PrefixSmart: Enhancing Large Language Model Efficiency through Advanced Prompt Management

Yunding Li, Kexin Chu, Wei Zhang
University of Connecticut

Nannan Zhao
Northwestern Polytechnical University

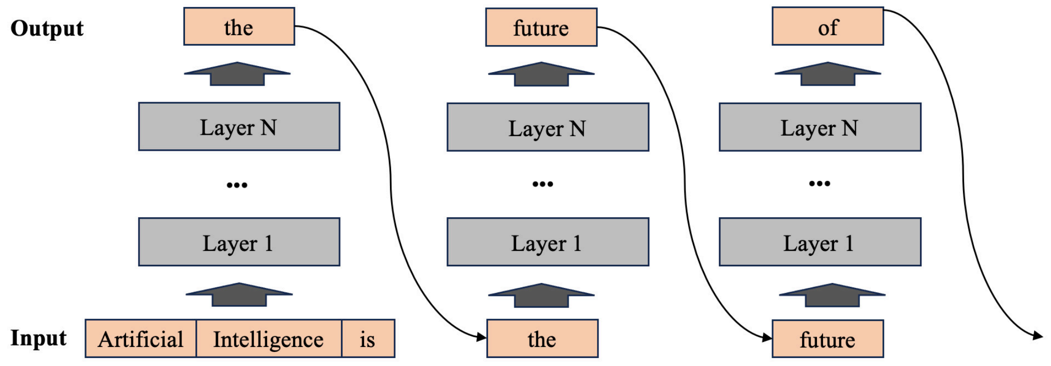
CONTENTS

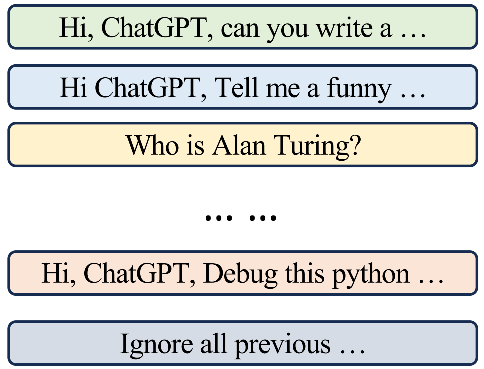
01

Intruduction of LLM and Motivation

Introductions of LLM Inference

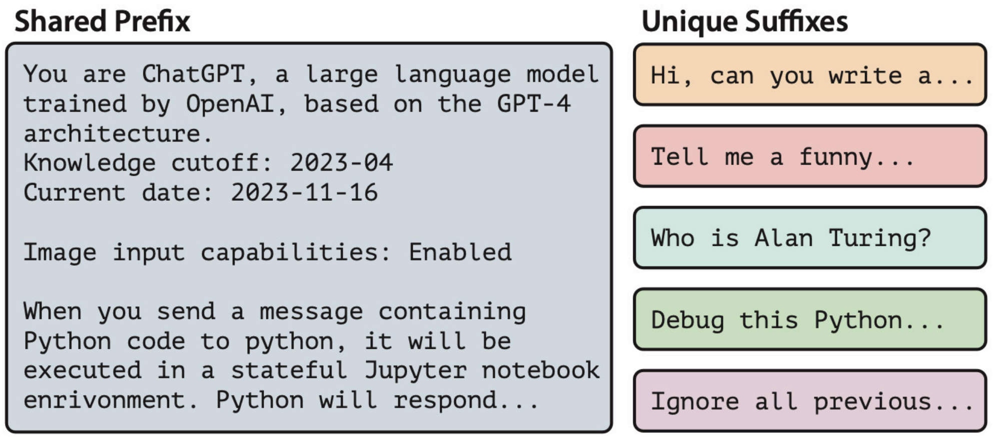
- Two stages:
  - Prefill: first iteration, compute all input tokens in a single pass
  - Decode: utilize previous generated token as input
- Regressive generation
- Use Batches of sequences to improve GPU utilization.




Hi, ChatGPT, can you write a …
Hi ChatGPT, Tell me a funny …
Who is Alan Turing?
… …
Hi, ChatGPT, Debug this python …
Ignore all previous …

Motivation of PrefixSmart

- All the tokens within a batch of requests is processed in prefill stage, even with duplicate tokens.
- In some scenarios, there exists similar prefix tokens, such as few-shot aibot applications.
- These common prefix tokens can only be execute once and shared by all the requests.

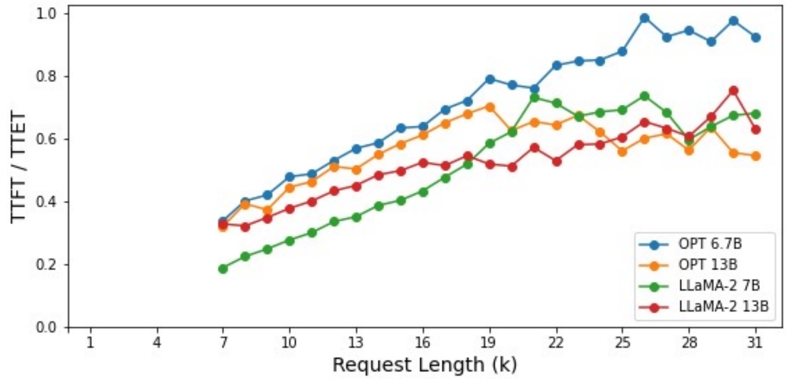


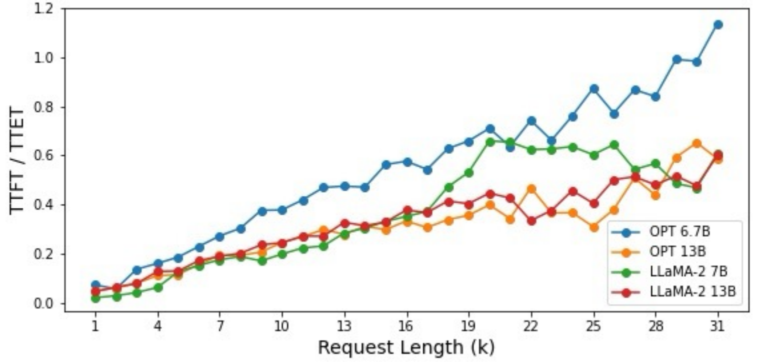
```
You are ChatGPT, a large language model
trained by OpenAI, based on the GPT-4
architecture.
Knowledge cutoff: 2023-04
Current date: 2023-11-16

Image input capabilities: Enabled

When you send a message containing
Python code to python, it will be
executed in a stateful Jupyter notebook
enrivonment. Python will respond...
```

```
Hi, can you write a...
Tell me a funny...
Who is Alan Turing?
Debug this Python...
Ignore all previous...
```

Prefill Latency vs End-to-End Latency

- The computational overhead during the prefill stage increases with the length of the prompts





Challenges of PrefixSmart

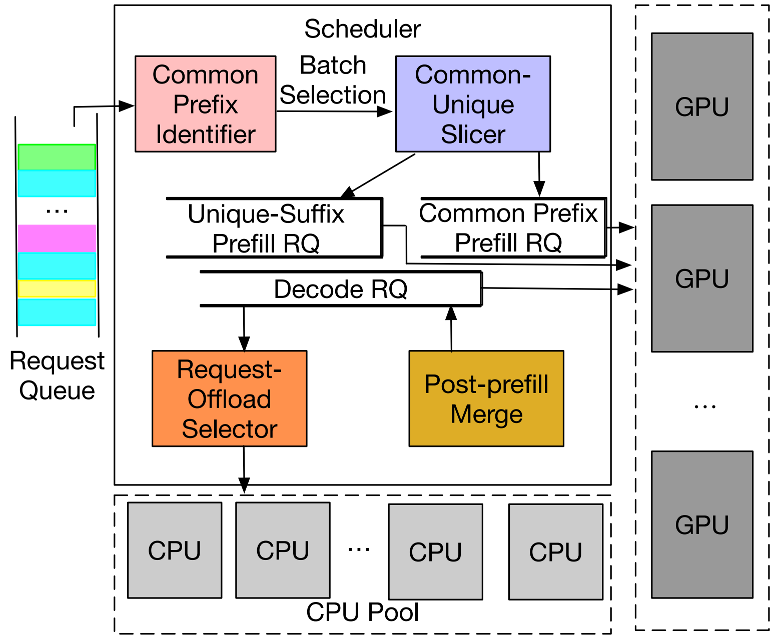
- How to detect the common prefix tokens efficiently.
    - Get shared prefix tokens
    - Group the requests with similar common prefix tokens.
- How to deal the interference from unequal-length suffixes
    - The prefill latency of different length prompt is different.
- Strategic offloading to CPUs
    - GPU memory is limited, the preemption will happen when GPU memory is not enough.
    - Use CPU rather than waiting for GPU resources.

02

Design Details of PrefixSmart

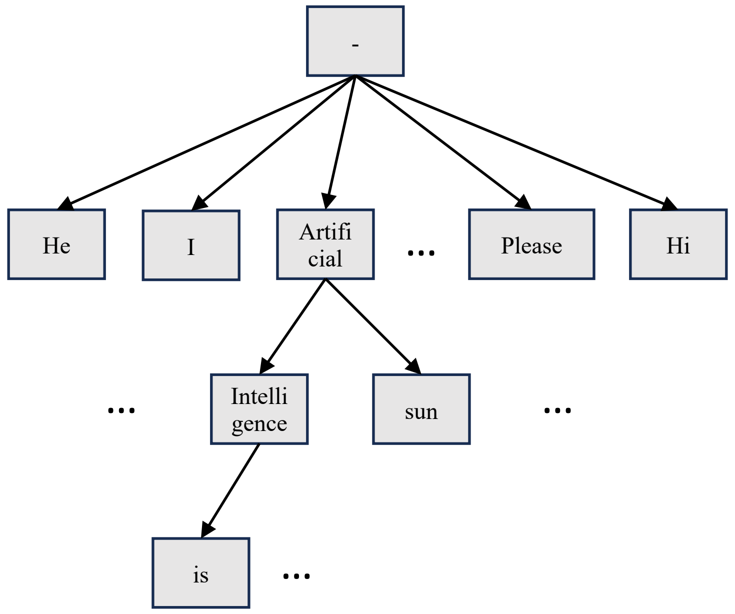
Overview of PrefixSmart

- Common-prefix Identifier
- Common-Unique Slicer
- Post-prefill merger
- Request-offload Selector



Common-prefix Identifier

- Efficiently detect shared prefixes in a large number of input prompts.
- Adopting a trie tree approach called radix tree for detecting shared prefixes within vast numbers of incoming requests.
- Prefix Radix Tree
  - Traditional trie tree can become unwieldy with deep and wide branches.
  - Compacting common phrases and frequently used word.
- Dynamic Pruning Strategy
  - Dividing the prefix tokens into ”Hot” and ”Cold” based on the frequency of them.
  - The “Cold” prefixes are actively pruned by the system.



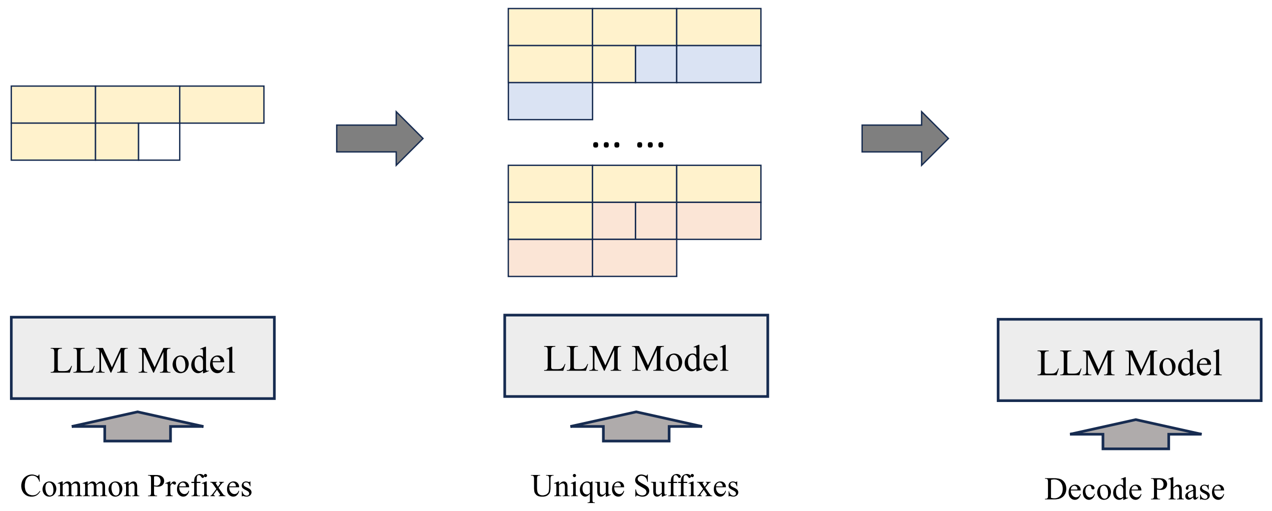
Common-Unique Slicer

- Intelligently segments prompts into shared prefix and unique suffix components.
  - Shared Prefix Trunk: Initial tokens shared across multiple prompts, allowing for single-time computation and reuse.
  - Unique Suffix Trunks: Distinct suffixes requiring individual processing for each prompt

- The shared prefix trunk and unique suffix trunks are assigned to different iterations of LLM inference.

Post-prefill Merger

- Integrates processed shared prefixes and unique suffixes into a coherent final output.
- Common-prefix Slicer: Dividing prompts into manageable parts
- Storage the KV cache through PagedAttention method



Request-offload Selector

- Offloading ‘Cold’ or less computation-intensive tasks to CPU and Host Memory.
- Frees GPU resources for more critical and ‘Hot’ taks, especially when GPU resources is unavailuable.
- The related KV cache should be offloaded.

03

Discussion and Future Works

Discussion and Future Works

- PrefixSmart improve the resource efficiency by addressing computation redundancies in the prefill stage.
- Future Works:
  - Optimization of Long-Context Prompt Handling
    - Use lightweight neural network model
    - Segment long-context prompts into smaller
  - Refinement of Prompt Slicing and Scheduling Algorithms

Thanks