Decomposed Prompting: A Modular Approach for Solving Complex Tasks

Tushar Khot[♣], Harsh Trivedi[♡], Matthew Finlayson[♣], Yao Fu[♠]*,
Kyle Richardson[♣], Peter Clark[♣], Ashish Sabharwal[♣]
[♣]Allen Institute for AI [♡]Stony Brook University [♠]University of Edinburgh
tushark@allenai.org, hjtrivedi@cs.stonybrook.edu, matthewf@allenai.org, yao.fu@ed.ac.uk,
kyler@allenai.org, peterc@allenai.org, ashishs@allenai.org

Focus: Complex Multi-Step Reasoning Tasks

Multi-Hop Questions

Question: Which team does the player named 2015 Diamond Head Classic's MVP play for?

Reasoning: The 2015 Diamond Head Classic's MVP was Buddy Hield. Buddy Hield played for the **Sacramento Kings** in 2015.

HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. Yang'18

Math Questions

Question: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Reasoning: Roger started with 5 balls. 2 cans of 3 tennis balls each is 2*3=6 tennis balls. In total, he has 5 + 6 = **11** tennis balls.

Training Verifiers to Solve Math Word Problems. Cobbe'21

Algorithmic Tasks

Task: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.

Reasoning: The last letter of "Augusta" is "a". The last letter of "Ada" is "a". The last letter of "King" is "g". Concatenating "a", "a", "g" using a space leads to "a a g". So, "Augusta Ada King" outputs "a a g".

Prompting Approach for Complex Tasks

Chain of thought Prompting:

Q => reasoning process + the **answer**

Chain-of-Thought Prompting

Q: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.
A: The last letter of "Augusta" is "a". The last letter of "Ada" is "a". The last letter of "King" is "g". Concatenating "a", "a", "g" using a space leads to "a a g". So, "Augusta Ada King" outputs **"a a g"**.

Q: Take the letters at position 1 of the words in "Alan Mathison Turing" and concatenate them using a space.
A:

Output:

The letter at position 1 of "Alan" is "A". The letter at position 1 of "Mathison" is "M". The letter at position 1 of "Turing" is "T". Concatenating "A", "M", "T" using a space leads to "A M T". So, "Alan Mathison Turing" outputs **"A M T"**.

Task: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.

Reasoning: The last letter of "Augusta" is "a". The last letter of "Ada" is "a". The last letter of "King" is "g". Concatenating "a", "a", "g" using a space leads to "a a g". So, "Augusta Ada King" outputs "a a g".

When the steps get harder?

Chain of thought Prompting:
Q => reasoning process + the **answer**

How can we help LLMs with these harder sub-tasks?

Decomposed Prompting:
- **Decomposer**: Q => decomposition into simpler sub-tasks + answer
- **Sub-Task Handlers**: Library of sub-task specific *tools* (LLMs, APIs, etc)

Chain-of-Thought Prompting

Q: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.
A: The last letter of "Augusta" is "a". The last letter of "Ada" is "a". The last letter of "King" is "g". Concatenating "a", "a", "g" using a space leads to "a a g". So, "Augusta Ada King" outputs **"a a g"**.

Q: Take the letters at **position 2** of the words in "Alan Mathison Turing" and concatenate them using a space.
A:

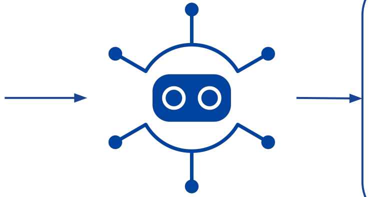
Output:

The letter at position 2 of "Alan" is "l". **The letter at position 2 of "Mathison" is "t". The letter at position 2 of "Turing" is "r".** Concatenating "l", "t", "r" using a space leads to "l t r". So, "Alan Mathison Turing" outputs **"l t r"**.

Providing tools for the sub-tasks

Q: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.
A: The last letter of "Augusta" is "a". The last letter of "Ada" is "a". The last letter of "King" is "g". Concatenating "a", "a", "g" using a space leads to "a a g". So, "Augusta Ada King" outputs **"a a g"**.

Q: Take the letters at **position 2** of the words in "Alan Mathison Turing" and concatenate them using a space.
A:



The letter at position 2 of "Alan" is "l". **The letter at position 2 of "Mathison" is "t". The letter at position 2 of "Turing" is "r".** Concatenating "l", "t", "r" using a space leads to "l t r". So, "Alan Mathison Turing" outputs **"l t r"**.

How do we use these tools?

How can we help LLMs with such hard sub-tasks?

Toolkit

Q: What are the words in "Augusta Ada King"?
A: ["Augusta", "Ada", "King"]
…`

split

Q: What is the letter at the position 4 in "Augusta"?
A: “u”
…

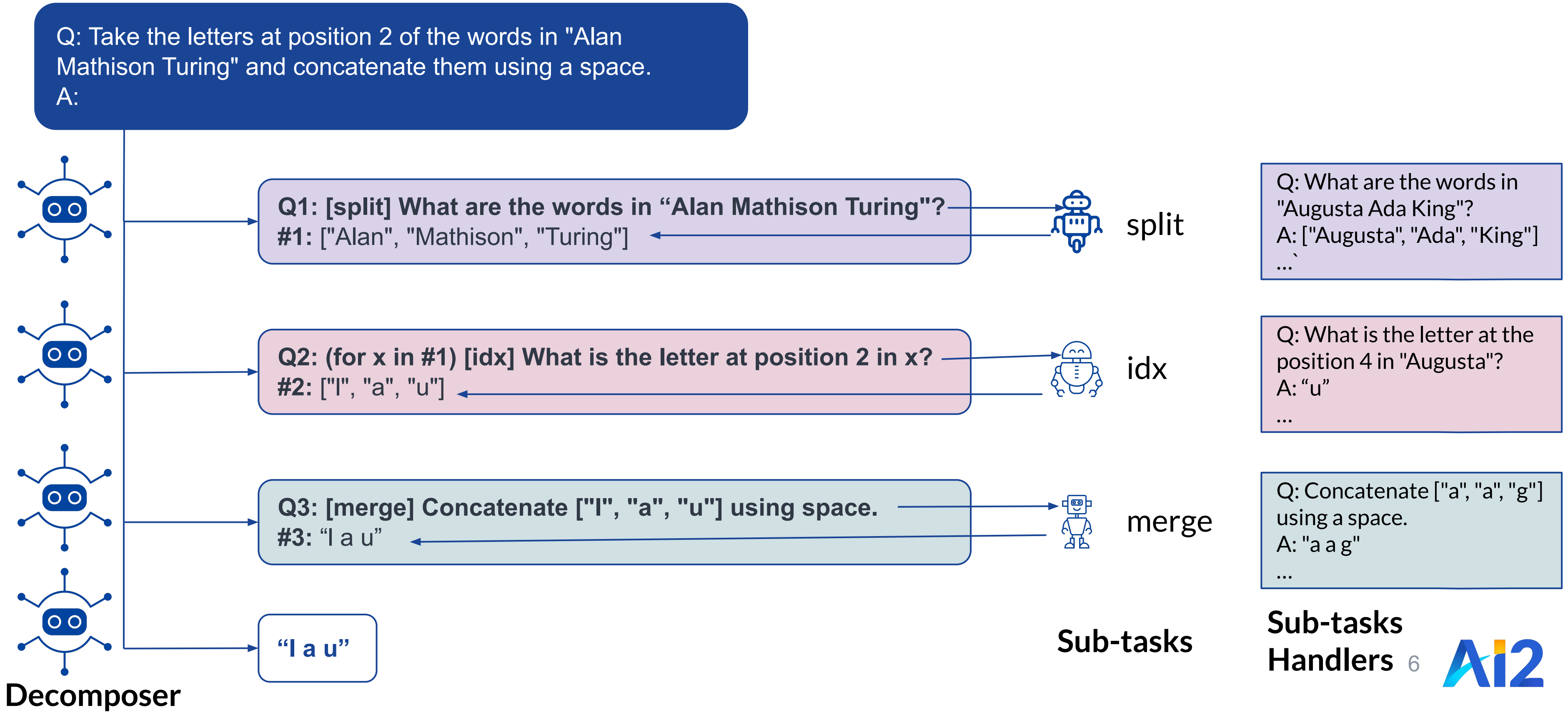
idx

Q: Concatenate ["a", "a", "g"] using a space.
A: "a a g"
…

merge

…

Decomposed Prompting



Q: Take the letters at position 2 of the words in "Alan Mathison Turing" and concatenate them using a space.
A:

Q1: [split] What are the words in "Alan Mathison Turing"?
#1: ["Alan", "Mathison", "Turing"]

Q2: (for x in #1) [idx] What is the letter at position 2 in x?
#2: ["l", "a", "u"]

Q3: [merge] Concatenate ["l", "a", "u"] using space.
#3: "l a u"

"l a u"

Q: What are the words in "Augusta Ada King"?
A: ["Augusta", "Ada", "King"]
…

Q: What is the letter at the position 4 in "Augusta"?
A: "u"
…

Q: Concatenate ["a", "a", "g"] using a space.
A: "a a g"
…

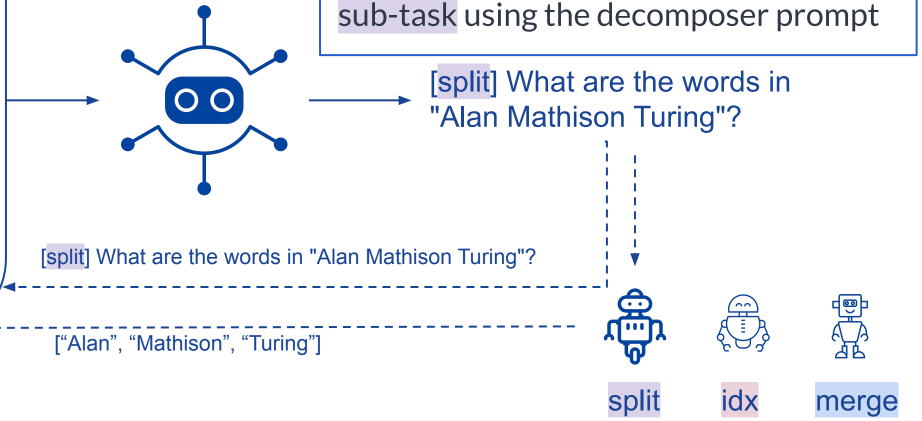
Decomposed Prompting: Decomposer

Decomposer

QC: Take the last letters of the words in "Augusta Ada King" and concatenate them using a space.
QS: [split] What are the words in "Augusta Ada King"?
A: ["Augusta", "Ada", "King"]
QS: (foreach) [idx] What is the last letter in "#1"?
A: ["a", "a", "g"]
QS: [merge] Concatenate #2 using a space.
A: "a a g"
QS: [EOQ]

QC: Take the letters at **position 2** of the words in "Alan Mathison Turing" and concatenate them using a space.
QS:

Iteratively generate next question and sub-task using the decomposer prompt


[split] What are the words in "Alan Mathison Turing"?
[split] What are the words in "Alan Mathison Turing"?
["Alan", "Mathison", "Turing"]
split
idx
merge

[split] [idx] [merge] → Indicates the sub-task name

(foreach) → Operators to efficiently and reliably handle structured outputs

[EOQ] → Indicates answer found

Append generated question and answer from the handler to the prompt to generate the next question.

Decomposed Prompting: Sub-Task Handlers

Build

Toolkit

Q: What are the words in "Augusta Ada King"?
A: ["Augusta", "Ada", "King"]
…`

split

Q: What is the letter at the position 4 in "Augusta"?
A: “u”
…

idx

Q: Concatenate ["a", "a", "g"] using a space.
A: "a a g"
…

merge

Reuse

Retriever

What is the weather in Seattle, USA?

Which paper introduces ELMo?

NL APIs

Calculators

SoTa Models

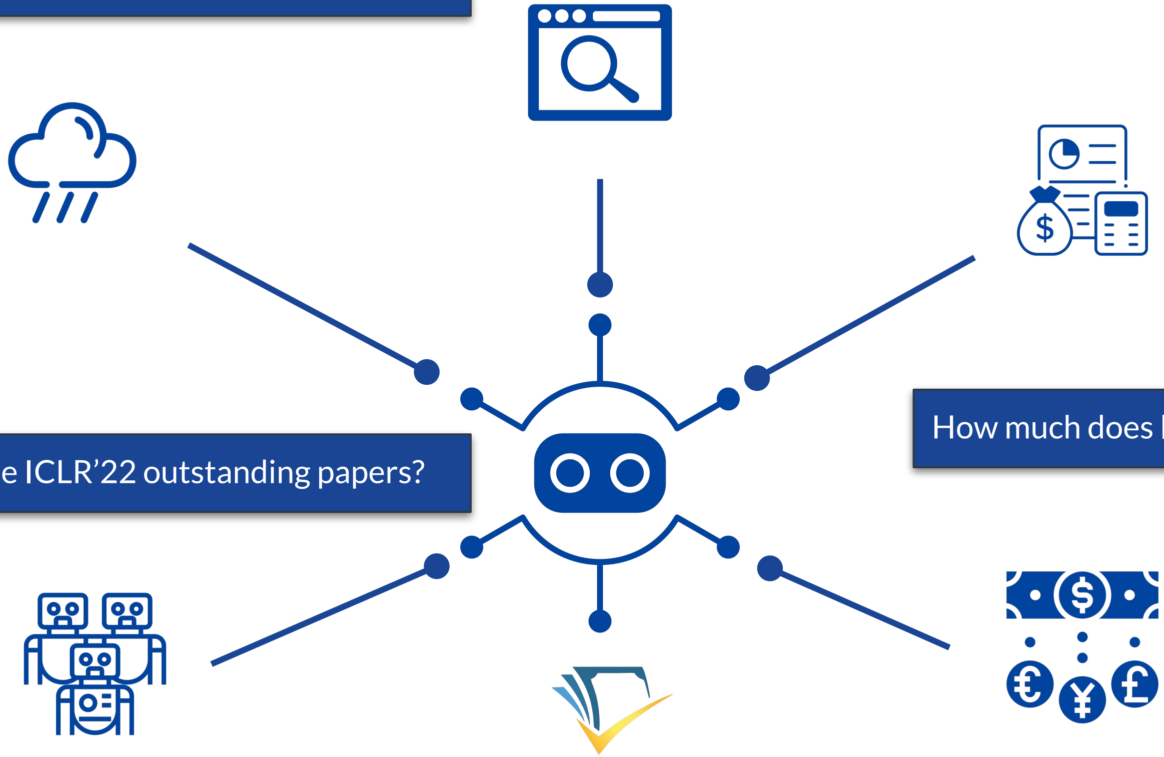
…

DecomP: LLMs w/ Tools

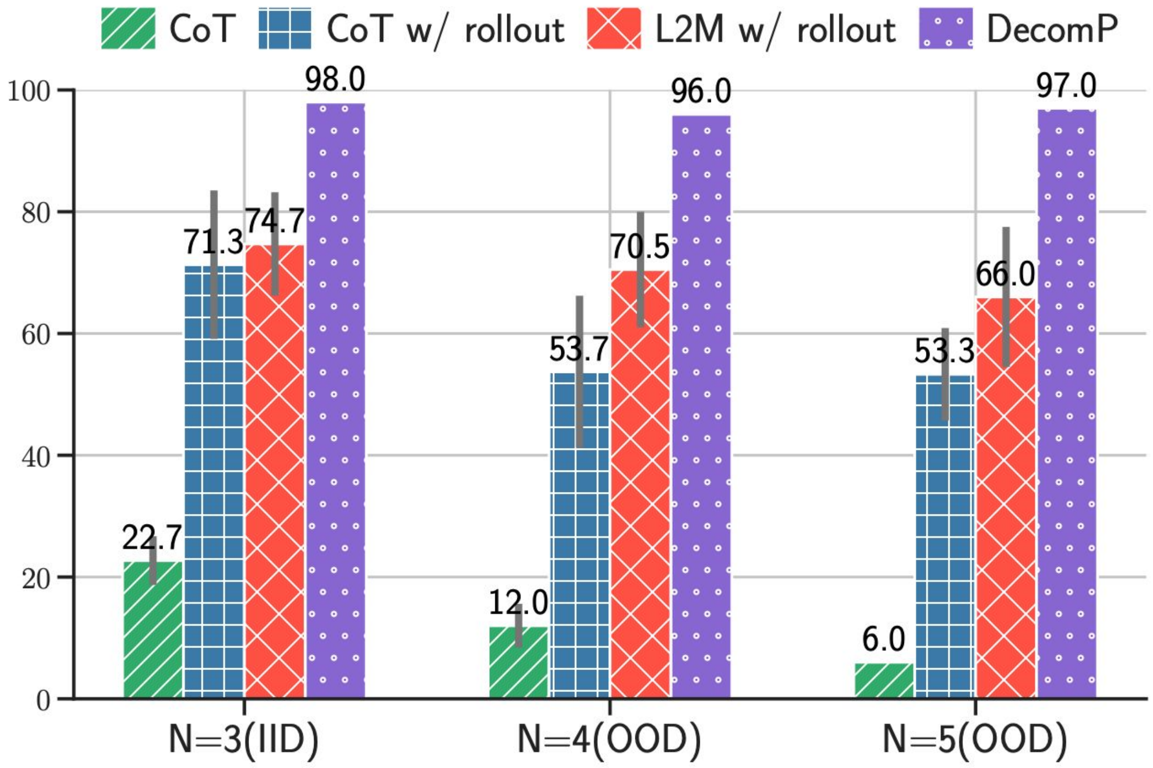
What is the expected weather for ICLR'23?

What is the TL;DR of the ICLR'22 outstanding papers?

How much does lunch cost (€) in Kigali?



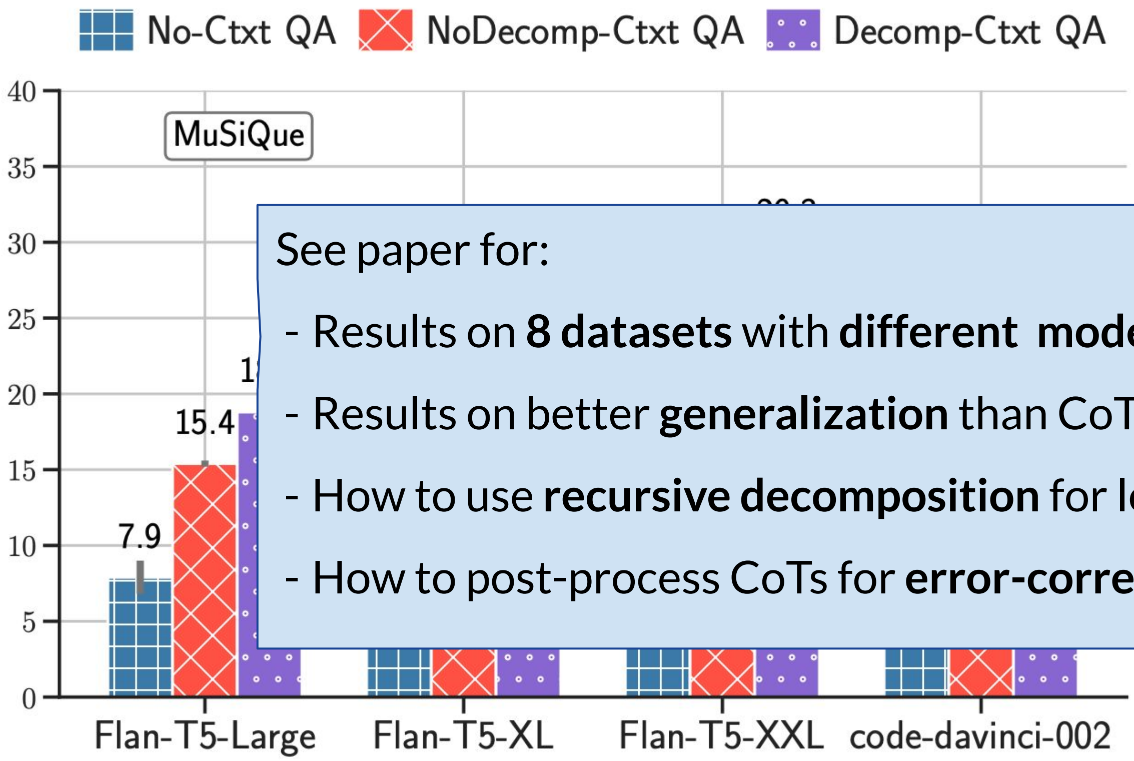
Results: Letter Concatenation



For hard sub-tasks, e.g., identifying the kth letter, building a special sub-task handler leads to improved accuracy

This also leads to better generalization to longer input sequences

Results: Augmenting with Retrieval



Across model sizes, DecomP can be used to improve QA performance over vanilla or ead LLMs

See paper for:

- Results on **8 datasets** with **different model sizes**
- Results on better **generalization** than CoT
- How to use **recursive decomposition** for long sequences
- How to post-process CoTs for **error-correction.**

Conclusion

https://github.com/allenai/decomp

- Decomposed Prompting separates the process of task decomposition and solving each sub-task -- can more effectively teach each skill

- Unlike concurrent work, allows for rich decomposition programs (e.g. hierarchical decomposition, recursion) with multiple sub-task handlers (*tools/plugins*)

- Future work:
  - Using DecomP for other complex tasks such as supporting documents for LLM generations, correcting consistency issues
  - Composing multiple small LLMs to achieve scores comparable to GPT3-scale models
  - Zero-Shot DecomP

Questions/Thoughts: **tushark@allenai.org**

Related Work

Prompting

- ReAct
- Program-of-Thought
- Program-Aided Language Models
- Least-to-most Prompting
- Successive Prompting

Key Difference:

A task-independent approach that can use any number of tools and only requires few-shot prompting to iteratively decompose any task

Conclusions

https://github.com/allenai/decomp

- As LMs get larger and only usable behind APIs, augmenting them with tools to circumvent their shortcomings becomes more critical
  - Capabilities of these LMs and the nature of these tools will keep changing, but the fundamental problem still remains
- Focus so far has largely been on fixing issues with knowledge (*hallucination*) and symbolic computation
- Still many other open issues:
  - Ensuring consistency in output (Tool: Consistency Checker)
  - Providing provenance for generations (Tool: Fact Verifier)
  - Multi-modality (Tool: Vision programs)
  - …