Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation

Xuefei Ning[*1]**,** Zinan Lin[*2] , Zixuan Zhou[*14], Zifu Wang[3], Huazhong Yang[1], Yu Wang[1]

foxdoraame@gmail.com linzinan1995@gmail.com zhouzx21@mails.tsinghua.edu.cn

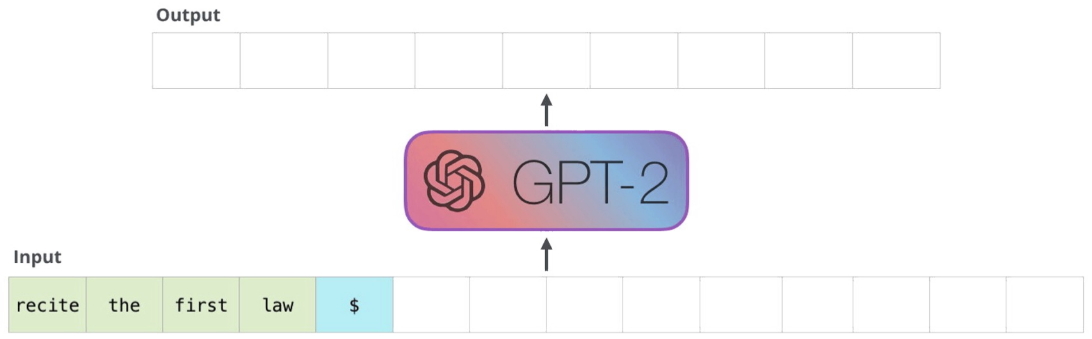
zifuwang0731@gmail.com yanghz@tsinghua.edu.cn yu-wang@tsinghua.edu.cn

[1]Department of Electronic Engineering, Tsinghua University
[2]Microsoft Research, Redmond, Washington, USA
[3]ESAT-PSI, KU Leuven, Leuven, Belgium
[4]Infinigence-AI
*Equal contribution

Menu

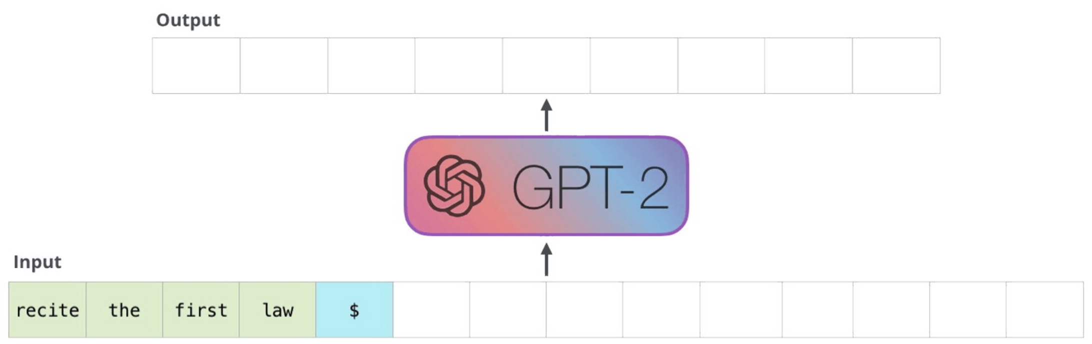
LLM Inference Process

- A typical LLM inference consists of two phases
  - **Prefilling** Phase: takes a prompt sequence to generate the key-value cache and the first token
  - **Decoding** Phase: utilizes and updates the KV cache to generate tokens one by one, where the current token generation depends on all the previously generated tokens



LLM Inference Process

- A typical LLM inference consists of two phases
  - **Prefilling** Phase: takes a prompt sequence to generate the key-value cache and the first token
    - Load every weight onto the GPU chip once to prefill the whole prompt sequence (N tokens)
  - **Decoding** Phase: utilizes and updates the KV cache to generate tokens one by one, where the current token generation depends on all the previously generated tokens
    - Load every weight onto the GPU chip once to decode only 1 token
    - Long end-to-end (E2E) latency: The latency of decoding 1 token is comparable to that of prefilling 128 tokens



Model	Latency of prefilling 128 token / Latency of decoding 1 token (ms)
LLaMA-7B	40 / 43
LLaMA-13B	54 / 58
LLaMA-33B	100 / 86

※ Run on NVIDIA A100 GPU, **batch size** is 1.

Causes of LLM's Slow Inference or Training

1. Model scale: A large number of weights
2. Architecture: Attention operation has quadratic complexity (computation & memory & memory access) w.r.t. input token length
3. Decoding approach in inference: Generate tokens one by one (fully sequential)

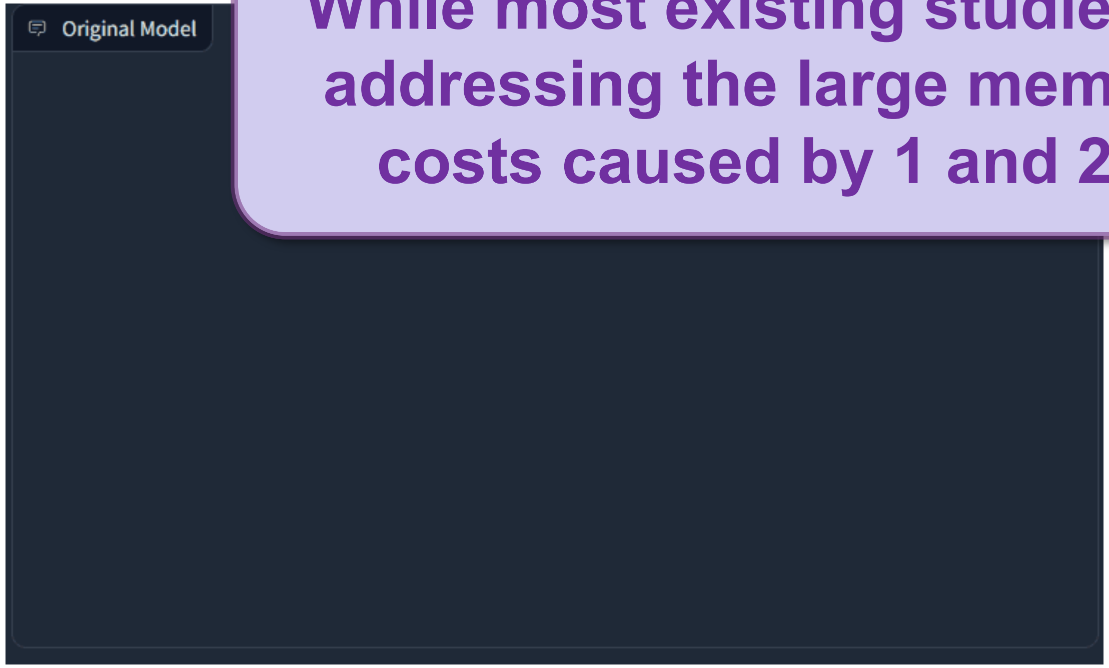
Due to 1 and 3, the decoding phase is heavily bottlenecked by weight loading, GPU computation units cannot be well utilized in this sequential decoding phase

Model	GPU Utilization in the decoding process
LLaMA-7B	**0.31%**
LLaMA-13B	**0.44%**
LLaMA-33B	**0.75%**

※ Run on NVIDIA A100 GPU, **batch size** is 1.

Causes of LLM’s Slow Inference or Training

1. Model scale: A large number of weights
2. Architecture: Attention operation has quadratic complexity (computation & memory & memory access) w.r.t. input token length
3. Decoding approach in inference: Generate tokens one by one (fully sequential)



While most existing studies improve the E2E latency by addressing the large memory access and computation costs caused by 1 and 2, we tackle the third cause!

heavily

utilized

e

Model	**GPU Utilization in the decoding process**
LLaMA-7B	**0.31%**
LLaMA-13B	**0.44%**
LLaMA-33B	**0.75%**

※ Run on NVIDIA A100 GPU, **batch size** is 1.

Menu

Skeleton-of-Thought: Intuition

- **Q**: Do humans always write the answer to a question **fully sequentially**?
- **A**: No. Our thought and answers are usually **structured**: Instead of searching in our minds for the next words sequentially, we usually develop a skeleton in our minds or on paper, and then explicate each point, especially in formal cases (e.g., consultancy, exams).

Take a test: "I can answer this question from 3 orthogonal aspects"
- "To explain the 1st aspect, I can refer to that famous saying"
- "To illustrate the 2nd aspect, I can craft an intuitive example"
- "To detail the 3rd aspect, I will discuss 5 dimensions, …"

Give a tutorial: "I should (1) first give an application example of A, (2) give the solid definition of A, (3) then explain the definition by formula and examples, (4) then connect and discriminate with other concepts, and finally (5) based on the application and connections, emphasize A is important"

Write a story: "I should first describe the character, putting a foreshadowing on the bad end, …. In the end, I will call back to foreshadowing"

…

Skeleton-of-Thought: Intuition

- **Q**: Do humans always write the answer to a question **fully sequentially**?
- **A**: No. Our thought and answers are usually **structured**: Instead of searching in our minds for the next words sequentially, we usually develop a skeleton in our minds or on paper, and then explicate each point, especially in formal cases (e.g., consultancy, exams).

Take a test: “I can answer this question from 3 orthogonal aspects”

ing”

le”

This intuition has our back to question the necessity of fully sequential decoding, which is a major cause of its long E2E latency.

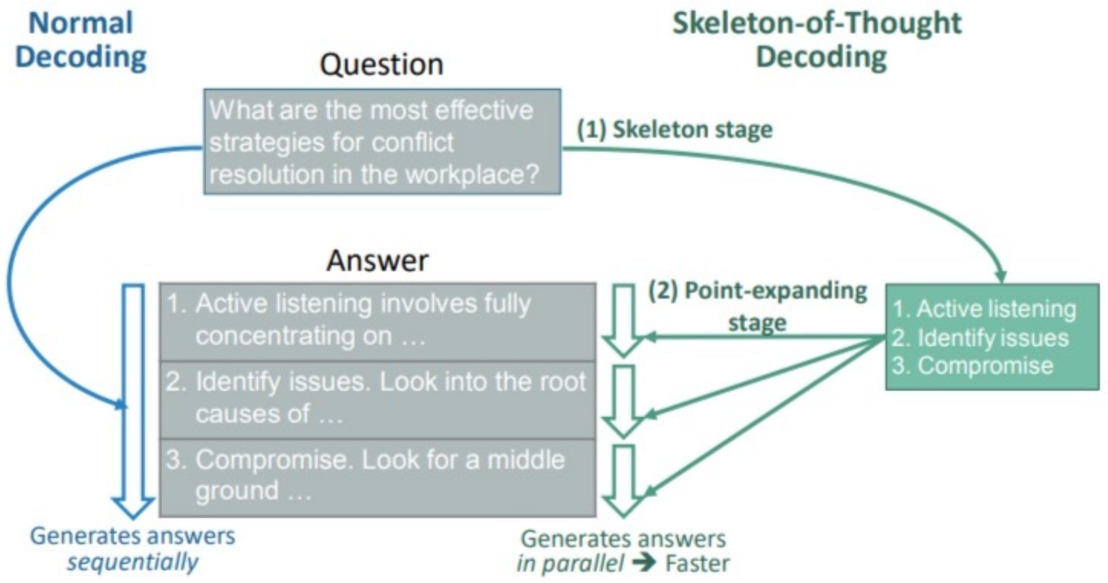
le of A, (2) give the solid definition of A, (3) then explain the definition by formula and examples, (4) then connect and discriminate with other concepts, and finally (5) based on the application and connections, emphasize A is important”

Write a story: “I should first describe the character, putting a foreshadowing on the bad end, …. In the end, I will call back to foreshadowing”

…

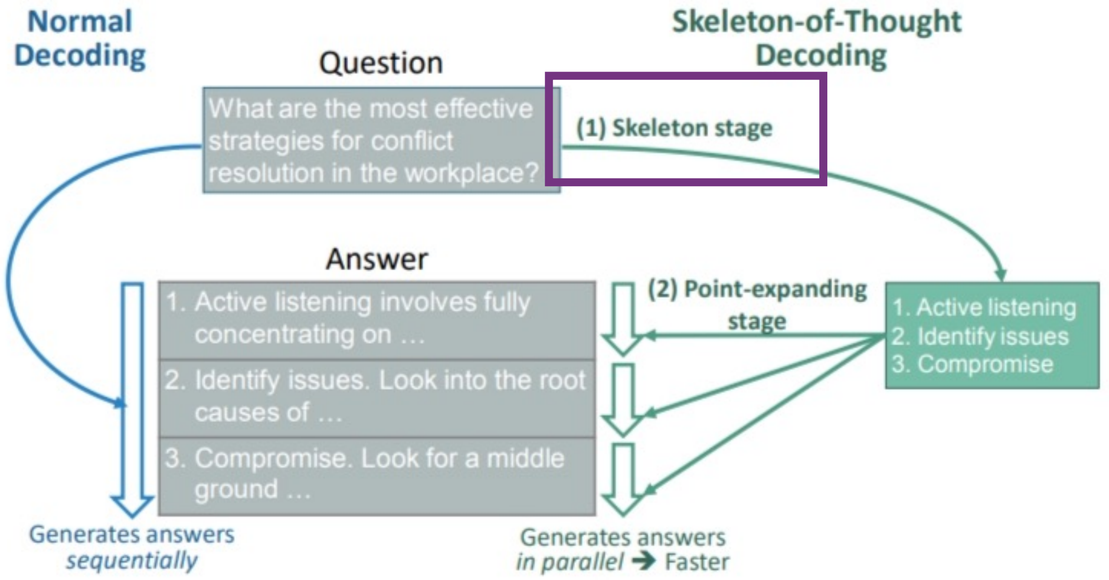
Skeleton-of-Thought: Method

- **Core idea**: SoT prompts the LLM itself to **give a skeleton first** and then **write the overall answer "as parallel as possible"**, instead of sequentially as in normal decoding.


Normal Decoding
Question
What are the most effective strategies for conflict resolution in the workplace?
Skeleton-of-Thought Decoding
(1) Skeleton stage
Answer
1. Active listening involves fully concentrating on …
2. Identify issues. Look into the root causes of …
3. Compromise. Look for a middle ground …
(2) Point-expanding stage
1. Active listening
2. Identify issues
3. Compromise
Generates answers sequentially
Generates answers in parallel → Faster

Skeleton-of-Thought: Method

- Skeleton-of-Thought (SoT) consists of two stages
  - (1) **Skeleton stage**: Guide the LLM to output a concise skeleton of the answer

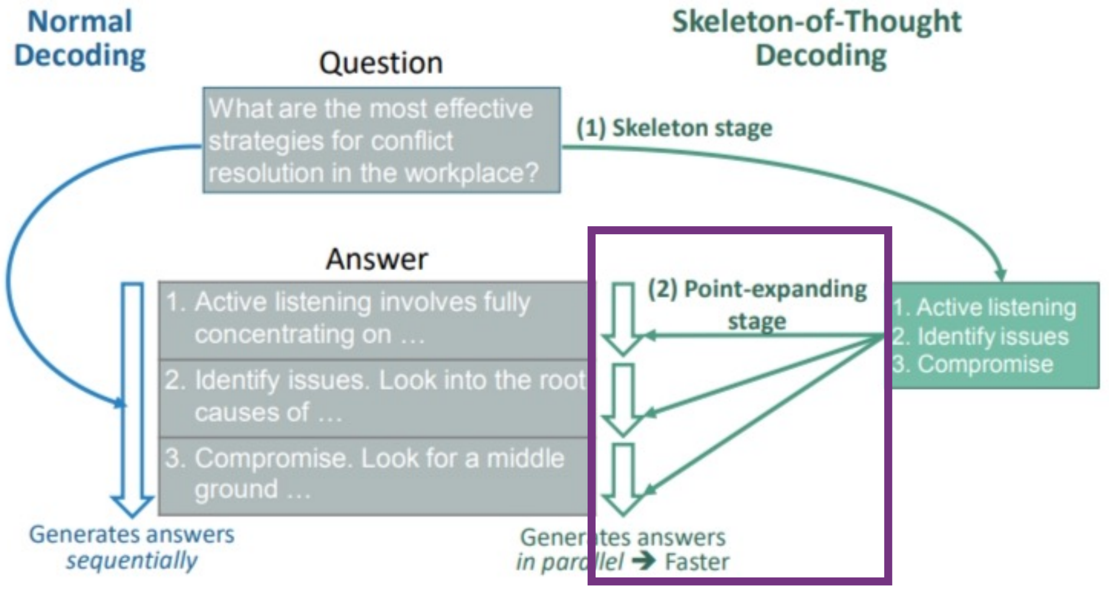

Normal Decoding
Question
What are the most effective strategies for conflict resolution in the workplace?
Skeleton-of-Thought Decoding
(1) Skeleton stage
Answer
1. Active listening involves fully concentrating on …
2. Identify issues. Look into the root causes of …
3. Compromise. Look for a middle ground …
(2) Point-expanding stage
1. Active listening
2. Identify issues
3. Compromise
Generates answers sequentially
Generates answers in parallel → Faster

Skeleton Prompt Template

You're an organizer responsible for only giving the skeleton (not the full content) for answering the question. Provide the skeleton in a list of points (numbered 1., 2., 3., etc.) to answer the question. Instead of writing a full sentence, each skeleton point should be very short with only 3~5 words. Generally, the skeleton should have 3~10 points.
……

Skeleton-of-Thought: Method

- Skeleton-of-Thought (SoT) consists of two stages
  - (2) **Point-expanding stage**: Based on the output skeleton, guide the LLM to expand on each point in parallel



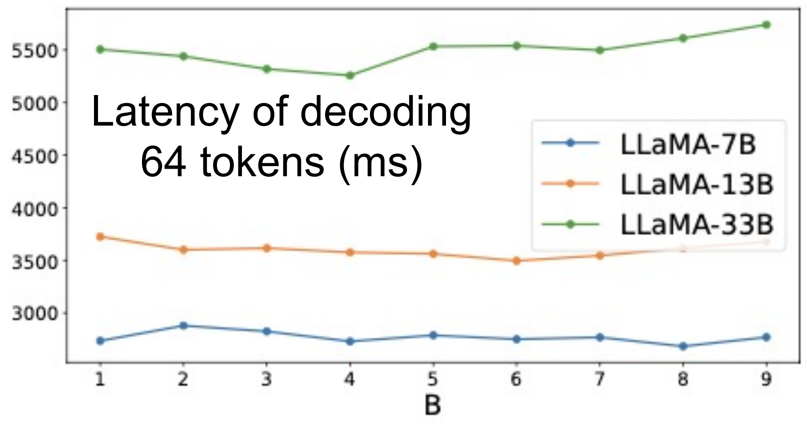
Normal Decoding
Question
What are the most effective strategies for conflict resolution in the workplace?
Skeleton-of-Thought Decoding
(1) Skeleton stage
Answer
1. Active listening involves fully concentrating on …
2. Identify issues. Look into the root causes of …
3. Compromise. Look for a middle ground …
(2) Point-expanding stage
1. Active listening
2. Identify issues
3. Compromise
Generates answers sequentially
Generates answers in parallel → Faster

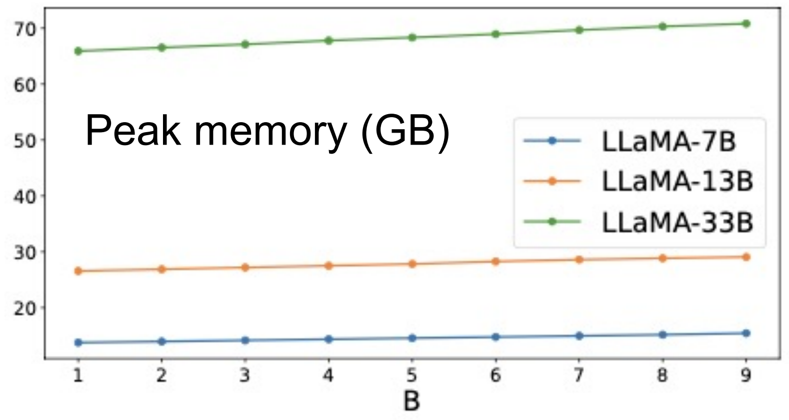
Point-Expanding Prompt Template

You're responsible for continuing the writing of one and only one point in the overall answer to the following question.
{question}
The skeleton of the answer is
{skeleton}
Continue and only continue the writing of point {point index}. Write it **very shortly** in 1~2 sentence and do not continue with other points

Skeleton-of-Thought: Method

- Skeleton-of-Thought (SoT) consists of two stages
  - (2) **Point-expanding stage**: Based on the output skeleton, guide the LLM to expand on each point in parallel
    - Parallel API call for APIs
    - Batched decoding for local models: When the query size is small, increasing the batch size does not increase the per-token latency and the peak memory much => With batch size B, we can get about Bx tokens on the same GPU within the same time



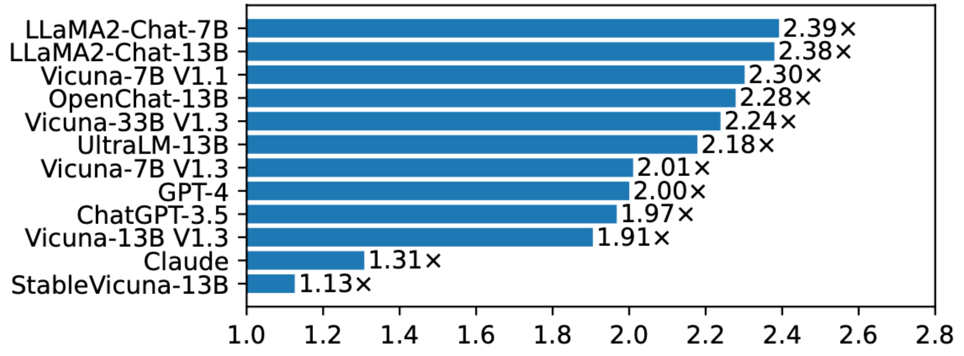


Skeleton-of-Thought: Efficiency Evaluation

We evaluate 12 models (including 9 open-source models and 3 API-based models) on the Vicuna-80 dataset (containing 9 categories).

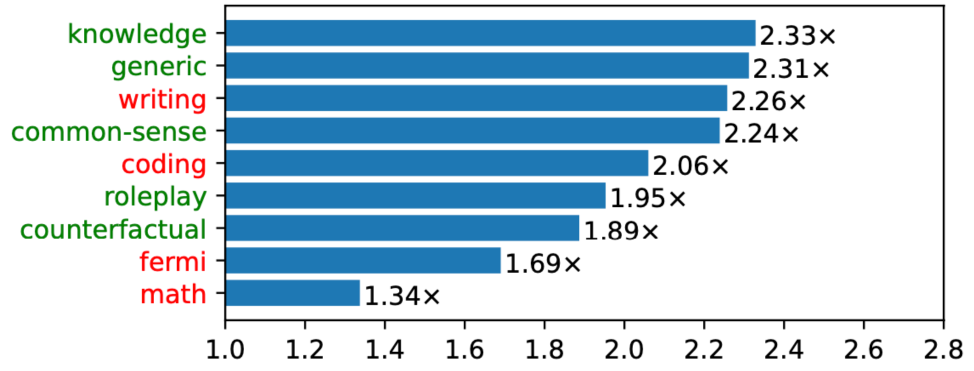
Speed-up Breakdown: Models

- More than **1.9×** speed-up on ten LLMs



Speed-up Breakdown: Question Categories

- **1.89× ~ 2.33×** speed-up on the categories that SoT can provide high-quality answers

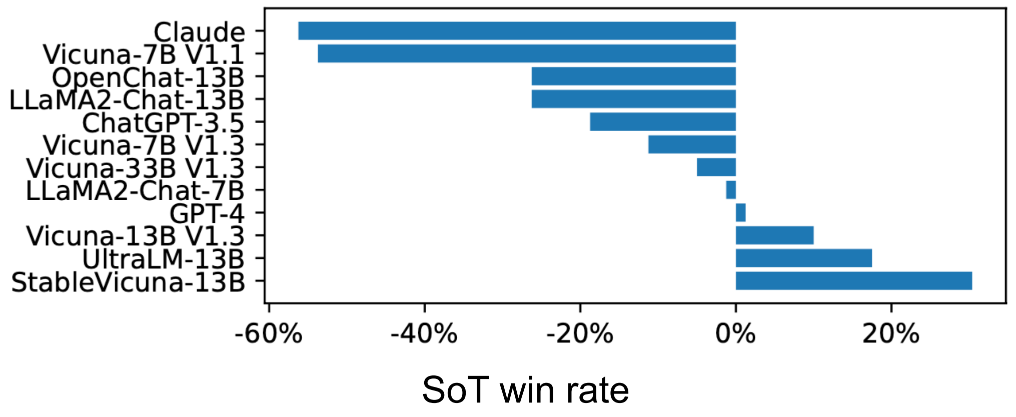


Skeleton-of-Thought: Quality Evaluation

We evaluate 12 models (including 9 open-source models and 3 API-based models) on the Vicuna-80 dataset (containing 9 categories).

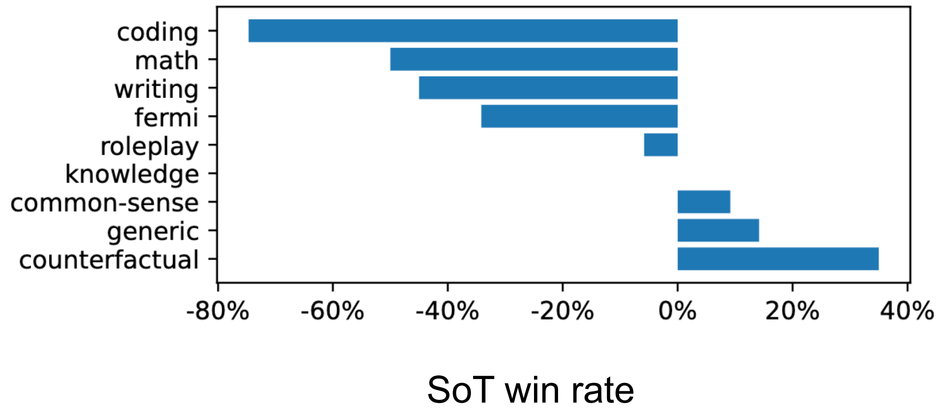
Quality Breakdown: Models

- **UltraLM-13B, Vicuna-13B V1.3, and GPT-4** have relatively high net win rates, while **Claude, Vicuna-7B V1.1, OpenChat-13B, LLaMA2-Chat-13B, and ChatGPT-3.5** have low net win rates.



Quality Breakdown: Question Categories

- SoT performs well on **counterfactual, generic, common-sense, knowledge, and roleplay**, but not so well on **coding, math, writing, and fermi**.

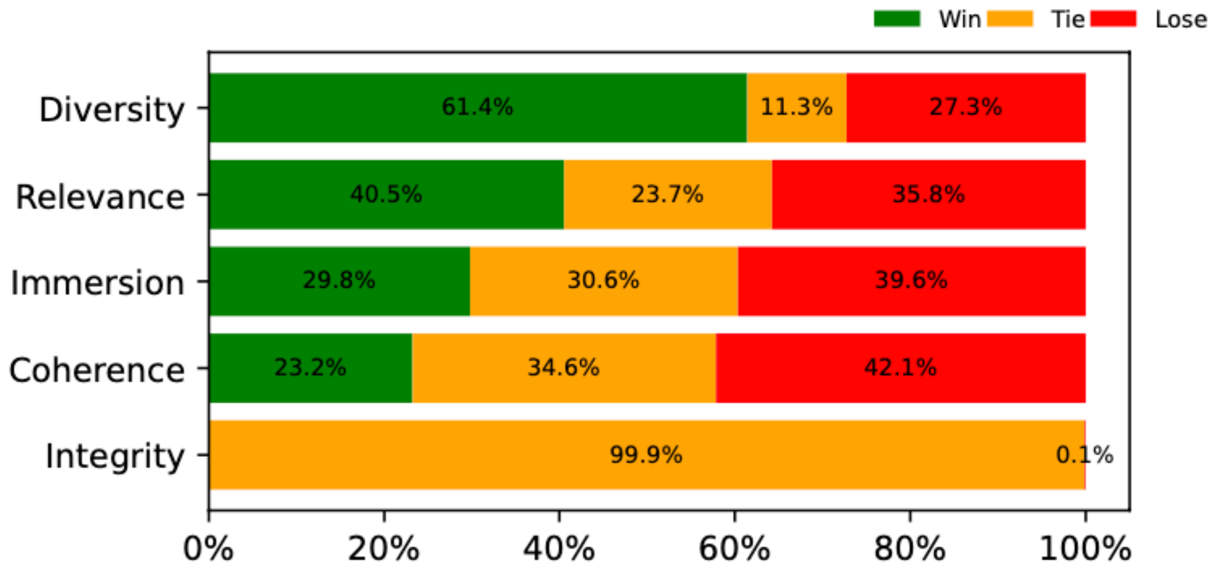


※ Evaluated with FastChat comparison prompt and GPT-4 as the judge.

Skeleton-of-Thought: Quality Evaluation

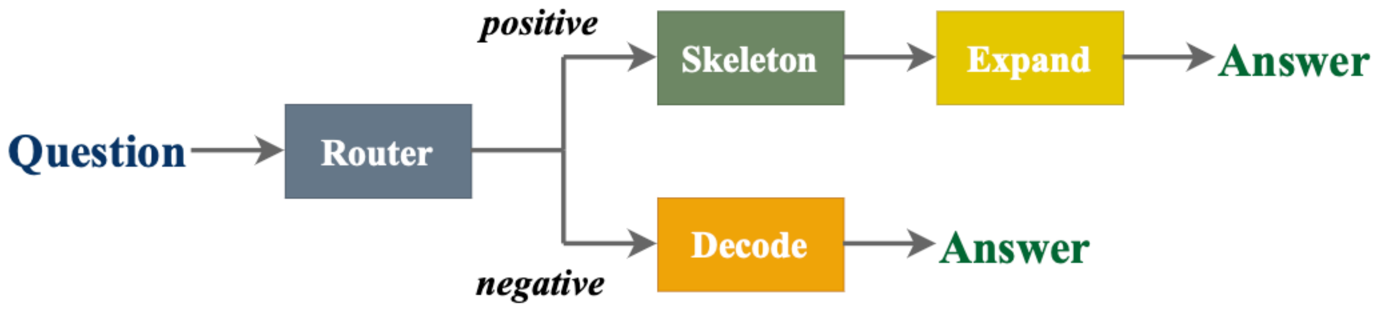
We evaluate 12 models (including 9 open-source models and 3 API-based models) on the Vicuna-80 dataset (containing 9 categories).

Besides achieving speedup, SoT even **has the potential to improve the answer quality** in terms of **diversity** and **relevance**, but currently **hurts the immersion and coherence quality**.



Skeleton-of-Thought with Router (SoT-R)

- **Core idea**: Use a router module to decide whether SoT should be applied for a given user query, such that the SoT generation mode is only triggered for suitable questions.



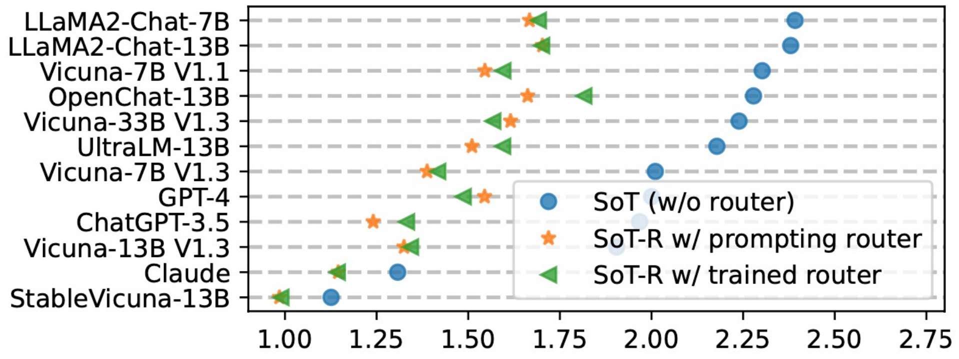
- Experiment with two types of router
  - Prompting Router: Ask an LLM if the question is suitable for SoT
  - Trained Router: Use RoBERTa to do sequence classification, trained on manually annotated LIMA train set (1030 QA pairs)

SoT-R: Efficiency and Quality Evaluation

We evaluate 12 models (including 9 open-source models and 3 API-based models) on the Vicuna-80 dataset (containing 9 categories).

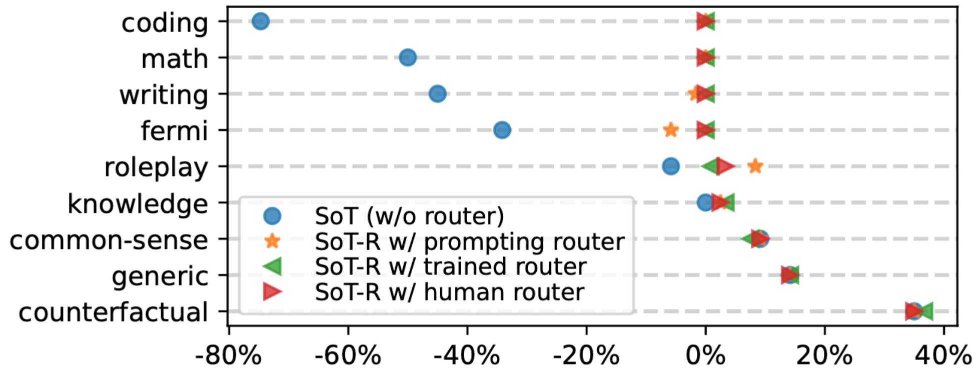
Speed-ups: Models

- SoT-R can still **keep >1 speed-up for most models**.



Quality: Question Categories

- For the question categories not suitable for SoT, SoT-R can fall back to the normal generation mode. Consequently, SoT-R can **improve answer quality on some categories** (e.g., counterfactual, generic, common-sense, knowledge), and **maintain good answer quality for all question categories**.

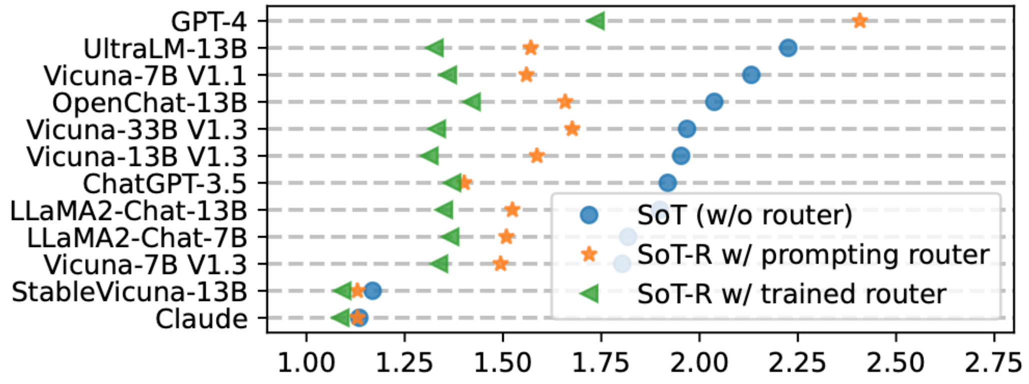


SoT-R: Efficiency and Quality Evaluation

We evaluate 12 models (including 9 open-source models and 3 API-based models) on the WizardLM dataset (containing 218 questions, 29 categories).

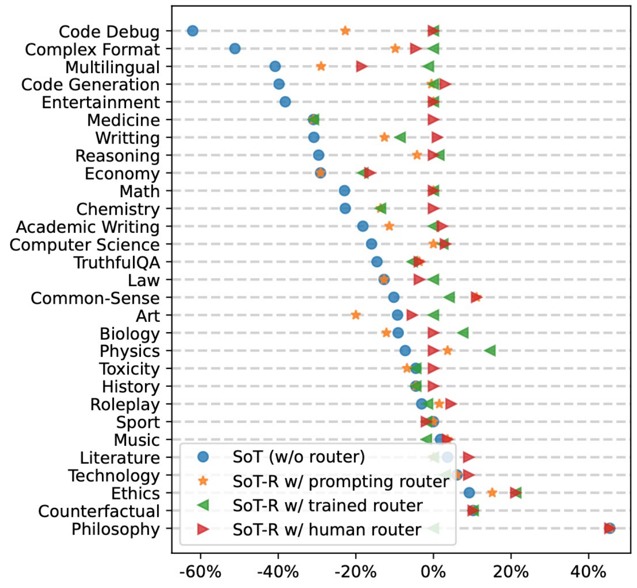
Speed-ups: Models

- SoT-R can still **keep >1 speed-up for most models**.



Quality: Question Categories

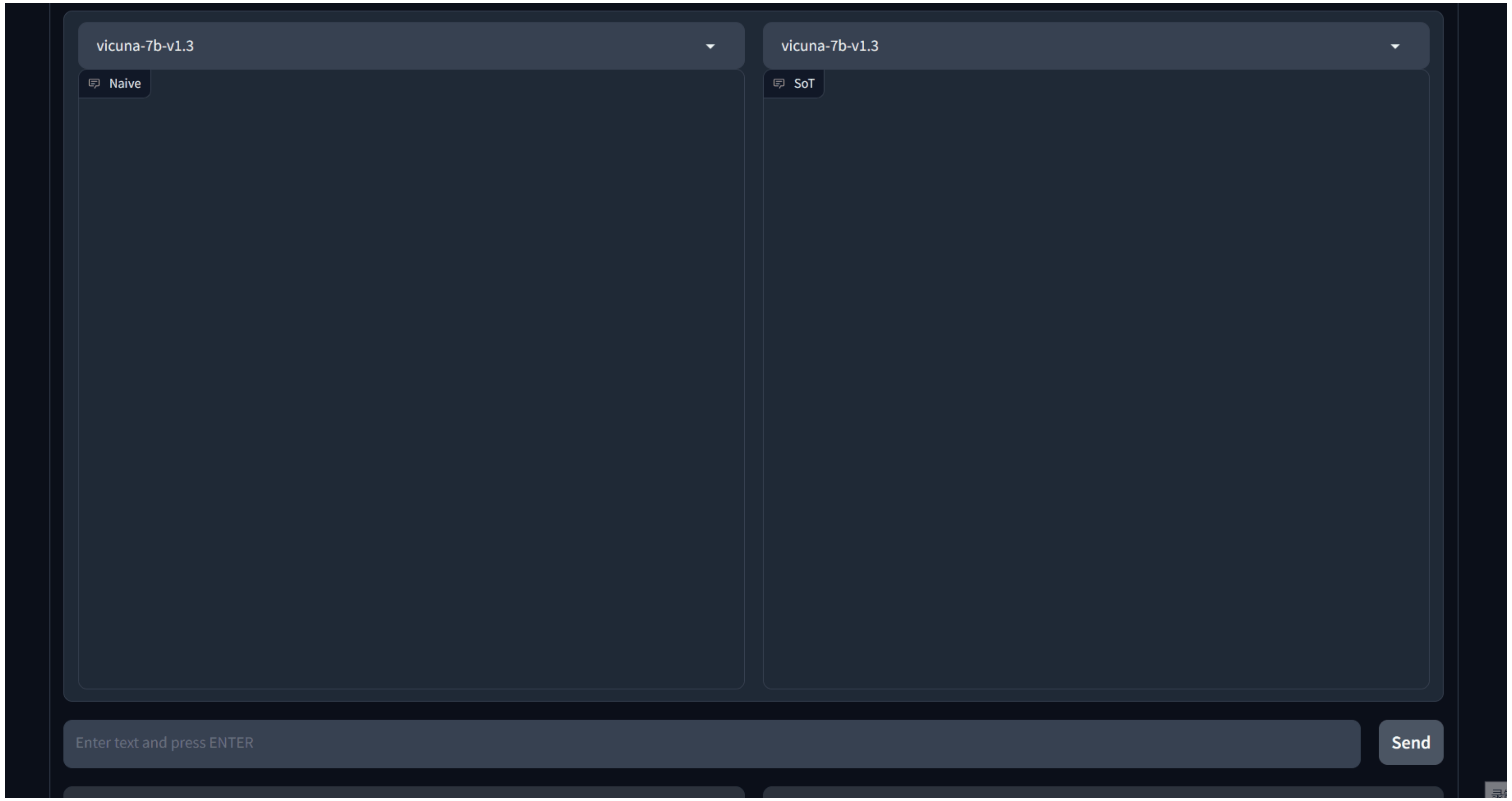
- SoT-R can **improve the answer quality on some categories**, and **maintain good answer quality for most question categories**.



Skeleton-of-Thought: Demo

Vicuna-7B model on one A100 GPU

2.1× end-to-end speed-up compared with sequential decoding



Website: https://sites.google.com/view/sot-llm

Menu

Summary of Skeleton-of-Thought (SoT)

- Accelerating LLM by parallel generation
- Up to **2.39× speed-up** on 12 LLMs, **better answer diversity and relevance**
- An attempt at **data-centric optimization for efficiency**
  - *SoT leverages the emerging ability of the LLM itself to organize the output structure. This introduces parallelism possibility into the generation process, enabling techniques for improving throughput (e.g., batch inference) to help with E2E latency, using the same hardware resource.*
- This explicit thinking strategy **has the potential to improve the answer quality**
  - *Explicitly articulating the structure or thought process in language can elicit high-quality answers from LLMs.*

Limitations and Future Work

- **Issue: Not generally applicable to all user queries**
  - **Organizing answers as dynamic graph-of-thought** might be able to generalize SoT to more complex questions, achieving speed-ups (parallel writing of independent subgraphs) and high quality (more comprehensive) in the meantime.
  - Let the LLM **self-trigger SoT**, e.g., tune the LLM to output "<SoT(ques, outline)>" to trigger parallel point expansion at any time during the generation.
- **Issue: Overhead of SoT in some scenarios brought by the current long SoT prompts**
  - Large token overhead for APIs, and might be unacceptable when charging token usage => **shorter prompt**; **self-trigger**
  - Computation overhead for serving, when the workload is computation-bounded (e.g., during periods with a saturated number of concurrent queries), SoT might hurt the throughput. => Integrate SoT as a new optional parallelism axis in the serving system, and **trigger SoT based on the current system workload**; **shorter prompt**; **self-trigger**

Limitations and Future Work

- **Issue: Not generally applicable to all user queries**
  - **Organizing answers as dynamic graph-of-thought** might be able to generalize SoT to more complex questions, achieving speed-ups (parallel writing of independent subgraphs) and high quality (more comprehensive) in the meantime.
  - Let the LLM **self-trigger SoT**, e.g., tune the LLM to output "<SoT(ques, outline)>" to trigger parallel point expansion at any time during the generation.
- **Issue: Overhead of SoT in some scenarios brought by the current long SoT prompts**
  - Large token overhead for APIs, and might be unacceptable when charging token usage => **shorter prompt**; **self-trigger**
  - Computation overhead for serving, when the workload is computation-bounded (e.g., during periods with a saturated number of concurrent queries), SoT might hurt the throughput. => Integrate SoT as a new optional parallelism axis in the serving system, and **trigger SoT based on the current system workload**; **shorter prompt**; **self-trigger**
- **Data-centric efficiency optimization**: As LLM capability is improving and LLM-generated sequence length is growing, **data-centric techniques for inference efficiency** that utilize LLMs' emerging capability can become more and more useful in the future!

Thanks!

Emails: foxdoraame@gmail.com linzinan1995@gmail.com zhouzx21@mails.tsinghua.edu.cn

zifuwang0731@gmail.com yanghz@tsinghua.edu.cn yu-wang@tsinghua.edu.cn

Code: https://github.com/imagination-research/sot

Website: https://sites.google.com/view/sot-llm