Reasoning

Ramaseshan Ramachandran

INTRODUCTION I

- LLMs have transformed artificial intelligence, extending beyond traditional NLP.
- They learn knowledge and reasoning from data, unlike formal logic systems that struggled with real-world complexity.
- LLMs' ability to acquire knowledge opens up a new avenue in reasoning.
- They may lead to verifiability in many tasks, such as legal analysis, scientific discovery, $\cdots$.
- Traditional reasoning in AI involves applying structured rules to derive conclusions, while LLM-based reasoning integrates natural language understanding and enables multi-step deduction and abstraction.
- Algorithmic reasoning may lead to multi-step thought processes, which may enhance the clarity and trustworthiness of LLM outcomes.

INTRODUCTION II

- Logical, common-sense, and mathematical reasoning are crucial for AI systems
- Provide analytical skills. Formal and symbolic logic-based reasoning will be distinguished from heuristic approaches like Chain-of-Thought prompting.
- The attention mechanism may facilitate coherent thought generation.
- Empirical evidence suggests a correlation between LLM scale and reasoning abilities.
- Parameters and training data (AKA scale) is critical for unlocking reasoning potential. LLM Performance is directly proportional to scale (demonstrated - ELMO$\rightarrow$ transformer)
- Deep dive into into the theoretical framework is needed to improve the model design, training, and prompting strategies.

What it is: Starting with a general rule and applying it to a specific situation to reach a certain conclusion.

Simple Example

- **General Rule:** All dogs bark.
- **Specific Case:** Fido is a dog.
- **Deductive Conclusion:** Therefore, Fido barks.

Generally good at simple deductions, but struggle with complex, multi-step logic or strict coherence in long arguments.

INDUCTIVE REASONING (BOTTOM-UP LOGIC)

Forming a general rule or conclusion from specific examples or observations. This conclusion is likely, but not necessarily true.

Simple Example

- **Observation 1:** I am a saggitarian. I spill coffee ;)
- **Observation 2:** My uncle is a saggitarian and he spills coffee
- **Inductive Conclusion:** Therefore, many saggitarians probably spill coffee.

LLMs excel at generalizing from observed patterns but might not invent entirely novel hypotheses far beyond their training data.

ABDUCTIVE REASONING (MAKING THE BEST GUESS)

Observing an outcome and guessing the most likely cause based on incomplete information, like a detective making an educated guess.

Simple Example

- **Observation:** The street outside is wet.
- **Possible Explanations:** It rained, a street cleaner went by, someone spilled water.
- **Abductive Conclusion (Best Guess):** The most likely explanation is that it rained (based on common occurrences).

Suggest plausible explanations based on data correlations, but lack true understanding of causality and context, limiting reliability in complex situations.

Comparing similar things or situations to learn, infer, or explain something about one of them.

Simple Example

Just like a **seed** needs soil and water to grow into a *plant*, a *child* needs care and education to grow into a capable *adult*.

Capable of generating basic analogies based on linguistic similarity, they may miss deeper, conceptual parallels due to a lack of real-world understanding of the underlying relationships.

COMMON SENSE REASONING

LLMs demonstrate common-sense knowledge through their pretraining data and techniques like Chain-of-Thought prompting. External knowledge bases can enhance LLMs' performance on common-sense tasks by providing context. LLM performance on common-sense tasks is assessed using benchmarks like CommonsenseQA, StrategyQA, HellaSWAG, PIQA, Social IQA, and OpenBookQA. LLMs struggle with common-sense reasoning, showing less improvement than logical or mathematical tasks, especially in smaller models. LLMs' knowledge for common-sense answers is often unreliable, potentially incorrect, and misleading. LLMs exhibit significant performance variations based on cultural context, highlighting potential biases in their understanding of the world.

MATHEMATICAL REASONING

- Mathematical thought is a critical capability for artificial intelligence.
- These tasks include algebraic manipulation, solving numerical problems, and even contributing to theorem proving.
- Techniques like Chain-of-Thought prompting have been effective in improving the performance of LLMs on mathematical problems.
- GSM8K contains grade-school level math word problems, while MATH includes problems from high school mathematics competitions.
- Other benchmarks like JEEBench and MATH 401 further test the capabilities of these models on more advanced mathematical problems.
- LLMs can be easily distracted by irrelevant information in mathematical word problems, indicating a lack of ability to discern essential details from extraneous ones.

CHAIN-OF-THOUGHT I

- Chain-of-Thought (CoT) prompting is a framework for step-by-step problem-solving in LLMs.
- It involves providing examples within the prompt that demonstrate the process of reasoning through intermediate steps.
- The LLM is encouraged to generate a similar sequence of thought steps when faced with new problems.
- CoT allows models to break down complex problems into smaller, more manageable parts.
- Standard CoT prompting has limitations, such as not always guaranteeing correct reasoning paths and being sensitive to specific examples chosen for the prompt.
- Recursion of Thought explores divide-and-conquer strategies for multi-context reasoning.
- Meta Chain-of-Thought aims to teach LLMs the general strategy of how to think by providing examples of effective reasoning processes.

CHAIN-OF-THOUGHT II

- Multi-agent strategies combine CoT with the idea of using multiple specialized agents that collaborate to solve complex problems.
- CoT's effectiveness depends on the quality of prompts.
- Research explores sophisticated variations to overcome limitations and improve reasoning capabilities.
- CoT works by providing a template for thought, highlighting the model's ability to learn and apply patterns.
- Decomposition of problems into smaller steps reduces complexity and makes it easier for the LLM to arrive at a correct answer.
- CoT has spurred research into prompt engineering as a method for enhancing LLM capabilities without retraining or fine-tuning.
- Self-Consistency Decoding is a strategy to improve LLM reasoning reliability by generating multiple diverse reasoning pathways and aggregating results to select the most consistent answer.

CHAIN-OF-THOUGHT III

- Self-consistency involves sampling a set of different reasoning paths from the LLM's decoder.
- The final answer is chosen by taking a majority vote across the set of answers produced by diverse reasoning paths.
- This approach is based on the idea that for complex problems, there are often multiple valid ways to arrive at the correct solution.
- If several different reasoning paths converge to the same answer, there is a higher likelihood that the answer is indeed correct.
- Self-consistency improves LLM reasoning accuracy and robustness.
- It samples multiple reasoning paths to mitigate greedy decoding limitations.
- Self-consistency reduces reliance on single sampled generation.
- It is an unsupervised method that can be applied directly to pre-trained LLMs.
- Self-consistency enhances the reliability of LLM reasoning.

CHAIN-OF-THOUGHT IV

- It leverages the idea that multiple independent reasoning processes increase confidence.
- Self-consistency builds upon Chain-of-Thought by ensuring consistency across multiple chains.
- It reduces the impact of flawed reasoning steps by sampling multiple paths and using a voting mechanism.
- Self-consistency offers a valuable approach to improve existing LLMs without additional training data or resources.
- Integration of External Knowledge:
- Retrieval-Augmented Generation (RAG) is a framework that enhances the reasoning capabilities of LLMs.
- RAG provides LLMs with access to relevant information from external knowledge sources.

CHAIN-OF-THOUGHT V

- The RAG process involves retrieving pertinent knowledge from an external database and using it to augment the LLM's response generation.
- RAG can address limitations of LLMs, such as hallucination, outdated information, and lack of specialized knowledge.
- The typical RAG workflow consists of indexing the external knowledge source, retrieving relevant information, and generating a response that incorporates the retrieved knowledge.
- Advanced RAG techniques focus on optimizing each stage of the workflow, such as fine-tuning knowledge segmentation during indexing, enhancing retrieval through query optimization, and improving the processing and utilization of retrieved information by the LLM.
- Knowledge Graphs (KGs) are structured knowledge repositories that store factual information as entities and their relationships.

CHAIN-OF-THOUGHT VI

- Integrating KGs with LLMs can enhance their reasoning abilities by providing a structured source of external knowledge.
- Various approaches have been explored for this integration, including methods that fuse representations learned from both the LLM and the KG.
- Frameworks where the LLM acts as an agent that can interact with and query the KG to perform reasoning have also been explored.
- GuideKG is a method that uses LLMs to generate knowledge-based explanations to improve the common-sense reasoning of smaller language models.
- KG-GPT is a framework designed to leverage the capabilities of LLMs for complex reasoning tasks that involve Knowledge Graphs.
- Effective integration of external knowledge requires careful consideration of how to elicit, filter, and seamlessly incorporate this information into the LLM's thought process.

- Retrieval techniques aim to identify the most relevant documents, passages, or structured data based on the input query.
- Once retrieved, this knowledge can be integrated into the LLM's prompt in various ways, such as by directly inserting relevant text or by using embeddings to represent the knowledge in a continuous vector space that the LLM can then process.
- RankRAG is a novel framework that instruction-tunes a single LLM to perform both the task of ranking retrieved contexts based on their relevance and generating an answer using these contexts.
- LINKED proposes a process of eliciting knowledge from LLMs, filtering it for accuracy and relevance, and then integrating it to improve their common-sense reasoning abilities.

- The integration of external knowledge through Retrieval-Augmented Generation and the utilization of Knowledge Graphs can address the limitations of LLMs' internal knowledge.
- These techniques can enhance LLMs' reasoning capabilities in tasks that heavily rely on factual information.
- By providing LLMs with access to up-to-date and domain-specific knowledge, these techniques can address issues of hallucination and outdated information.
- Grounding the LLM's reasoning in verifiable external sources is crucial for increasing confidence in its outputs, especially in sensitive or critical domains.

- **Reasoning with Mathematics:** Mathematics provides a universal framework for solving problems through structured logic and quantitative analysis. It's about breaking down complex questions into manageable parts using numbers, symbols, and rules—think of it as a mental toolkit for precision.
- **Relevance to Computer Science:** In CS, mathematical reasoning underpins everything: designing efficient algorithms, proving correctness, optimizing systems, and modeling data structures like stacks. It's the engine driving computational thinking.
- **Focus of This Talk:** We'll explore stack permutations—a classic CS problem where we push numbers (0 to 9) in order and pop them to match a target sequence, like 4568793210. It's a perfect sandbox to test reasoning skills.
- **Preview:** We'll dissect this using logical reasoning (step-by-step proofs) and touch on probabilistic reasoning (pattern-based guesses), showing how Grok 3 adapts to such challenges.

- **Definition:** Mathematical reasoning is the process of applying logical principles to numerical or symbolic problems, moving from premises to conclusions. It's about clarity—every step must hold up under scrutiny.
- **Basic Example:** Consider: "How many hours to earn $120 at $15 per hour?" Define the total cost $C = 120$, rate $r = 15$, and time t. Then:

$$C = r \cdot t \implies 120 = 15 \cdot t \implies t = \frac{120}{15} = 8 \text{ hours.}$$

  This is reasoning distilled: identify variables, set up an equation, solve systematically.
- **Formal Notation:** Take a linear function $f(x) = ax + b$. If $a = 15$ (rate), $b = 0$ (no base pay), then $f(x) = 15x$ models earnings over x hours. Plug in $f(t) = 120$, and solve: $15t = 120$, $t = 8$.
- **Why It Matters:** This foundational skill scales to complex CS problems—like optimizing code or proving theorems—building intuition for patterns and deductions.

- **Core Concepts:** Discrete mathematics deals with countable, distinct objects—unlike continuous math (e.g., calculus). It's the backbone of CS, covering:
  - **Sets:** Collections like $A = \{1, 2, 3\}$, $B = \{2, 4\}$. Operations model data grouping.
  - **Logic:** Statements like $P \wedge Q$ (P and Q) govern program conditions.
  - **Graphs:** $G = (V, E)$, with vertices V and edges E, represent networks or dependencies.
- **Example—Set Union:** Compute $A \cup B$:

$$A \cup B = \{1, 2, 3\} \cup \{2, 4\} = \{1, 2, 3, 4\}.$$

  This operation might model merging user lists in a database—simple, yet powerful.
- **CS Connection:** Stacks (our focus) rely on discrete sequences, logic dictates their behavior, and graphs could model push/pop dependencies. Discrete math is the language of computation.

- **Definition:** A stack is a data structure $S = [s_1, s_2, ..., s_n]$, where elements are added and removed from one end—the top. It's like a stack of plates: you add to the top, take from the top.
- **Operations:**
  - **Push:** Add x to the top: $S \leftarrow S \cup \{x\}$, so x becomes s_{n+1}. E.g., $S = [1, 2]$, push 3, $S = [3, 2, 1]$.
  - **Pop:** Remove and return the top element s_n. E.g., $S = [3, 2, 1]$, pop returns 3, $S = [2, 1]$.
- **Math:** The top is:

$$\text{top}(S) = s_n,$$

  reflecting the Last-In, First-Out (LIFO) principle—last pushed, first popped.
- **Relevance:** Stacks model function calls, undo operations, and permutations (our case), making them a CS staple.

- **Problem Setup:** Given an input sequence $I = [0, 1, 2, ..., 9]$ (ascending order), we push these onto a stack and pop them to produce an output $O = [o_1, o_2, ..., o_{10}]$.
- **Goal:** O must be a permutation of I (all 10 numbers, used once), achievable via valid stack operations—push in order, pop anytime, print on pop.
- **Formal Condition:**

  O is a permutation of I if there exists a sequence of pushes and pops s.t. popping yield
- **Example:** $O = 4568793210$. We'll test if pushing 0-9 and popping strategically matches this. It's a puzzle: can LIFO rearrange 0-9 into this order?

- **Objective:** Verify if $O = 4568793210$ is stack-sortable by simulating push/pop operations on $I = [0, 1, ..., 9]$.
- **Detailed Steps:**
  - Push 0-4: $S = [4, 3, 2, 1, 0]$, pop 4 (output: 4).
  - Push 5: $S = [5, 3, 2, 1, 0]$, pop 5 (output: 45).
  - Push 6: $S = [6, 3, 2, 1, 0]$, pop 6 (output: 456).
  - Push 7-8: $S = [8, 7, 3, 2, 1, 0]$, pop 8, then 7 (output: 45687).
  - Push 9: $S = [9, 3, 2, 1, 0]$, pop 9, then 3, 2, 1, 0 (output: 4568793210).
- **Result:** The sequence matches exactly after 10 pushes and 10 pops, respecting LIFO—each pop was the top element at that moment.

- **Stack-Sortable Condition:** A sequence is stack-sortable if it avoids a "231 pattern" in relative order—where $a < c < b$, but b appears before c, and a after, making a unreachable behind c.
- **Math Example:** In 231, push 1, 2, 3 ($S = [3, 2, 1]$), pop 2 ($S = [3, 1]$), need 3 but 1 is on top—impossible.
- **Check 4568793210:**
  - Positions: 4, 5, 6, 8, 7, 9, 3, 2, 1, 0 (index in I).
  - Test 8, 7: $8 > 7$, 7 after 8—stack had $[8, 7, ...]$, pop 8, 7 is top, valid.
  - No 231: E.g., 4, 6, 5 (4 < 6 < 5)? No, order is 4, 5, 6. No violations found.
- **Insight:** Absence of 231 suggests sortability, but we need more.

- **Accessibility Condition:** At each step i, the next pop o_i must be either the top of the stack or the next number to push:

$$o_i = \text{top}(S) \text{ or } o_i = I[k] \text{ (next in } I\text{).}$$

- **Apply to 4568793210:**
  - $o_1 = 4$: Push 0-4, $S = [4, 3, 2, 1, 0]$, top $= 4$, pop 4.
  - $o_4 = 8$: Push 7-8, $S = [8, 7, 3, 2, 1, 0]$, top $= 8$, pop 8.
  - $o_7 = 3$: After pops, $S = [3, 2, 1, 0]$, top $= 3$, pop 3.
- **Validation:** Every o_i was accessible—either freshly pushed or left at the top by prior pops. Combined with no 231, this proves sortability.

- **Simulation Analysis:** For n elements, we perform at most n pushes (one per number) and n pops (one per output digit).
- **Time Complexity:**

$$T(n) = n_{\text{push}} + n_{\text{pop}} \leq n + n = 2n \implies O(n).$$

  Each operation (push/pop) is $O(1)$ in a stack, so total time is linear.
- **Example:** For 4568793210, $n = 10$:
  - 10 pushes (0-9), 10 pops (4, 5, ..., 0).
  - $T(10) \approx 20$ operations, confirming $O(n)$.
- **Significance:** Efficient verification makes stack sorting practical for larger inputs, a key CS concern.

- **How LLMs Operate:** Language models like Grok 3 predict the next token w_t based on prior context $w_{1:t-1}$, using a softmax probability:

$$P(w_t|w_{1:t-1}) = \frac{e^{s_t}}{\sum_j e^{s_j}},$$

  where s_t is a score for w_t. It's statistical, not logical.
- **Example:** Asked "Is 4568793210 valid?" I might guess: "It's 10 digits, uses 0-9, likely valid," based on pattern recognition from training data.
- **Limitation:** No proof—just a hunch. I could hallucinate "invalid" if skewed examples dominate. Here, I avoided this, opting for logic.

- **Deductive Form:** Logic follows: "If all x satisfy $P(x) \to Q(x)$, and $P(a)$ holds, then $Q(a)$ holds."

$$\forall x(P(x) \to Q(x)), \quad P(a) \vdash Q(a).$$

  It's certain, not probable.
- **Stack Example:**
  - $P(x)$: x follows LIFO rules (push/pop correctly).
  - $Q(x)$: x is stack-sortable.
  - For 4568793210: P (LIFO holds via simulation) $\to Q$ (sortable).
- **Power:** Deduction guarantees correctness—vital for CS proofs like this.

- **Definition:** CoT breaks problems into explicit, sequential steps, mimicking human reasoning. It's about showing the journey, not just the destination.
- **Math Representation:** Stack simulation as transitions:

$$S_t \xrightarrow{\text{push/pop}} S_{t+1}, \quad \text{e.g., } [4,3,2,1,0] \xrightarrow{\text{pop 4}} [3,2,1,0].$$

  Each S_t is a state, each move a reasoned choice.
- **Example 4568793210:** Push 0-4, pop 4 ($S = [3,2,1,0]$); push 5, pop 5; up to push 9, pop 9-0. Each step asks: "Does this get me closer?"
- **Used Here:** CoT ensured no leaps—every pop was justified, boosting accuracy.

- **Definition:** These are implicit, self-generated questions guiding CoT—like "What's next?" or "Is this valid?" They're not user-provided but emerge to structure reasoning.
- **Math:** Sub-goals as:

$$S_t \to o_i, \quad \text{e.g., } [4,3,2,1,0] \to 4 \text{ (pop 4)}.$$

  Each transition reflects a prompt like "Can I pop o_i now?"
- **Example 4 5 6 8 7 9 3 2 1 0:**
  - "What's first?" $\to$ Push 0-4, pop 4.
  - "Next is 5?" $\to$ Push 5, pop 5.
  - "Can I get 3?" $\to$ After pops, $S = [3,2,1,0]$, pop 3.
- **Role:** These prompts drove the CoT, keeping me systematic without guessing.

- **Deductive Reasoning:** From rules to facts:

$$A \to B, \quad A \text{ true} \Rightarrow B \text{ true}.$$

  E.g., “LIFO holds $\to$ 4 5 6 8 7 9 3 2 1 0 is valid.” It’s certain, ideal for proofs.

- **Probabilistic Reasoning:** From patterns to likelihood:

$$P(B|A) \approx 0.9, \quad \text{may be false}.$$

  E.g., “Sequence looks valid—9 0

- **When to Use:** Deductive for CS/math precision; probabilistic for creative or ambiguous tasks.

- **Summary:**
  - **Logic + CoT + Prompts:** Combined for precise stack solutions—simulation and theory proved 4 5 6 8 7 9 3 2 1 0 valid.
  - **Probability:** Offers flexibility for less rigid tasks, but sidelined here for rigor.
- **Most of AI chatbot's Approach:** adapt reasoning to the task—logical for maths problems, probabilistic for chat
- **Future Directions:** Apply this to more math/CS challenges—graph algorithms, optimization, or beyond.