact:
The law will be me the world's largest world.
--------------------
```

Task 2

Example 1

```
Instruction: Identify and swap nouns with verbs, and verbs with nouns, in the given sentence.
Input: The quick brown fox jumps over the lazy dog.
Model Output: The right world a Cennated then.

--------------------------------------------------
Instruction: दिए गए वाक्य में संज्ञा को क्रिया से और क्रिया को संज्ञा से पहचानें और बदलें।
Input: राजा ने लाओ-त्ज़ु के बताये अनुसार प्रशासन में सुधार किये और धीरे-धीरे उसका राज्य आदर्श राज्य बन गया.
```

```
Model Output: बं ने कुायों के यह है तब काम में है और आप-सी मुझे बात भारत पहले कर एक भारत
कायली गया.

--------------------------------------------------
Instruction: Identify and swap nouns with verbs, and verbs with nouns, in
the given sentence.
Input: His name, he replied, was Willoughby, and his present home was at
Allenham.
Model Output: His do, he like, was Marianne, and his own well was at first
of.

==================================================
```

Example 2

```
Instruction: In the given sentence, find nouns and replace them with verbs,
and replace verbs with nouns.
Input: The tall giraffe eats leaves from the highest branch.
Model Output: Theers Iram Marianne from the same mind.

--------------------------------------------------
Instruction: दिए गए वाक्य में संज्ञा को क्रिया से और क्रिया को संज्ञा से बदलें।
Input: विद्यार्थी ने परीक्षा के लिए कठिन परिश्रम किया और अच्छे अंक प्राप्त किये।
Model Output: यहिया ने करने के लिए लोगों परासिक भी और कुछ है।

--------------------------------------------------
Instruction: In the given sentence, find nouns and replace them with verbs,
and replace verbs with nouns.
Input: The artist painted a beautiful landscape, which later hung in the
grand hall.
Model Output: The also young the also came, which then child in the use.

==================================================
```

Example 3

```
Instruction: In the given sentence, find nouns and replace them with verbs,
and replace verbs with nouns.
Input: The tall giraffe eats leaves from the highest branch.
Model Output: Thely Hash-s Mrs. that the own love of other man.

--------------------------------------------------
Instruction: दिए गए वाक्य में संज्ञा को क्रिया से और क्रिया को संज्ञा से बदलें।
Input: विद्यार्थी ने परीक्षा के लिए कठिन परिश्रम किया और अच्छे अंक प्राप्त किये।
Model Output: यमा ने किया के लिए लिया पररमा कोबले और करने आप किया।

--------------------------------------------------
Instruction: In the given sentence, find nouns and replace them with verbs,
and replace verbs with nouns.
```

```
Input: The artist painted a beautiful landscape, which later hung in the
grand hall.
Model Output: The than those and Elinoring in the end use, which all child
in the have.

================================================
```

Metrics

Data and Tokenizer Statistics

```
Found train file for english:
/home/skills/ansh/dataset_gen/delme/ds_gen/data/processed/train_english.txt
Tokenizing train_english.txt: 35309702 lines [26:55, 21860.26 lines/s]
Found train file for hindi:
/home/skills/ansh/dataset_gen/delme/ds_gen/data/processed/train_hindi.txt
Tokenizing train_hindi.txt: 9386954 lines [09:19, 16777.74 lines/s]
Found train file for sanskrit:
/home/skills/ansh/dataset_gen/delme/ds_gen/data/processed/train_sanskrit.tx
t
Tokenizing train_sanskrit.txt: 1951518 lines [02:46, 11730.45 lines/s]


--- EVALUATION REPORT ---
{
  "Model Statistics": {
    "Total Parameters": "166.16M",
    "Trainable Parameters": "166.16M",
    "Model Architecture": "qwen3",
    "Number of Layers": 12,
    "Hidden Size": 768,
    "Number of Attention Heads": 12,
    "Vocabulary Size (from model config)": 32003
  },
  "Tokenizer": {
    "Class": "LlamaTokenizerFast",
    "Vocabulary Size": 32003
  },
  "Dataset (Train Split)": {
    "English": {
      "File Size (MB)": "6589.23",
      "Number of Sentences/Lines": 35309702,
      "Number of Tokens": "1697.12M"
    },
    "Hindi": {
      "File Size (MB)": "4561.78",
      "Number of Sentences/Lines": 9386954,
      "Number of Tokens": "540.52M"
    },
    "Sanskrit": {
```

```
      "File Size (MB)": "1520.55",
      "Number of Sentences/Lines": 1951518,
      "Number of Tokens": "216.50M"
    }
  }
}
```


Perplexity Calculation

```
--- Starting Perplexity Calculation ---

Evaluating on English test set...
Calculating PPL for english: 100%|
████████████████████████████████████████████████████████████████████
██████████| 33172/33176 [08:14<00:00, 67.04it/s]

Evaluating on Hindi test set...
Calculating PPL for hindi: 100%|
████████████████████████████████████████████████████████████████████
██████████| 10626/10630 [02:38<00:00, 67.05it/s]

Evaluating on Sanskrit test set...
Calculating PPL for sanskrit: 100%|
████████████████████████████████████████████████████████████████████
██████████| 4231/4235 [01:03<00:00, 66.99it/s]

--- Perplexity Results ---
{
  "Perplexity (English)": 305.4585571289062,
  "Perplexity (Hindi)": 205.635986328125,
  "Perplexity (Sanskrit)": 192.0955200195312,
  "Perplexity (Overall)": 268.9
}

--- Generating Qualitative Examples ---
Device set to use cuda:0

--- Language: English ---
Prompt: <en>Once upon a time in a land of algorithms,

Cleaned Generated Text:
Once upon a time in a land of algorithms, the most of the first are not be
able to get you. I have find that their other year or can like to be out
about he did this as the home and that's not too of the world on his new
school

--- Language: Hindi ---
Prompt: <hi>एक समय की बात है, डिजिटल दुनिया में

Cleaned Generated Text:
एक समय की बात है, डिजिटल दुनिया में अबकरक के साथ सकती। - " – : “ ””
```

```
--- Language: Sanskrit ---
Prompt: <sa>कदाचित्कालः आसीत्, यत्र संगणकाः

Cleaned Generated Text:
कदाचित्कालः आसीत्, यत्र संगणकाः, तदवौ । - ↑ अप्रिकमश्यादस्य "तस्म्रति" । 'अधुभं' इति सर्वभूतम् ।

--- Evaluation Complete ---
```

Dataset and Checkpoints

Find all of them at: Dataset and Checkpoints

More at: https://anshium.github.io/minimalist/projects/courses/lma/mini-project