中山大學
SUN YAT-SEN UNIVERSITY

ARSYSU

EFIM: Efficient Serving of LLMs for Infilling Tasks with Improved KV Cache Reuse

Tianyu Guo, Hande Dong, Yichong Leng, Feng Liu, Cheater Lin, Nong Xiao and Xianwei Zhang

Email: guoty9@mail2.sysu.edu.cn

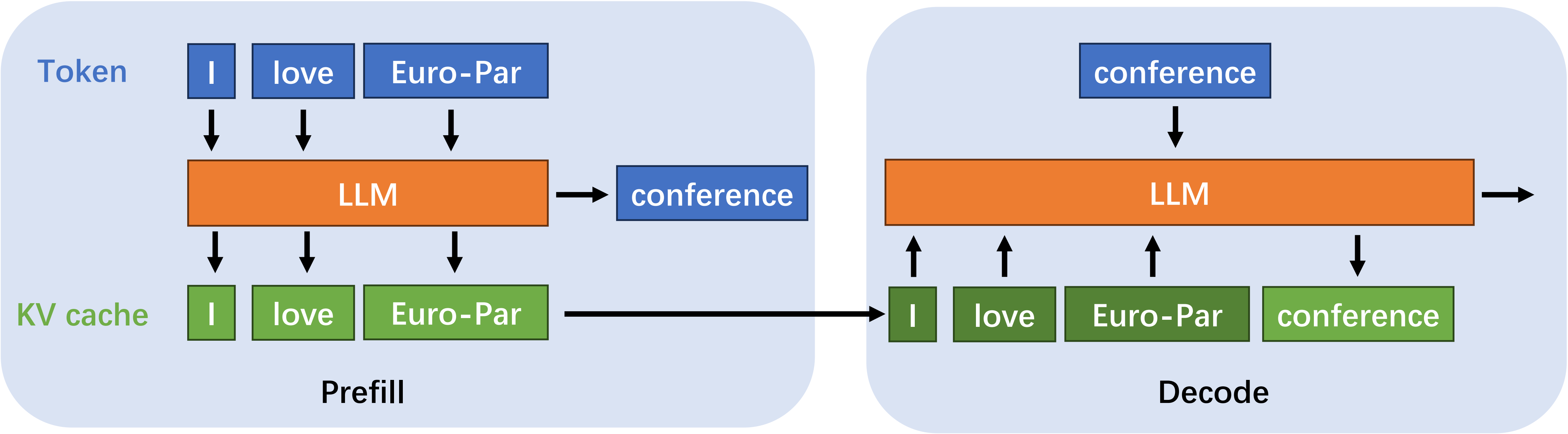
Time: 16:20 - 16:40, August 28th, 2025

My Homepage

EURO-PAR
CONFERENCE 2025

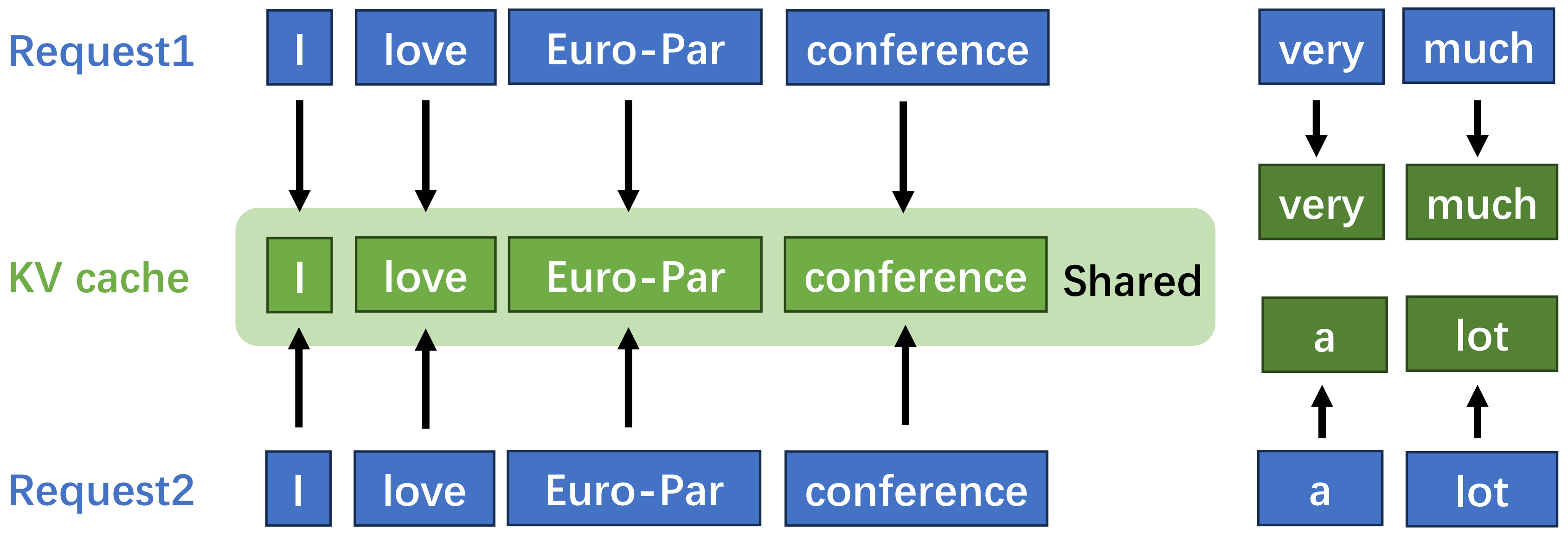
LLMs Inference Procedure & KV Cache

- LLM Inference: Autoregressive Decoding with KV cache
  - Decoding: Next token prediction based on previous tokens
  - Autoregressive: Generate token one by one
  - KV cache: Intermediate data kept for decoding



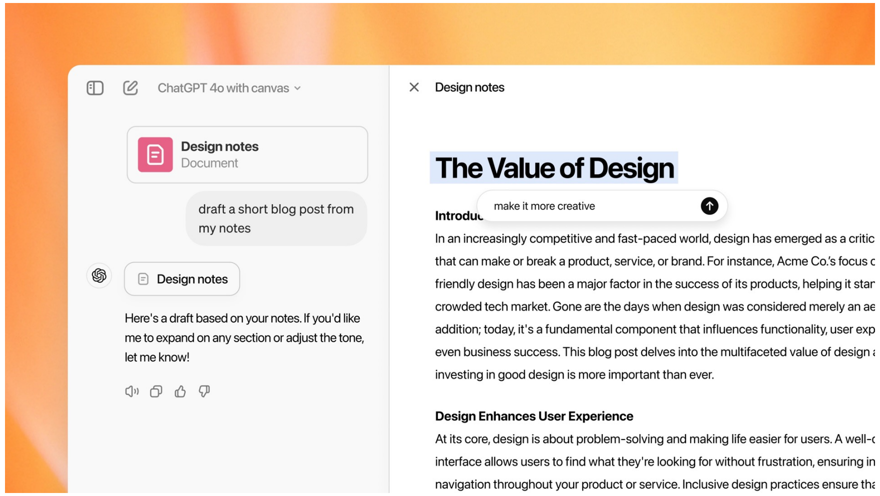
Cross-request KV Cache Reuse

- KV cache can be shared across different requests
  - KV cache depends on the preceding tokens
  - KV cache of requests with the same preceding tokens can be shared

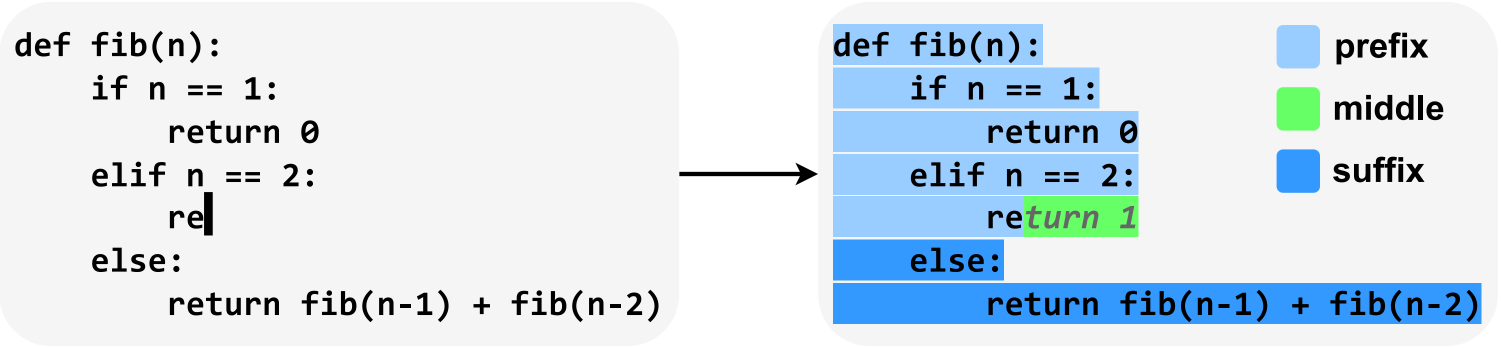


Infilling Tasks with LLMs

- Infilling: Fill in the missing information based on the available data
  - OpenAI Canvas, GitHub Copilot
  - Prompt consists of a prefix, a middle and a suffix
  - Most of the Context (prefix or suffix) remains consistent
  - Interactive with LLMs in multi-turns



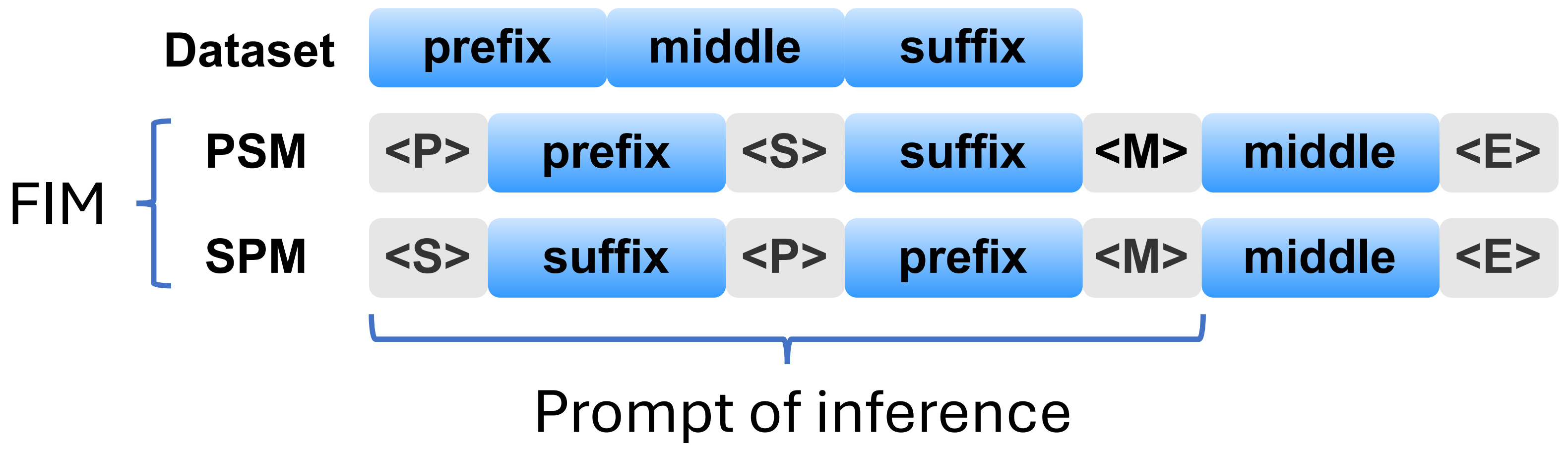
OpenAI Canvas: Writing and Creation

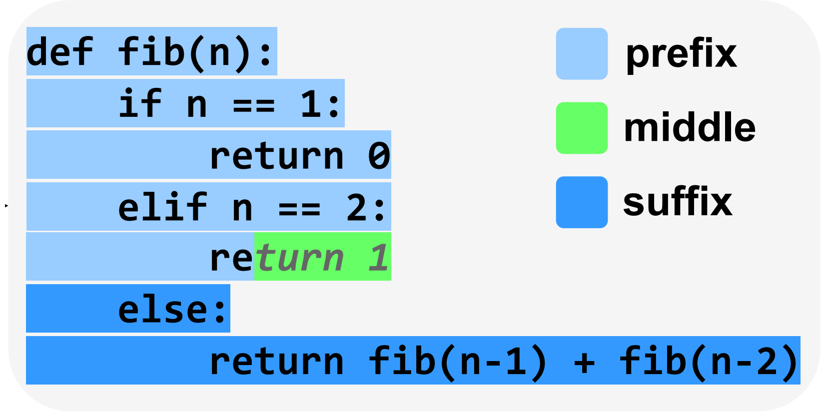


GitHub Copilot: Code Completion

FIM: Fill In the Middle

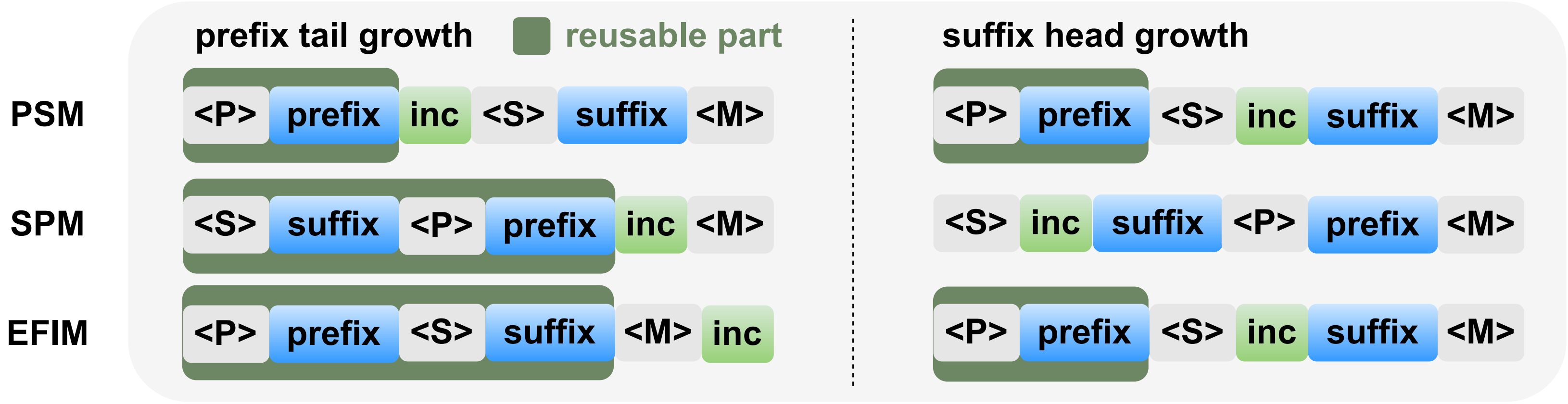
- FIM: Prompt format used in infilling scenario
  - Dataset is split into three parts: prefix, middle and suffix
  - FIM can be used in two ways denoted as PSM and SPM
  - Middle part is the missing information in LLM inference

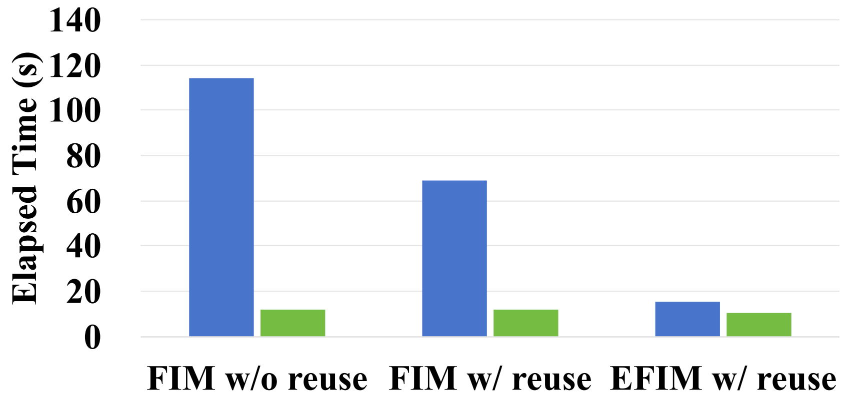




KV Cache Reuse Inefficiency with FIM

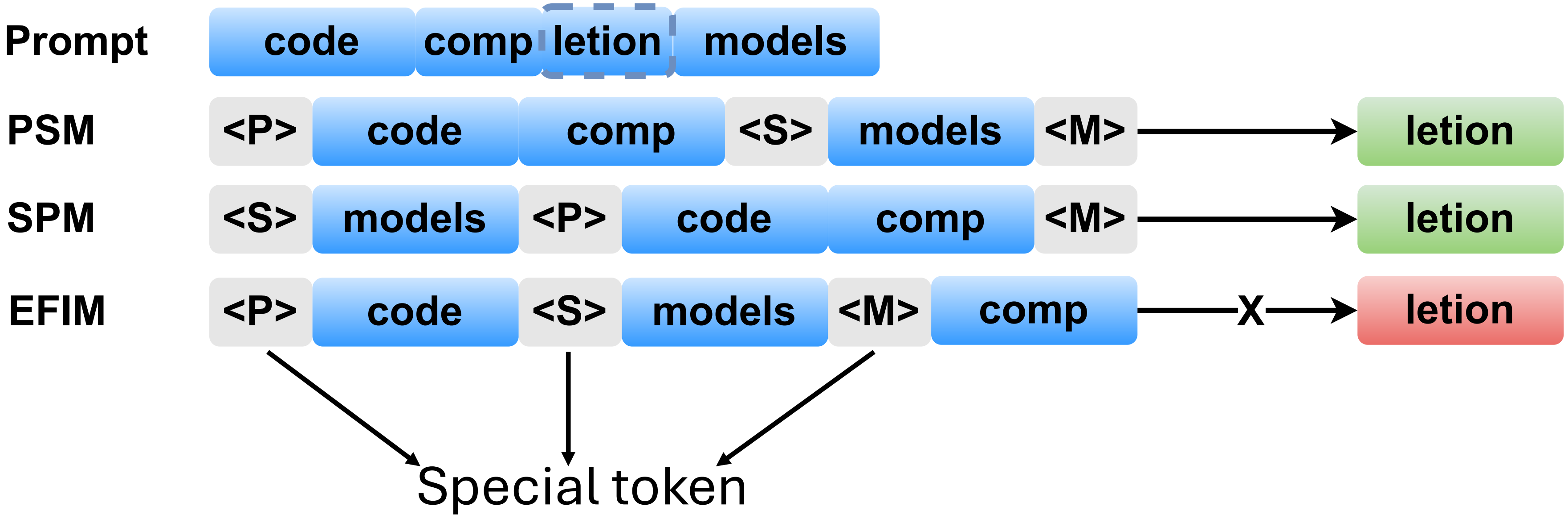
- Common situations in infilling tasks:
  - Prefix tail growth (50%)
  - Suffix head growth (10%)
- EFIM combines the advantage of PSM and SPM
  - EFIM greatly reduce the latency of prefill computation





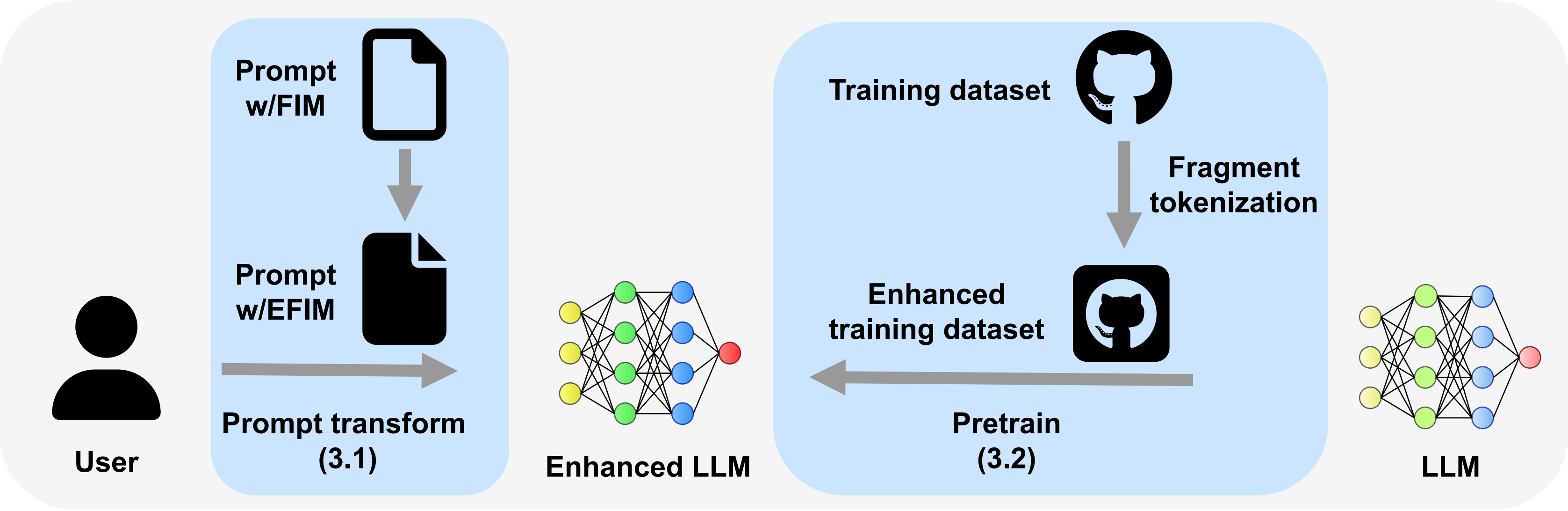
Subtoken Generation Capability with LLMs

- Subtoken: Incomplete word like ‘pri’ in ‘print’
  - Current LLMs do NOT have the ability to generate subtoken
  - Infilling LLMs can generate subtoken only after **special token**
  - Infilling LLMs can NOT generate remaining subtoken after initial subtoken



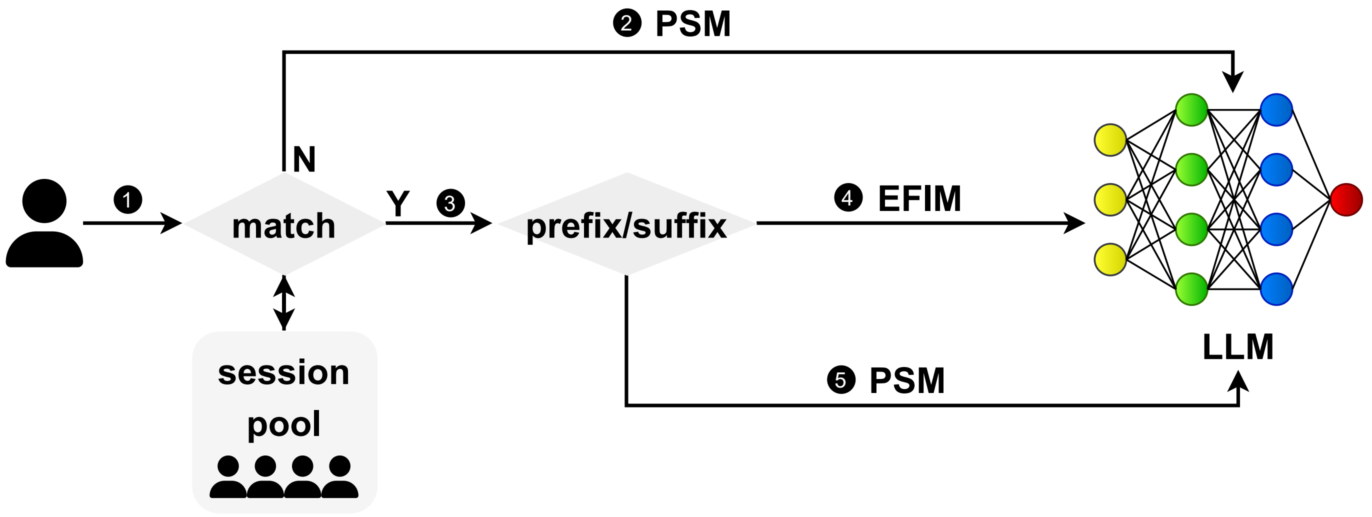
EFIM: Efficient FIM

- We propose EFIM, the first method to transform FIM prompt format, unblocking the potential of KV cache reuse
- To enhance subtoken generation ability, we introduce a fragment tokenization training method on data processing



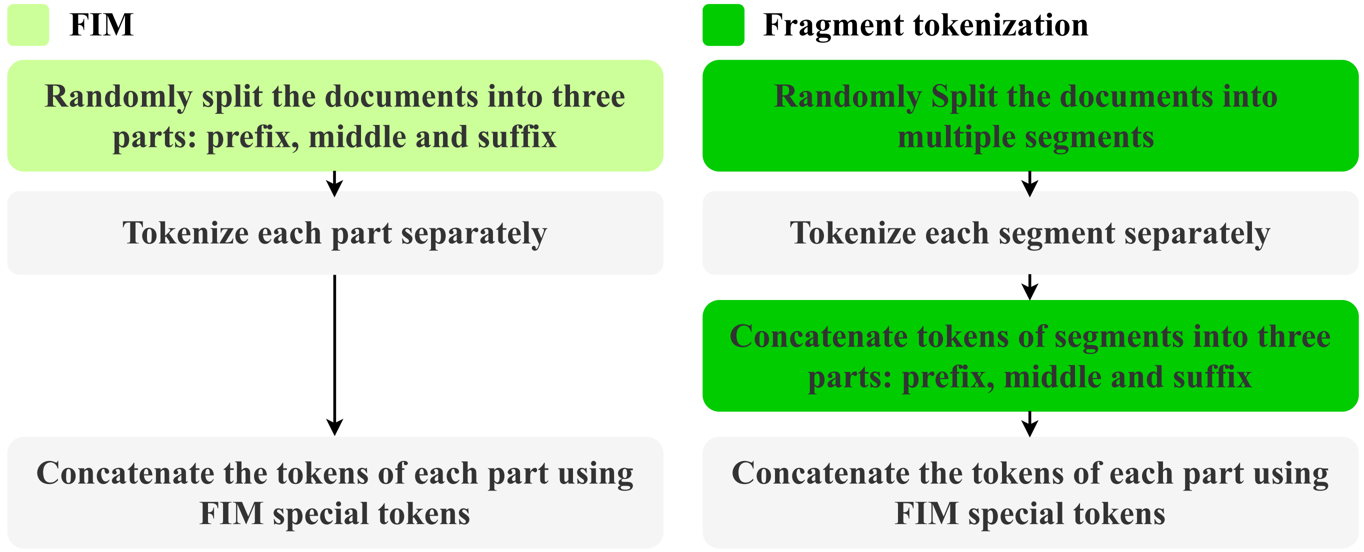
From FIM to EFIM

- Session pool contains users' historical prefix and suffix
  - Not hit in the session pool: Send the prompt in PSM format
  - Hit in the session pool
    - Prefix tail growth: send the prompt in EFIM format
    - Suffix head growth: send the prompt in PSM format

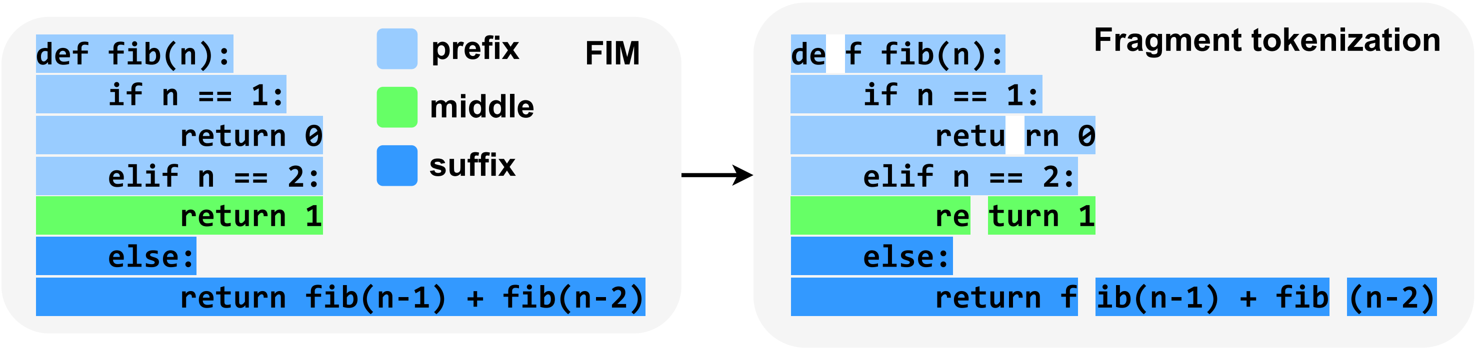


Fragment Tokenization Training Method

Data processing diagram between FIM and fragment tokenization

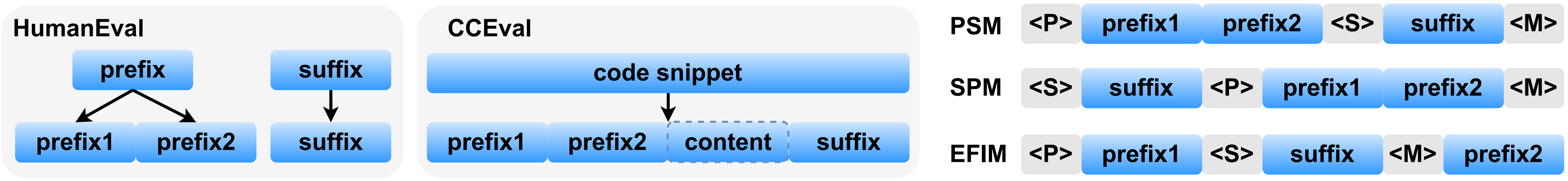


Data processing example between FIM and fragment tokenization



Experimental Methodology

- LLMs: Deepseek-coder-6.7B and Llama3.1-8B
- Benchmarks: HumanEval Infilling and CrossCodeEval (CCEval)
- Metrics
  - Quality: Pass@1, Exact Match (EM) and Edit Similarity (ES)
  - Speed: Latency, Throughput, Reuse Rate
- Schemes
  - Quality: oLLM/eLLM with FIM/EFIM
  - Speed: PSM w/o reuse, PSM w/ reuse, EFIM w/ reuse



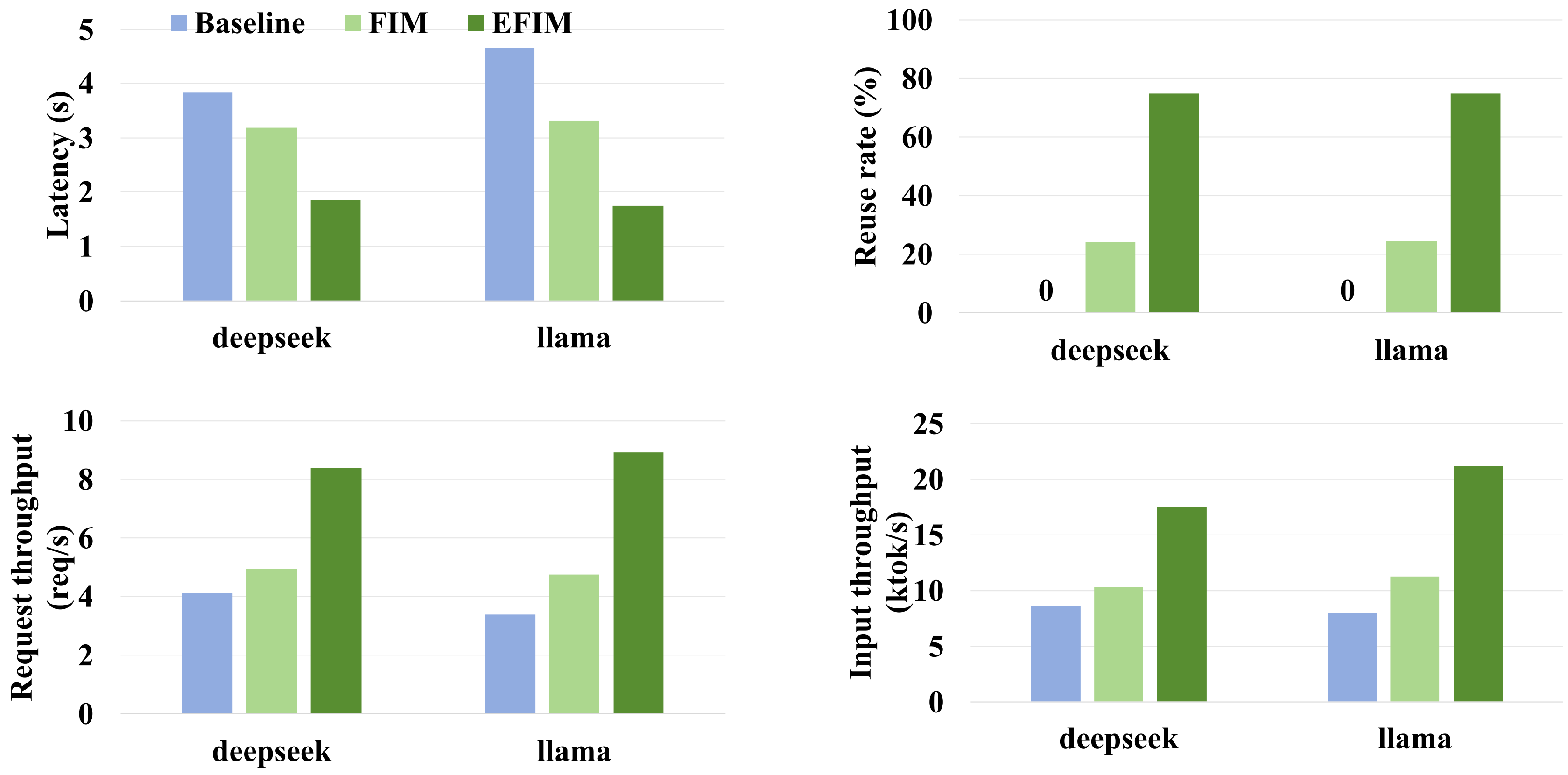
Infilling and Subtoken Generation Ability

- Single-line (S) and multi-line (M) in HumanEval do NOT introduce subtoken
  - Performance of oLLM w/FIM (1st line) and oLLM w/EFIM (2nd line) **remains close**
- Random-span (R) in HumanEval and CCEval contains subtoken
  - Performance of oLLM w/EFIM (2nd line) **drops a lot** (underlined number)
- Fragment tokenization makes eLLM own subtoken generation ability
  - Performance of eLLM w/EFIM (4th line) and oLLM w/FIM (1st line) **remains close**

Benchmark	HumanEval Infilling						CCEval			
Model	Deepseek			Llama			Deepseek		Llama	
	S	M	R	S	M	R	EM	ES	EM	ES
oLLM w/FIM	89.64	61.96	76.77	87.32	56.90	62.99	33.51	78.43	29.40	71.30
oLLM w/EFIM	90.03	62.25	<u>52.44</u>	86.35	56.54	<u>38.35</u>	<u>11.19</u>	<u>71.04</u>	<u>6.82</u>	<u>53.44</u>
eLLM w/FIM	88.48	61.62	75.12	87.12	57.73	67.20	33.27	79.24	31.51	71.15
eLLM w/EFIM	89.64	62.82	75.61	86.83	56.35	64.27	32.51	78.91	30.91	70.48

Inference Speedup

- EFIM achieves 52% latency reduction and 98% throughput increase
  - EFIM achieves 70%+ KV cache reuse rate far beyond 20%+ of FIM



Summary

- We identify that the efficiency of LLM inference for infilling tasks is hindered by the FIM format, as the KV cache of the prefix/suffix part is frequently invalidated by the growing suffix/prefix
- We propose EFIM, the first method to transform the FIM prompt format, unlocking the potential of KV cache reuse
- To enhance subtoken generation ability, we introduce a fragment tokenization training method on data processing
- Experiments on two pretrained LLMs show that EFIM reduces average latency by 52% and increases throughput by 98%, while preserving model capability

EFIM: Efficient Serving of LLMs for Infilling Tasks with Improved KV Cache Reuse

中山大學
SUN YAT-SEN UNIVERSITY

ARSYSU

GitHub https://github.com/gty111/EFIM

instinctguo/deepseek-coder-6.7b-enhance
instinctguo/llama3.1-8b-train
instinctguo/llama3.1-8b-enhance

Tianyu Guo, Hande Dong, Yichong Leng, Feng Liu, Cheater Lin, Nong Xiao and Xianwei Zhang

Email: guoty9@mail2.sysu.edu.cn

My Homepage

Thank You!

EURO-PAR CONFERENCE 2025