# Dual-System Language Models via Next-Action Prediction

**Zhehang Du** [*1] **Weijie Su** [1]

## Abstract

In current Large Language Model (LLM) practices, each token is appended sequentially to the output. In contrast, humans are capable of revising and correcting what we write. Inspired by this gap, in this position paper, we propose a dual-system to simultaneously model the thought process and the output process via the introduction of action tokens. This is achieved by (a) maintaining two sequences of tokens, which include a thought system simulating the human thought process and an output system for storing responses, and (b) introducing removal tokens as action tokens: when a removal token is generated, it is appended only to the thought system, while simultaneously removing certain tokens from the output system. The model uses both systems for next-action prediction. This method allows the retraction of previously generated tokens in the final response and maintains a record of intermediate steps in the thought system. Our framework enables the training of language models to improve the interaction between the thought and output systems, mirroring the way humans refine their thinking for effective written communication. Moreover, it can be implemented with slight modifications to existing LLM architectures and allows for end-to-end training.

## 1. Prologue

*This paper proposes a novel design for LLMs with the potential to improve their reasoning and planning abilities. We do not present empirical results at this stage. Implementing our dual-system design requires a model capable of simultaneously processing two sequences and applying timestamp-based masking – functionalities unsupported by current open-source LLMs. This makes fine-tuning infeasible. Training a suitable LLM from scratch exceeds the computational resources of most researchers. Consequently, we present this work as a position paper to share the concept broadly, inspiring and encouraging future research with the necessary resources to empirically validate the approach.*

## 2. Introduction

In the landscape of artificial intelligence, Large Language Models (LLMs) have showed remarkable capabilities in understanding and generating human language. By scaling up the model size, models such as GPT (Achiam et al., 2023; Brown et al., 2020), PaLM (Chowdhery et al., 2023; Anil et al., 2023), and LLaMA (Touvron et al., 2023a;b) have already shown emergent capabilities that involve reasoning (Wei et al., 2022a; Huang & Chang, 2022). Despite reaching human-level performance on a wide variety of tasks, they still lack planning abilities and often make apparent mistakes in the intermediate reasoning steps that are easily spotted by humans (Bubeck et al., 2023).

To enhance LLM capabilities, many existing methods focus on augmenting standard models with separate *planning and reasoning modules*, including problem decomposition (Cobbe et al., 2021), prompting (Luo et al., 2023), self-consistency (Wang et al., 2022), chain-of-thought and tree-of-thought (Wei et al., 2022b; Yao et al., 2023; Long, 2023), self-refinement (Madaan et al., 2023), Reflexion (Shinn et al., 2024), agentic approaches such as WebGPT (Nakano et al., 2021) and LangChain (Chase), and optimizing reasoning paths by search algorithms (Trinh et al., 2024), some with reinforcement learning methods (Ma et al., 2023; Zhou et al., 2023; Liu et al., 2023). Others aim to directly learn strong policies (Ruoss et al., 2024). However, prompt-based and search-based methods often rely on external guidance. While RL is powerful with clear rewards (e.g., in games like Go (Silver et al., 2017)), defining such rewards for abstract reasoning remains a challenge. Additionally, common approaches like RLHF demand extensive human effort (Ouyang et al., 2022). The ability of LLMs to self-correct also remains limited (Huang et al., 2023; Pan et al., 2023).

A common method for understanding the limitations of current LLMs is to compare them with the two-systems theory of human cognition (Sloman, 1996). System 1 is

[*]Equal contribution [1]University of Pennsylvania, Philadelphia, PA, USA. Correspondence to: Zhehang Du <duz@wharton.upenn.edu>.

*Proceedings of the 41[st] International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

fast, automatic, and instinctive, guiding our quick decisions and emotional reactions. System 2 is slower, more deliberate, and conscious, used for complex problem-solving and logical reasoning. In contrast, LLMs generate each token with nearly a fixed amount of computation, which resemble System 1. Without the ability to allocate additional computational resources for complex reasoning tasks, LLMs tend to quickly produce responses based on statistical language patterns acquired during training. This "forces" them outputting their thoughts directly without any internal planning process.

Recent works suggest that we can further enhance models' capabilities by embedding the search dynamics into the model weights directly (Lehnert et al., 2024; Gandhi et al., 2024). In this position paper, We aim to propose a computational model of reasoning and planning inspired by analogous processes in humans, where the iterative process of writing, revising, and editing is a well-established approach to creating effective written work. We identify two limitations in prevalent LLM practices: the lack of (a) interaction with *external states*, and (b) the capability to perform *actions*. For example, when solving complex problems, humans can write intermediate steps on paper while mentally processing the entire sequence of thoughts. We are capable of erasing incorrect steps on the paper and returning to an earlier stage, with experience gained from unsuccessful attempts. Such a trial-and-error process is essential for both intelligent development and practical problem-solving, yet current LLMs are mostly limited to sequentially appending tokens to create the final response. In this analogy, the paper represents the *external states*, while the process of removing unsuccessful attempts represents the *action*.

To bridge this gap, we propose a novel framework that empowers language models to utilize and interact with external states and execute actions. Our goal is twofold: firstly, to refine the *architecture design*; and secondly, to optimize the *training pipeline*. Our dual-system framework mainly exhibits two differences compared to current LLMs:

1. **Dual Token Sequences:** Unlike current LLMs that rely on a single sequence of tokens, our framework uses multiple sequences collaboratively to generate the final response. One sequence acts like a fast, intuitive thought process, mirroring System 1 thinking. This sequence can be seen as the internal generation of ideas. Simultaneously, another sequence accumulates the refined output, similar to how we translate thoughts into coherent language.

2. **Next-Action Prediction:** In standard LLMs, the generation process is linear, with each token being appended sequentially to the final output. Our framework expands the role of tokens to include control, modification, and decision-making within the model, and each token defines how the states of these token sequences transition to the next state. This mirrors how humans review their writing and combine it with internal thoughts to decide the next action. Therefore, in our framework, "next-token prediction" can also be called "next-action prediction".

In this paper, we mainly focus on removal tokens as action tokens, which are used to delete previously generated tokens from the end of the output sequence while retaining them in the thought process. Similar concepts, such as the backtrack token, have been explored empirically (Cundy & Ermon, 2023). Expanding on this concept, we also introduce additional action tokens. For instance, by using cursor tokens, the model has the ability to control a cursor within the output, facilitating the insertion or deletion of tokens at any point in the output. By strategically using learning tokens at the end of dialogues, the model can autonomously trigger back-propagation, adjusting its own parameters whenever it encounters particularly insightful dialogues. By introducing a global state that dictates whether a newly generated token is appended to the final output and alter tokens to alter this global state, the model can now decide on its own whether a newly generated token belongs in the thought system or the final response. By using highlight tokens, the model can focus on important points and make more accurate predictions or decisions based on the highlighted information. By using comment tokens, the model can output a more detailed thought process. By using asking-a-simpler-question tokens, the model is encouraged to decompose complex questions into simpler, more manageable parts. By using pause tokens (Goyal et al., 2023), the model can strategically delay the response.

Training language models within our framework is not easy, in part due to the lack of existing datasets that accurately reflect the process underlying the text production. However, in Appendix A, we will provide several possible strategies to overcome this hurdle and several avenues that could be readily implemented within our new framework. We believe that our approach holds the potential to improve the capabilities of language models.

## 3. The Dual-System Model

### 3.1. Preliminaries

**Tokens.** Tokens are the building blocks that language models use to understand and generate text. Modern language models typically employ subword units as tokens to represent words, along with various special tokens for specific functions. These special tokens are traditionally used for language construction, and we refer to them as *language tokens*. However, this paper introduces a novel category, *action tokens*, designed for decision-making and control within the model. In this paper, we use the following special tokens: end of a sentence `<EOS>`, end of model response `<END>`, and markers differentiating user input `<USER>` from model output `<MODEL>`. We also allow language models to interact with code environments using the `<PYTHON>` token. Additionally, we will introduce removal tokens as action tokens. It is important to note that the specific special tokens might vary in actual LLM implementations; the tokens outlined here are defined only for the sake of demonstration.

**Next-Token Prediction.** In an autoregressive language model, each token is predicted based on the tokens that precede it. For instance, the $i$-th token $x_i$ is sampled from $P_\theta(x_i|x_1, x_2, \ldots, x_{i-1})$, where $\theta$ denotes the model parameters.

### 3.2. Approach

We showcase a straightforward implementation of language models within our framework, utilizing a dual-system model and removal tokens. First, we introduce our dual-system model. This model comprises two systems: the thought system, denoted by $T$, which simulates the human thought process by keeping the complete sequence of intermediate steps, and the output system, denoted by $S$, which stores the final response, similar to written communication. Each system holds a sequence of tokens that are dynamically updated based on user input and model inference. We define a set of removal tokens as action tokens: {`<RMV-T>`, `<RMV-S>`, `<RMV-R>`, ...}. The functionalities of these removal tokens are as follows:

- `<RMV-T>` (Removal of Generated Tokens): This token triggers the removal of the last $n$ tokens in the output system $S$.
- `<RMV-S>` (Removal of Sentence): This token leads to the removal of the most recent sentence in the output system $S$, identified by the `<EOS>` token (or the `<MODEL>` token for the first sentence).
- `<RMV-R>` (Removal of Current Response): This token is used to remove the current response tokens generated by the model in the ongoing conversation in the output system $S$, identified from the last occurrence of the `<MODEL>` token to the present point, while preserving previous completed dialogs.

The removal tokens are generated during the model's inference step. For user inputs and all tokens, except for the removal tokens, they are appended to both the thought system $T$ and the output system $S$. Removal tokens, however, are appended only to $T$ and are responsible for triggering the removal of tokens in $S$. The removal tokens play a dual role: in the thought system $T$, they add context and depth to the ongoing internal deliberation of the model; in the output system $S$, they act as a corrective mechanism. During inference, the generation of the next token $x_i$ is determined by both $T$ and $S$, with the model parameters $x_i \sim P_\theta(x_i|T, S)$. At the end of a model's response, only the newly generated tokens in the output system $S$ are converted into text and subsequently presented to the user.

### 3.3. Example

To demonstrate, here is an example of the model's intended behavior when prompted by a user to solve the equation $6x = 4(x + 3)$ step-by-step.

User: Can you solve $6x = 4(x + 3)$ step-by-step?

Thought system $T$:

`<USER>` Can you solve $6x = 4(x + 3)$ step-by-step? `<EOS>`
`<MODEL>` Let's solve it step-by-step. First, expand the right-hand side: $6x = 4x + 12$. `<EOS>`
Next, subtract $4x$ from both sides to get $2x = 4$. `<EOS>`
Dividing both sides by 2 gives $x = 2$. `<EOS>`
Checking the solution: Substituting $x = 2$ into the equation, we get $6 \times 2 = 12$ and $4 \times (2 + 3) = 20$, so $6 \times 2 \neq 4 \times (2 + 3)$. `<EOS>`
This seems incorrect. Let me try again. `<EOS>`
`<RMV-R>`
Let's solve it step-by-step. First, expand the right-hand side: $6x = 4x + 12$. `<EOS>`
Subtract $4x$ from both sides to get $2x = 12$. `<EOS>`
Now, divide both sides by 2 to find $x = 6$. `<EOS>`
Checking the solution: Substituting $x = 6$ into the equation, we get $6 \times 6 = 36$ and $4 \times (6 + 3) = 36$, so $6 \times 6 = 4 \times (6 + 3)$. This seems correct. `<EOS>`
`<RMV-S>`
Therefore, the solution is $x = 6$. `<EOS>`
`<END>`

Output system $S$:

`<USER>` Can you solve $6x = 4(x + 3)$ step-by-step? `<EOS>`
`<MODEL>` Let's solve it step-by-step. First, expand the right-hand side: $6x = 4x + 12$. `<EOS>`
Subtract $4x$ from both sides to get $2x = 12$. `<EOS>`
Now, divide both sides by 2 to find $x = 6$. `<EOS>`
Therefore, the solution is $x = 6$. `<EOS>`
`<END>`

Model: Let's solve it step-by-step. First, expand the right-hand side: $6x = 4x + 12$. Subtract $4x$ from both sides to get $2x = 12$. Now, divide both sides by 2 to find $x = 6$. Therefore, the solution is $x = 6$.

In this dialogue, a user asks the model to solve the equation $6x = 4(x + 3)$ step by step. Initially, the model correctly expands the equation but makes an error in subsequent steps, leading to an incorrect solution of $x = 2$. Upon verifying its solution and finding a discrepancy, the model uses the `<RMV-R>` token to remove its incorrect response. It then correctly solves the equation on a second attempt, arriving at $x = 6$, which is verified and confirmed to be accurate. Then, the `<RMV-S>` token is used to remove the sentences that checked the answer before the conversation is concluded.

### 3.4. Architecture

To construct the neural network within our dual-system framework using the transformers architecture (Vaswani et al., 2017), we propose two architectural designs, as shown in Figure 1.

**The Dual Architecture.** This architecture first feeds the token sequences $T$ and $S$ into two separate decoders. To effectively learn the interaction between these systems, we recommend selectively applying cross-attention between the decoders. Specifically, in the output-priority architecture (Figure 1 on the left), the output system's decoder integrates information from the thought system's decoder. This may benefit tasks where the final output needs to be highly refined and informed by complex thought processes. Conversely, in the thought-priority architecture (Figure 1 on the right), the process is inverted, with the output system's decoder providing context to the thought system's decoder. This approach may be advantageous where creative responses are required, as it emphasizes the role of internal thought processes in shaping the output. Further decoder blocks are used to synthesize the information from both systems to compute the final output probability.

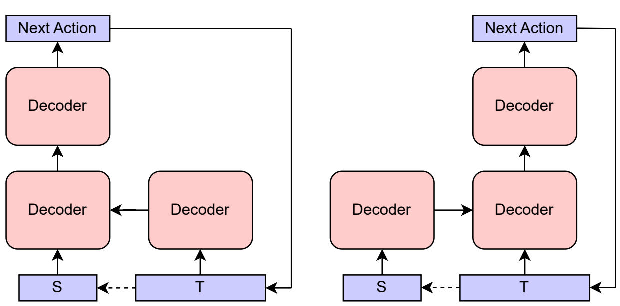


*Figure 1.* A simple illustration of the two architectures. Left: the output-priority architecture; right: the thought-priority architecture.

**Masking.** In the decoder blocks, we need to ensure that each token only depends on the tokens generated before it by applying masking. We can implement masking by assigning a timestamp to every token added to $S$ and $T$, starting the count from one. Since $T$ logs all tokens, its timestamp consistently increments from one. However, due to the potential removal of tokens in $S$, there may be jumps in its counter. The mask is then set based on these timestamps.

**Potential Limitations.** The proposed architectures have not yet been empirically tested, so further experiments are necessary to evaluate their performance. Additionally, alternative architectural designs are also open for exploration, such as using a single encoder to model both $S$ and $T$. However, it should be noted that in our system, the length of the sequence in the output system $S$ can vary significantly due to removal tokens. A sudden deletion of tokens in $S$ during inference time can potentially cause distribution shifts, potentially impacting its performance. Therefore, these potential issues might require additional engineering efforts to resolve.

### 3.5. Generalization

In a high-level overview, our dual-system model can be viewed as a coordinated structure consisting of two systems, $T$ and $S$. These systems interact at each model inference step, with new actions $x_i$ generated based on the states of both $T$ and $S$, and the model parameters $\theta$: $x_i \sim P_\theta(x_i|T, S)$. The actions, represented by $x_i$, are no longer confined to the task of constructing text; they now facilitate communication and interaction between systems $T$ and $S$, conveying instructions, feedback, and decisions. Since $S$ is a sequence of tokens, the same as $T$, we can integrate both $T$ and $S$ in the next-action prediction with only slight modifications to current LLM architectures. During training, this integration enhances language generation and also improves the coordination and control between the two systems.

In our dual-system model, $S$ functions as a simple text editor, limited to only appending and removing characters at the end of the string. To extend the applicability of our framework, one could incorporate a *varying number of systems* (for example, using a summary system to address long-range dependency issues), diverse *types of action* (such as enabling copy, paste, insert, search, and substitute functions), and the inclusion of *external tools* (like integrating search engine data retrieval, code execution, or interfacing with a storage system).

## 4. Potential Benefits of the Dual-System

Our framework shares similarities with Scratchpad (Nye et al., 2021) that utilizes a scratchpad to showcase intermediate computations and Self-Notes (Lanchantin et al., 2024), which generates multiple internal reasoning notes incorporating both the input context and question. Similar to our approach of modeling the thought process and final response separately, these methods also interleave input and reasoning steps. A key difference lies in our proposed architecture, which leverages two separate token sequences instead of the conventional one-sequence approach.

**Synergistic Environment.** Our framework enables training that improves synergy between multiple token sequences, mirroring the refinement of internal thought (one sequence) for better output (the other sequence). This expanded environment has the potential to improve model behavior. Our core belief is that an agent's intelligence is intrinsically connected to the environment in which it learns.

**Focus on Thought Process.** We prioritize predicting the internal thought process rather than the final output. Since neural networks are better at fitting functions of lower complexity, modeling fast-thinking (i.e., the thought process) is generally simpler than directly modeling deliberate, complex reasoning. Therefore, our framework has the potential to achieve reasoning capabilities equal to those of larger current LLMs. Additionally, this thought sequence acts as a latent space where the model can plan internally before generating the final response.

**Unified Approach.** Our framework offers a unified model capable of self-planning, self-evaluation, and learning from mistakes. All components are integrated, requiring only minor modifications to existing LLM architectures and allowing for end-to-end training.

### 4.1. Advantages of Dual-Sequence over Single-Sequence

While it is true that the output $S$ can be uniquely derived from $T$, maintaining only $T$ for final output generation presents limitations. We believe a dual-system offers more advantages over single-sequence approaches. Here is how single-sequence models might handle deleted tokens, and the potential drawbacks:

**Masking Deleted Tokens Leads to Repetition of Errors.** If we ignore the deleted tokens by setting masking to the deleted tokens, it limits the model's ability to learn from its mistakes. Newly generated tokens won't see the deleted portions, potentially leading the model to repeat the same wrong path. Our dual-system approach prevents this by maintaining the history of the thought process in sequence $T$. This history informs attention calculations, allowing the model to use past experiences to guide new trials and avoid repeating incorrect paths.

**Including All Tokens Leads to Information Overload.** Including all tokens without masking deleted tokens, along with positional embeddings in one sequence, means the model has to process a large amount of information. Positional embeddings tell the model something about the order of tokens. In this case, a previously generated but now discarded token might still have an embedding representing an earlier position in the sequence. This can lead to confusion about which parts are relevant to the current problem-solving step and which should be ignored, especially when the problem is complex and requires multiple trial-and-error attempts. Our dual-system approach provides a cleaner problem-solving environment by separating the sequences. This mirrors how humans separate drafts from formal solutions when working on complex problems, making the thought process easier to follow.

## 5. Conclusions and Future Work

In this work, (a) we revised the architecture design to allow language models to think and plan internally, selectively output their thoughts, and remove inappropriate content during inference time; and (b) we suggest that intelligent development should focus on teaching models both language and action simultaneously in an interactive way, rather than modeling language alone. Future research could build upon this framework by integrating more types of actions and exploring multi-modal interactions. The ultimate goal is to achieve more human-like capabilities in AI through richer, more interactive learning processes.

## Limitations

However, our framework also has its limitations. The biggest concern is the lack of training data that integrates both language and action, which is essential for the framework's effectiveness. Additionally, the introduction of complex internal mechanisms, such as action tokens and multiple token sequences, complicates the training process, which may present challenges that we are unable to address at this moment. Despite these limitations, we encourage researchers to explore training methods that reflect the development of human intelligence, as our framework offers a potential pathway for addressing several existing challenges on the path to artificial general intelligence.

## Impact Statement

Our proposed framework introduces a new way of conceptualizing and developing intelligent systems by integrating action with language. This framework aims to simulate human cognition and narrow the gap between human and machine intelligence. It offers the possibility of reducing model sizes, addressing computational resource intensity, and enhancing global model accessibility. The framework can be potentially beneficial to various industries and sets a foundation for future research.

However, it is also crucial to address the potential risks of our framework. As AI systems become more advanced in their decision-making processes, there is an increased responsibility to prevent biases, misinformation, or harmful behaviors. Using diverse, ethically sound datasets and conducting rigorous testing are essential to mitigate these risks. From a security standpoint, more complex and interactive AI systems could be more vulnerable to malicious exploitation. Robust security measures are necessary to ensure they are not used harmfully. Additionally, when collecting training data, utmost care must be taken to prevent bias in the data, protect individual privacy, and ensure compliance with data protection regulations.

## References


Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Arora, K., Asri, L. E., Bahuleyan, H., and Cheung, J. C. K. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. *arXiv preprint arXiv:2204.01171*, 2022.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Chase, H. Langchain. URL https://github.com/langchain-ai/langchain. Software available from https://github.com/langchain-ai/langchain.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cundy, C. and Ermon, S. Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking. *arXiv preprint arXiv:2306.05426*, 2023.

Dong, H., Xiong, W., Goyal, D., Pan, R., Diao, S., Zhang, J., Shum, K., and Zhang, T. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

Guo, S., Zhang, B., Liu, T., Liu, T., Khalman, M., Llinares, F., Rame, A., Mesnard, T., Zhao, Y., Piot, B., et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.

Hao, S., Gu, Y., Ma, H., Hong, J. J., Wang, Z., Wang, D. Z., and Hu, Z. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Huang, J. and Chang, K. C.-C. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Lanchantin, J., Toshniwal, S., Weston, J., Sukhbaatar, S., et al. Learning to reason and memorize with self-notes. *Advances in Neural Information Processing Systems*, 36, 2024.

LeCun, Y. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1), 2022.

Lehnert, L., Sukhbaatar, S., Mcvay, P., Rabbat, M., and Tian, Y. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step, 2023.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Long, J. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Luo, Y., Tang, Y., Shen, C., Zhou, Z., and Dong, B. Prompt engineering through the lens of optimal control. *arXiv preprint arXiv:2310.14201*, 2023.

Ma, Q., Zhou, H., Liu, T., Yuan, J., Liu, P., You, Y., and Yang, H. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Pan, L., Saxon, M., Xu, W., Nathani, D., Wang, X., and Wang, W. Y. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2023.

Ruoss, A., Delétang, G., Medapati, S., Grau-Moya, J., Wenliang, L. K., Catt, E., Reid, J., and Genewein, T. Grandmaster-level chess without search. *arXiv preprint arXiv:2402.04494*, 2024.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Sloman, S. A. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3, 1996.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022b.

Xiang, J., Tao, T., Gu, Y., Shu, T., Wang, Z., Yang, Z., and Hu, Z. Language models meet world models: Embodied experiences enhance language models. *arXiv preprint arXiv:2305.10626*, 2023.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models, 2024.

Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., and Huang, F. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

# A. Model Training

## A.1. Training Objective

In our dual-system model with removal tokens, $S$ is uniquely determined by $T$. We define the compression operator $\mathrm{Comp}$ as the transformation of the token sequence in $T$ to the token sequence in $S$.

Consider a specific token sequence in $T$, represented as $x_1, x_2, \ldots, x_N$. We define the compressed output of the text in $S$ as $\hat{x}_1, \ldots, \hat{x}_n$, obtained through the compression operator: $\hat{x}_1 \cdots \hat{x}_n = \mathrm{Comp}(x_1 x_2 \cdots x_N)$. Our dual-system language model formally takes the form:

$$h_{w_3}(f_{w_1}(x_1 x_2 \cdots x_N), g_{w_2}(\hat{x}_1 \cdots \hat{x}_n)),$$

where $f_{w_1}$ processes the input sequence in $T$, $g_{w_2}$ processes the compressed sequence in $S$, and $h_{w_3}$ synthesizes both to produce the final output. Each function is parameterized by $w_1$, $w_2$, and $w_3$ respectively. The loss for this token sequence is defined as:

$$\sum_{m=1}^{N-1} L(h_{w_3}(f_{w_1}(T_{1:m}), g_{w_2}(\mathrm{Comp}(T_{1:m}))), x_{m+1}),$$

where $T_{1:m}$ denotes the subsequence of the first $m$ tokens of $T$, and $L$ denotes a specified loss function. The training objective is to find the parameters $w_1, w_2, w_3$ that minimize the sum of the above loss across all token sequences in the training data.

Apart from training the model using supervised learning (by directly collecting the desired dataset and optimizing the loss above), we can also employ preference data in both offline (Ouyang et al., 2022) and online (Bai et al., 2022) regimes. For instance, reinforcement learning can be used within the classical RLHF framework (Christiano et al., 2017) with PPO (Schulman et al., 2017). Additionally, recently developed methods like DPO (Rafailov et al., 2023; Guo et al., 2024), IPO (Azar et al., 2023), SLiC (Zhao et al., 2023), Raft (Dong et al., 2023), and RRHF (Yuan et al., 2023) are also applicable.

## A.2. Training Data

The dual-system model requires data that incorporate reasoning and thought processes, which are absent in standard LLM training datasets. We propose several data sources.

**Mathematical Reasoning and Puzzle Data.** Synthetic data derived from mathematical problem-solving can effectively simulate logical reasoning and thought processes. Additionally, we can also generate data include strategic board games like Chess, numerical challenges like Sudoku and 24 points game, spatial reasoning tasks such as solving a Rubik's Cube, and logic puzzles like crosswords and Nonograms. Moreover, non-mathematical language logic puzzles are also valuable, enabling the use of removal tokens to remove unsuccessful attempts. Not only are these puzzle data easy to obtain, but they also have the potential to enhance the models' planning abilities. An interesting aspect to explore is whether training on such puzzle-based data can transfer and enhance performance in language reasoning tasks.

**Synthetic Language Data.** We can leverage existing LLMs to generate synthetic datasets for training a dual-system model. The growing body of research on reasoning and planning provides a basis for this. We can create datasets by concatenating different trials and using removal tokens to mark incorrect sequences. Additionally, we can prompt LLMs to describe their reasoning process along with providing answers. These outputs will include both the final answer and intermediate steps, allowing us to construct synthetic datasets that capture the thought process. During this process, we should ensure that the generated data is free of biases and inaccuracies and aligns with ethical guidelines.

**Version Control Data.** Code repositories like GitHub often include commit histories and pull requests, which can showcase the revision process in code development. Collaborative writing platforms such as Wikipedia and Overleaf, with their edit histories and discussion pages, also provide data on how written content is developed and revised. The version history in Dropbox and Google Drive provides iterative process of text creation and modification.

Similarly, websites like Stack Overflow reveal the thought process behind problem-solving or explaining complex concepts. Online forums can also display the evolution of ideas and opinions. Using these datasets requires additional processing to effectively extract the thought process. Furthermore, it is essential to follow ethical guidelines, particularly concerning user privacy and the permissions for using their data. If the datasets contain personal data from users, getting clear permission from them is key for ethical and legal considerations.

### A.3. Fine-tuning

In a training approach that involves initial pre-training followed by fine-tuning, the best practice for pre-training would be to directly use data annotated with the human thought process. However, such data sources are relatively scarce compared to the data used for training state-of-the-art large language models. Therefore, we suggest using these thought process-annotated datasets for fine-tuning. In the future, there is potential to construct high-quality datasets annotated with thought processes that are large enough for pre-training. Achieving this, however, is beyond the scope of individual efforts and requires a collaborative effort within the research community.

### A.4. Mathematical Reasoning

Mathematical reasoning is a crucial part of language model capabilities (Trinh et al., 2024). It is interesting to study whether models can self-improve their mathematical reasoning abilities. For instance, we can use techniques like Self-Rewarding (Yuan et al., 2024) or STaR (Zelikman et al., 2022). A key premise for this method is that the model is able to provide high-quality reward signals. However, current models often suffer from hallucinations in mathematical reasoning, and therefore, the rewards provided by the model may not be reliable. Therefore, we should strictly control the hallucination rate, and then we can use Self-Rewarding that could potentially lead to self-improvement.

We propose that the ratio of incorrect to correct answers can serve as an effective surrogate metric for the model's hallucination rate if the model is capable of generating three types of responses: (a) correct answers, (b) incorrect answers, and (c) acknowledgment of its inability to solve a problem. If this ratio falls below a specific threshold, such as $2\%$, this would indicate a low hallucination rate. This is because frequent hallucinations would make it unlikely for the model to achieve over a $98\%$ correct percentage within the problems it claims to be able to solve, since incorrect intermediate steps are likely to cause an incorrect final answer.

In assigning reward values to answers, it is crucial to consider a diverse range of factors. For instance, we should not only reward the correct steps that lead to correct answers but also reward responses that successfully remove failed trials. However, if a simple problem is solved correctly but with many error trials, it should be penalized. It is also important to reward models for recognizing and admitting their inability to solve certain challenging math problems that are beyond the model's ability, rather than allowing them to arrive at an incorrect answer. This would otherwise indicate a hallucination, which should definitely be prohibited in our framework.

Then, we propose gradually introducing the model to problems of increasing difficulty, while closely maintaining this ratio of incorrect to correct answers under a specific threshold. The most challenging aspect of this method is to ensure a low hallucination rate, particularly in the context of mathematical reasoning. However, we can apply a similar pipeline to simpler tasks where the hallucination rate is more manageable, especially in tasks governed by simple rules yet requiring significant planning abilities. For instance, we could initially train the model in games like the Game of 24, where the objective is to reach 24 using basic arithmetic operations given four numbers, or in constructing crosswords, where the challenge lies in fitting words into a grid based on clues and intersecting letter patterns.

## B. Potential Benefits of the Dual-System

### B.1. Practical Considerations

In this subsection, we argue that the current limitations in LLMs stem largely from the absence of human-like thought process data in the training dataset: while these texts result from deliberate human thinking, they fail to capture the thought process itself that leads to the production of these texts. Consider, for example, the phenomena of hallucinations and arithmetic operations in LLMs.

Current LLMs are pre-trained on large text corpora. This approach aims to expose the model to the vast diversity of human language. However, this exposure can be a double-edged sword, as it may lead to issues such as hallucinations. In these texts, factual information is typically written by humans who know it to be valid, often involving verification of information resources before text production. However, this internal verification mechanism is not captured in the training data itself. Therefore, LLMs might not learn to question the veracity of their information or to express when they "do not know" something. This could lead to overconfidence in their responses and hallucinations.

One potential mitigation strategy is to extract all factual information from training datasets, link it to its sources, and annotate it to indicate the acquisition or internal verification process, followed by the use of removal tokens to remove the verification process. Ideally, when faced with a query involving factual information, the model could then discern whether the query is related to factual information and attempt to verify its source.

A similar principle applies to basic mathematical arithmetic. The model should be trained to discern, as humans do, the complexity of arithmetic operations. For instance, humans can easily recognize that while calculating $3 \times 3$ is straightforward, computing $33333 \times 33333$ is far more complex and typically requires external tools like calculators. Yet, this kind of contextual understanding is absent in the training data for LLMs. For example, existing training data might include:

$33333 \times 33333 = 1111088889.$ `<EOS>`

However, ideal training data would reflect the thought process:

$33333 \times 33333 =?$ `<EOS>` This calculation would be quite challenging. I need to use the calculator. `<EOS>` `<RMV-S>` `<PYTHON>` `<RMV-T>`[1] 1111088889. `<EOS>`

These examples highlight a significant limitation in the text data currently used for training LLMs. The data is typically produced after thorough deliberation, or it represents outcomes derived from using external tools for complex arithmetic tasks. Generating such texts accurately in current LLM practice already proves to be a challenging task. Therefore, we hypothesize that the large size of current LLMs partially results from their design, which focuses on directly outputting the desired answers without generating an internal thought process. Unlike these models, humans can gauge the complexity of reasoning tasks and generate deeper thought processes for more challenging tasks before starting responding. Current LLMs do not have this adaptive mechanism, and to compensate for this, they may *only* depend on increased sizes—more parameters and more data—to *directly* and *statistically* model the final response. This aligns with our current observations that larger models have better reasoning capabilities, and emergent abilities follow a predictable pattern as the model size increases, provided that continuous and linear metrics are used (Schaeffer et al., 2023).

Consequently, our framework aims to address this issue by incorporating this "missing part" of internal thought, enabling models to engage in thorough "thinking" before responding. Moreover, it requires the model to predict the next token with a difficulty similar to fast-thinking, in contrast to the final response required by current LLMs. Since our framework's training objectives are less complex, we expect that models developed within it could achieve reasoning capabilities on par with those of larger, existing LLMs.

### B.2. Necessity for Trial-And-Error

Lightman et al. (2023) showed that reward models were used to improve mathematical reasoning capabilities through either outcome supervision (Uesato et al., 2022), which provides feedback for the final result, or process supervision (Uesato et al., 2022), which provides feedback for each intermediate reasoning step. It was found that process supervision significantly outperformed outcome supervision.

In our framework, we can go one step further than process supervision. Note that when humans solve problems, whether mathematical or puzzles like Sudoku, it involves trying various options, discarding incorrect paths, and iterating toward the correct solution. We argue that this trial-and-error process is a crucial aspect of intelligence development since it is nearly impossible to solve every complex puzzle correctly in one attempt. However, if this process is not explicitly presented in LLM training, it may lead to overconfidence in generated solutions and a tendency for error accumulation (Arora et al., 2022). This limitation may contribute to errors cascading from early mistakes to incorrect final answers. Therefore, our proposed framework is designed to accommodate this trial-and-error process. We can adjust parameters through reference optimization by ranking the candidate answers. In this way, the model can learn from different attempts and understanding their effectiveness, rather than only relying on correct answers. This method mirrors human learning, where understanding and correcting mistakes are integral to developing knowledge and skills.

Prevalent methods guide models towards better planning but don't directly teach them iterative problem-solving. Our framework embeds this learning process within the model itself. For true artificial general intelligence, models must seamlessly plan, generate, and revise outputs as a unified process, not as separate steps. Therefore, our framework has the potential to reduce the dependence on external methods for guiding the models' reasoning steps.

[1] The `<RMV-T>` token is used to remove the question mark and the subsequent `<EOS>` token.

### B.3. Connections with the World Model

The long-standing goal in AI research has been to achieve human-level intelligence. However, the complete operational mechanism of the human brain is still not fully understood. Since we lack the capability to model the brain directly, we turn to language as an accessible and powerful *interface*, focusing on modeling linguistic structures as a reflection of human intelligence. Current successes in AI are largely attributed to the ability of language models to recognize patterns in vast amounts of text data, leveraging word embedding to represent words as vectors in high-dimensional space. This allows words to be processed by neural networks.

However, language is indeed not the only *interface* machines can use. Human intelligence is deeply rooted in the ability to interact with and learn from the environment through various senses (LeCun, 2022). Thus, the idea of a world model (Hao et al., 2023; Xiang et al., 2023; Zhou et al., 2023) in AI is to to form a mental representation of the environment. This allows the model to explore different actions, evaluate potential outcomes, and select the most effective strategies.

However, creating a perfect world model is indeed a complex and currently unattained goal. So, why not start with a simpler one? By embedding actions into vectors (action embedding) and incorporating them into language models, the model can now utilize another *proxy* of human intelligence: actions. Our dual-system model and removal tokens can be seen as an initial step within this framework, where a textual agent is provided with access to external states presented in textual form, allowing it to interact and make decisions based on text-based information.

A study (Hao et al., 2023) suggests that current LLMs do not incorporate a world model. Specifically, these models (a) lack a representation of the world's state, (b) do not possess a reward mechanism to direct the model's actions, and (c) struggle to strike a balance between exploration and exploitation. In our framework, the model itself functions as a prototype of a world model and has the potential to address these limitations. Specifically, (a) the thought system $T$ can be seen as the language model agent, and the output system $S$ represents the observation of the environment. The agent $T$ can interact with the environment $S$ by adding or removing tokens in $S$. (b) The model serves as its own reward model: at each inference step, it has full access to both the environment observation $S$ and the internal state $T$. The model can explore various paths and then revert to the previous state of the environment in $S$ using removal tokens, while keeping the exploration information in $T$. This is the same as planning. The model navigates through all steps and makes decisions based on $S$, $T$, and its own parameters, thereby forming its own reward mechanism. (c) The model balances exploration and exploitation on its own. While Monte Carlo Tree Search (MCTS) is an effective algorithm for balancing exploration and exploitation, it might not be as necessary in our framework. Humans can plan and explore efficiently without understanding mathematical concepts like MCTS and Markov Decision Processes, as these abilities can be acquired non-mathematically. Considering that current LLMs can obtain reasonable reasoning skills when trained only to produce the desired answer, our framework further enables models to be trained to improve the internal thought and planning processes. We therefore hypothesize that a model well-trained in our framework is likely to possess high-level planning abilities.

In conclusion, a model within our framework functions analogously to a world model. Although learning to generate removal tokens seems very hard at this moment, we believe our dual-system approach has the potential to address this challenge. Consider the analogy to human problem-solving: verifying the correctness of a solution and backtracking to a previous state are often much easier than generating the correct solution in the first place. We hypothesize that current LLMs may struggle with this because they lack an environment that supports backtracking and may rely too heavily on correct answers during training. Our dual-system creates a more suitable environment. If we train models in this type of environment, learning to generate removal tokens may become less difficult than currently assumed. Moreover, the model's vast parameter space can collectively contribute to the planning process. Although the success of such a model would depend heavily on its implementation and the effectiveness of its training regime, it certainly opens up exciting possibilities for the future of artificial intelligence.