A Proposal for a Language Model Based Cognitive Architecture

Kobe Knowles, Michael Witbrock, Gillian Dobbie, Vithya Yogarajan

NAOInstitute, Waipapa Taumata Rau - The University of Auckland, New Zealand
kobe.knowles@auckland.ac.nz

Abstract

Large Language Models (LLMs) have shown impressive performance on a wide variety of tasks. However, apparent limitations hinder their performance, especially on tasks that require multiple steps of reasoning or compositionality. Arguably, the primary sources of these limitations are the decoding strategy and how the models are trained. We propose, and provide a general description of, an architecture that combines LLMs and cognitive architectures, called *Language Model based Cognitive Architecture* (LMCA), to overcome these limitations. We draw an analogy between this architecture and "fast" and "slow" thinking in human cognition.

Introduction

Large language models (LLMs) have shown impressive performance on many tasks, such as programming, reasoning, translation, and question answering, often surpassing human performance (Anil et al. 2023; Bubeck et al. 2023; Katz et al. 2023; OpenAI 2023). However, even within LLMs' impressive capabilities, there are evident limitations: LLMs reason semantically, not symbolically (Tang et al. 2023); struggle with pure causal reasoning (Jin et al. 2023); fail to understand simple identifier swaps in Python (Miceli Barone et al. 2023); often perform poorly on tasks that require multi-step reasoning and compositionality (Dziri et al. 2023); and produce confabulations (OpenAI 2023). These weaknesses raise questions about whether artificial general intelligence can be achieved with LLMs and whether they have a deep intuitive understanding.

Two apparent constraints on contemporary LLMs are (1) the next-token prediction decoding strategy and (2) the lack of explicit high-level cognition, which would enable slow thinking (Kahneman 2011). In (1), the decoding strategy does not allow back-tracking from a token once it has been produced, potentially limiting the LLM's ability to solve complex problems that are not trivial to answer by just predicting the next token. There have been attempts to remedy this, including Tree-of-Thought (Long 2023), Self-Refine (Madaan et al. 2023), and Voyager (Wang et al. 2023a), which embed LLMs in a higher-level architecture that allows for iteration over the LLM's output. Related is (2), high-level cognition, which would allow more complex operations than just predicting the next token. Cognitive architectures' working memory systems exemplify this extension (Anderson et al. 2004; Laird, Lebiere, and Rosenbloom 2017; Laird 2019).

Fast and slow thinking are terms associated with the dual-processing theory (Evans and Stanovich 2013; Wason and Evans 1974), commonly described as System 1 and System 2, respectively (Kahneman 2011). These represent different modes of thinking: System 1 is fast, biased, and intuitive; System 2 is slow, effortful, and involves symbolic reasoning. While concerns have been raised about its validity (Conway-Smith and West 2023; Kruglanski 2013), this theory still serves as a useful base in comparing modes of thinking.

Contemporary LLMs are more akin to a version of System 1, whose output could be compared to a thought that pops into a human's mind. While we can prompt LLMs, e.g., with chain-of-thought prompting (Wei et al. 2023), they still do not have the ability to slowly think about a solution like human cognition—although, the importance of this can be argued about; if correct choices are made early enough and with sufficient accuracy, during feed-forward next-token prediction, the performance could in principle be similar. This consideration includes the ability to deliberate on different strategies for solving a problem, concentrate on solving a specific part of a problem, and backtrack from currently generated text. There have been many approaches that attempt to add System 2 abilities to machine learning models (Chen et al. 2019; Fabiano et al. 2023; Lin et al. 2023; Miech et al. 2021). While they implement limited aspects of a higher-level cognition, they lack other critical components that contribute to the multiple modes of thinking in human cognition.

We propose an architecture intended to give abilities associated with human cognition to LLMs (i.e., System 2), specifically by combining previously described AI cognitive architectures (Anderson, Matessa, and Lebiere 1997; Laird, Lebiere, and Rosenbloom 2017; Laird 2019) and LLMs. We call this *Language Model based Cognitive Architecture* (LMCA); subsequent sections provide a brief background about LLMs and cognitive architectures, detail our LMCA, and provide concluding remarks regarding its prospects.

Background

A transformer is an architecture that eschews recurrence by relying only on attention, allowing for increased parallelization and the ability to model long-range dependencies (Vaswani et al. 2017). Transformers consist of many stacked layers, each consisting of a multi-head attention module and a point-wise feed-forward module. Transformers are the building block of current language models (Devlin et al. 2019; OpenAI 2023; Raffel et al. 2019). Language models with parameters in the billions are referred to as large language models (LLMs). Prominent LLMs include GPT-4 (OpenAI 2023), PaLM 2 (Anil et al. 2023), and Llama (Touvron et al. 2023a,b).

While their performance is impressive on many accounts (Bubeck et al. 2023; Katz et al. 2023), recent research puts into question their level of understanding compared against humans' (Jin et al. 2023; Miceli Barone et al. 2023; Tang et al. 2023). Many approaches exist to improve LLMs' capabilities: in-context learning (Chen et al. 2022; Dong et al. 2023; Lampinen et al. 2022; Wei et al. 2023; Wu et al. 2023; Zhang, Feng, and Tan 2022), including chain-of-thought prompting (Shi et al. 2022; Wei et al. 2023; Zhang et al. 2022), and instruction tuning (Jang et al. 2023; Longpre et al. 2023; Ouyang et al. 2022; Peng et al. 2023; Wang et al. 2023c,b; Xu, Shen, and Huang 2023; Ye et al. 2023). While these approaches each have benefits, they have not yet raised the level of understanding to that of humans.

Cognitive architectures support research in modelling the human mind in terms of the underlying mechanisms that produce behaviour (Kotseruba and Tsotsos 2020; Laird, Lebiere, and Rosenbloom 2017); they are software implementations of general theories of intelligence (Laird 2022). Kotseruba and Tsotsos (2020) categorise cognitive architectures based on their representation and type of information processing: symbolic, emergent, and hybrid. Symbolic cognitive architectures (Forbus and Hinrich 2017; Gore et al. 2011) are those that represent concepts with symbols that are manipulated by pre-defined instructions. They perform well at planning and formal reasoning but are not flexible and robust. Emergent cognitive architectures (O'Reilly, Hazy, and Herd 2016; Rohrer 2011) are those associated with the concept of a neuron (biologically plausible or not); information is processed through the propagation of signals. They perform well at learning and adaptability, and allow for enormously parallel processing, but are not transparent and perform more poorly at logical inference than their symbolic counterparts. Hybrid cognitive architectures (Laird 2019; Ritter, Tehranchi, and Oury 2019; Sun 2017) combine elements of both symbolic and emergent architectures; some are more symbolic than emergent and vice versa.

The structure of a cognitive architecture generally consists of short-term and long-term memory, a motor module, and a perception module (Anderson et al. 2004; Laird 2019; Sun 2017). Short-term memory is usually called working memory, and long-term memory is split into declarative and procedural memory. Some procedure retrieves structures from long-term memory and stores them in working memory; alternatively, the perception module can add structures into working memory. Operations can be applied to the structures in working memory to create new structures, modify structures, and remove structures. When appropriate, the motor module produces an action to be executed in the environment, given the current state of working memory. The Common Model of Cognition (Laird, Lebiere, and Rosenbloom 2017) mirrors this characterisation of cognitive architectures.

Language Model Based Cognitive Architecture

We provide a general description of the proposed architecture, LMCA, depicted in Figure 1, to be used in a textual input-output setting. LMCA models the interplay between short-term and long-term memory, with the objective of completing the input task. The types of tasks we consider are those limited to the generation of a singular response, i.e., they are answerable. Long-term memory consists of multiple modules: the Memory Module, Thought Module, and Action Module. Short-term "working" memory, stores structures created by long-term memory modules. A Retrieval Module attends to structures in working memory, which are input to the modules in long-term memory.

Working Memory and Retrieval

Working memory stores relevant structures for solving the current task. A *state* in working memory is a snapshot of the contents of working memory at a given time. We propose five buffers in working memory: the Memory Buffer, Task Buffer, Thought Buffer, Struct Buffer, and Action Buffer. A *structure* in working memory is defined quite generally and differs depending on the working memory buffer; each buffer has its own structure variant, some being more flexible than others. The minimum requirements of a structure are that it has a natural language representation that can be input to other modules (i.e., the Retrieval Module, Memory Module, Thought Module, and Action Module), a unique identifier, and an integer attention value representing whether the current structure is attended to. The implementation of attention in the architecture could correspond to soft attention (a structure can be partially attended to) or hard attention (can either be fully attended to or not at all). There could be an absolute hard cut-off value independent of other structures in working memory and a relative cut-off value dependent on other structures in working memory based on the context length of the modules in long-term memory.

The Memory Buffer, Thought Buffer, and Action Buffer store a history of outputs from the Memory Module, Thought Module, and Action Module from long-term memory, respectively. The structures in these buffers will have a time value (when each structure was created), an attention value, a unique identifier, and a natural language description (the output of their corresponding modules). Structures in the Task Buffer are similar, only that there is an additional *boolean* value indicating if the structure corresponds to the input task and a slot for an answer to the task to be inserted. A sub-task (a task that is not the input task) could be solved in a sub-working memory where the input task is the sub-task.

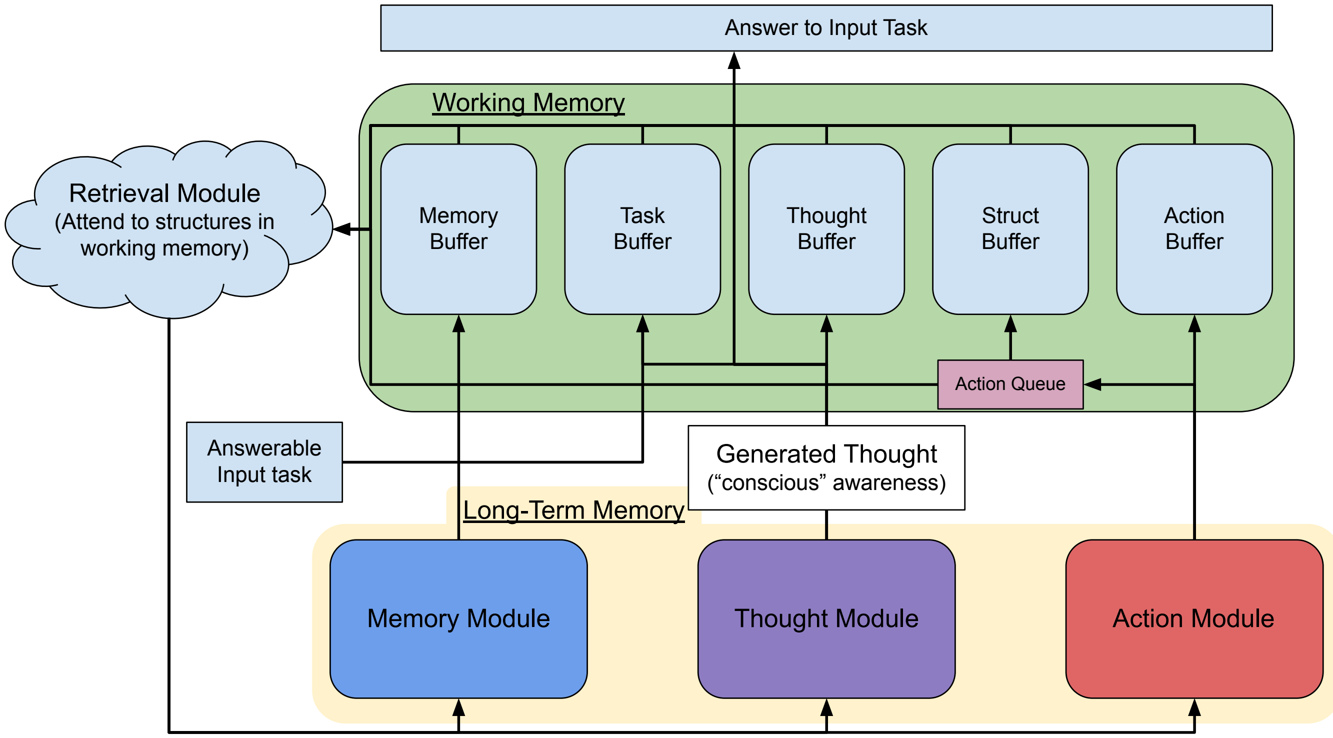

Figure 1: The proposed LMCA in a text-only input-output setting. The architecture consists of three long-term memory modules realised as language models and a retrieval module that attends to structures in working memory; working memory stores structures that are iteratively added to by long-term memory over multiple time steps until an action is produced, prompting the architecture to answer the input task. Each long-term memory module receives as input a textual representation of the attended-to structures in working memory. The Action Module produces actions which are appended to the Action Queue; actions fulfil the purpose of creating and modifying existing structures in working memory, which are stored in five buffers.

The Struct Buffer, named after structs in C, consists of more complex structures. A fluctuating number of variables can be defined in any structure and these variables can have one of several different data types. An *operator* is something that modifies structures in working memory. Allowed operators, such as modifying a variable (i.e., incrementing an integer) or performing operations involving two or more structures, can be defined for structures in the Struct Buffer. Structures are created by the Action Module.

The Retrieval Module has the sole purpose of attending to salient structures in working memory based on their relevance to the current action and is biased by recent thoughts and memories. If the action is to generate memories from the Memory Module, then salient structures in working memory should be attended to. The Retrieval Module could be realised as a language model that has a large enough context length to process all of the working memory at once, producing an attention value for each structure, or instead outputting in order the most salient structures autoregressively by referencing their identifiers.

Memory Module

The Memory Module has the sole purpose of generating relevant memories given the attended-to structures of working memory and is a form of declarative memory. It can be realised as a language model, a database of some sort, or a combination of the two. A *memory* has varying levels of abstraction. At the lowest level of abstraction, we can generate past states of working memory directly. At higher levels of abstraction, we can generate subsets of previous working memory states, based on what was attended to in that working memory state. At the highest level of abstraction, this could be a natural language description of a previous working memory state, summarising the most important elements and their impacts on producing a solution.

Thought Module

The Thought Module's role is to produce thoughts related to metacognition, given the attended-to element of working memory, and is language model based. The thoughts could be related to monitoring and regulation of cognition, a plan on how to solve the current task, the next step in solving the task, generating an answer to the current task, and generating a new intermediary task that will help solve another task. The thoughts are stored in the Thought Buffer in working memory, where they can influence the next action, memory generated, and what is attended to in working memory.

Action Module

The Action Module's purpose is to produce actions that can add structures, modify structures, and remove structures in working memory, and can determine when an output should be generated for the currently attended-to task, given the retrieved structures from working memory. This module is a form of procedural memory. The Action Module will be a language model where the decoding process is constrained to produce valid actions. Multiple actions can be proposed with one call to the Action Module, where they will be added to the Action Buffer in working memory and appended to the Action Queue. The queue will be first-in, first-out; the actions will be executed sequentially until the queue is empty. When the Action Queue is empty, the Action Module will be tasked with producing more actions.

Possible actions include generating a thought based on working memory, generating memories based on working memory, generating structures in the Struct Buffer, generating a new task through generating a thought, modifying structures in the Struct Buffer in working memory via applying operators to structures, and generating an answer to a task. The action-generating process concludes when the input task is answered.

Realising Fast and Slow Thinking

In LMCA, fast thinking occurs when an action is generated initially to complete a task. For example, when the task is to answer "What is 2+2?", the Action Module should generate an action tasking the Thought Module to answer without other actions being applied to working memory. The first thought that the architecture produces is the answer to the task's question, i.e., 4; a fast, intuitive, and automatic thought similar to a thought a human would produce given the same task. Slow thinking occurs when a way to complete the task cannot be found easily by the architecture. Thoughts associated with planning, reasoning, and deliberation are generated. To count as slow thinking, the thought process should be categorized by substantial effort and involve many steps.

Training

The main challenge in realising LMCA is in training it. The Memory Module, Thought Module, Action Module and Retrieval Module are the four components with parameters that need to be trained.[1] Initially, each module should be pre-trained, i.e., each module should have some innate knowledge. The challenge will be acquiring data involving the structures of working memory when solving a task and the associated optimal outputs of each module. This includes the optimal structures to retrieve from a working memory state and the optimal memories, thoughts, and actions given a working memory state.

Instruction tuning (Wang et al. 2023b) is one approach that can be used for training. A high-quality set of examples to train each module should be created. This entails constructing examples of the desired behaviour when solving a problem, i.e., ideal working memory states and the desired outputs of each module. Generating data, both manually and automatically, is a challenge that needs to be overcome. Another option is to pre-train the modules on other representative tasks, ideally in a self-supervised manner. The modules can be trained further on the previously mentioned instruction tuning examples or in a multi-agent reinforcement learning scenario (Nguyen, Nguyen, and Nahavandi 2020; Oroojlooy and Hajinezhad 2023; Zhang, Yang, and Başar 2021).

A key aspect of this architecture is its capacity for continual learning, which depends on updating both long-term memory—including the storage of experiences—and the Retrieval Module as the model acquires experience. One key area to consider supporting continual learning is utilising thought module states (its "thoughts") to identify and reason about errors made. Mechanisms are needed to store memories and update the parameters of the architecture given these thoughts, which includes the need to generate errors internally. Also salient is how to identify and penalise incorrect thoughts, based on reflection and stored memories of previous thoughts.

[1]If the Memory Module is realised as a database with no language model component then there are no parameters to train.

Conclusion

We have proposed an architecture, LMCA, that aims to mitigate the limitations of contemporary LLMs by combining their capabilities with structures drawn from previous work on cognitive architectures in AI. In the context of fast and slow thinking, we aim to give LLMs slow thinking capabilities similar to human cognition, but in a fully trainable setting. In LMCA, LLMs are prominent, being utilised in all four cognitive modules. Avenues to support training the architecture are mentioned, along with considerations of importance and difficulty of generating data and the possibility of generating errors internally dictated by a generated thought. The next, and vital, step will be realising this architecture in a software implementation.

References

Anderson, J. R.; Bothell, D.; Byrne, M. D.; Douglass, S.; Lebiere, C.; and Qin, Y. 2004. An Integrated Theory of the Mind. *Psychological Review*, 111(4): 1036–1060.

Anderson, J. R.; Matessa, M.; and Lebiere, C. 1997. ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention. *Human–Computer Interaction*, 12(4): 439–462.

Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; Chu, E.; Clark, J. H.; El Shafey, L.; Huang, Y.; Meier-Hellstern, K.; Mishra, G.; Moreira, E.; Omernick, M.; Robinson, K.; Ruder, S.; Tay, Y.; Xiao, K.; Xu, Y.; Zhang, Y.; Abrego, G. H.; Ahn, J.; Austin, J.; Barham, P.; Botha, J.; Bradbury, J.; Brahma, S.; Brooks, K.; Catasta, M.; Cheng, Y.; Cherry, C.; Choquette-Choo, C. A.; Chowdhery, A.; Crepy, C.; Dave, S.; Dehghani, M.; Dev, S.; Devlin, J.; Díaz, M.; Du, N.; Dyer, E.; Feinberg, V.; Feng, F.; Fienber, V.; Freitag, M.; Garcia, X.; Gehrmann, S.; Gonzalez, L.; Gur-Ari, G.; Hand, S.; Hashemi, H.; Hou, L.; Howland, J.; Hu, A.; Hui, J.; Hurwitz, J.; Isard, M.; Ittycheriah, A.; Jagielski, M.; Jia, W.; Kenealy, K.; Krikun, M.; Kudugunta, S.; Lan, C.; Lee, K.; Lee, B.; Li, E.; Music Li; Li, W.; Li, Y.; Li, J.; Lim, H.; Lin, H.; Liu, Z.; Liu, F.; Maggioni, M.; Mahendru, A.; Maynez, J.; Misra, V.; Moussalem, M.; Nado, Z.; Nham, J.; Ni, E.; Nystrom, A.; Parrish, A.; Pellat, M.; Polacek, M.; Polozov, A.; Pope, R.; Qiao, S.; Reif, E.; Richter, B.; Riley, P.; Castro Ros, A.; Roy, A.; Saeta, B.; Samuel, R.; Shelby, R.; Slone, A.; Smilkov, D.; So, D. R.; Sohn, D.; Tokumine, S.; Valter, D.; Vasudevan, V.; Vodrahalli, K.; Wang, X.; Wang, P.; Wang, Z.; Wang, T.; Wieting, J.; Wu, Y.; Xu, K.; Xu, Y.; Xue, L.; Yin, P.; Yu, J.; Zhang, Q.; Zheng, S.; Zheng, C.; Zhou, W.; Zhou, D.; Petrov, S.; and Wu, Y. 2023. PaLM 2 Technical Report. *arXiv:2305.10403*.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. *arXiv:2303.12712*.

Chen, D.; Bai, Y.; Zhao, W.; Ament, S.; Gregoire, J. M.; and others. 2019. Deep Reasoning Networks: Thinking Fast and Slow. *arXiv:1906.00855*.

Chen, M.; Du, J.; Pasunuru, R.; Mihaylov, T.; Iyer, S.; Stoyanov, V.; and Kozareva, Z. 2022. Improving In-Context Few-Shot Learning via Self-Supervised Training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3558–3573. Seattle, United States: Association for Computational Linguistics.

Conway-Smith, B.; and West, R. L. 2023. Clarifying System 1 & 2 Through the Common Model of Cognition. *arXiv:2305.10654*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; Li, L.; and Sui, Z. 2023. A Survey on In-Context Learning. *arXiv:2301.00234*.

Dziri, N.; Lu, X.; Sclar, M.; Li, X. L.; Jian, L.; Lin, B. Y.; and others. 2023. Faith and Fate: Limits of Transformers on Compositionality. *arXiv:2305.18654*.

Evans, J. S. B. T.; and Stanovich, K. E. 2013. Dual-Process Theories of Higher Cognition: Advancing the Debate. *Perspectives on Psychological Science: a Journal of the Association for Psychological Science*, 8(3): 223–241.

Fabiano, F.; Pallagani, V.; Ganapini, M. B.; Horesh, L.; Loreggia, A.; Murugesan, K.; Rossi, F.; and Srivastava, B. 2023. Fast and Slow Planning. *arXiv:2303.04283*.

Forbus, K. D.; and Hinrich, T. 2017. Analogy and Relational Representations in the Companion Cognitive Architecture. *AI Magazine*, 38(4): 34–42.

Gore, B. F.; Hooey, B. L.; Wickens, C. D.; Socash, C.; Gosakan, M.; Gacy, M.; Brehon, M.; and Foyle, D. C. 2011. Workload as a Performance Shaping Factor in MIDAS v5. In *20th Behavior Representation in Modeling and Simulation (BRIMS) Conference*.

Jang, J.; Kim, S.; Ye, S.; Kim, D.; Logeswaran, L.; Lee, M.; Lee, K.; and Seo, M. 2023. Exploring the Benefits of Training Expert Language Models over Instruction Tuning. *arXiv:2302.03202*.

Jin, Z.; Liu, J.; Lyu, Z.; Poff, S.; Sachan, M.; Mihalcea, R.; Diab, M.; and Schölkopf, B. 2023. Can Large Language Models Infer Causation from Correlation? *arXiv:2306.05836*.

Kahneman, D. 2011. Thinking, Fast and Slow. *Farrar, Straus and Giroux*.

Katz, D. M.; Bommarito, M. J.; Gao, S.; and Arredondo, P. 2023. GPT-4 Passes the Bar Exam.

Kotseruba, I.; and Tsotsos, J. K. 2020. 40 Years of Cognitive Architectures: Core Cognitive Abilities and Practical Applications. *Artificial Intelligence Review*, 53(1): 17–94.

Kruglanski, A. W. 2013. Only One? The Default Interventionist Perspective as a Unimodel—Commentary on Evans & Stanovich (2013). *Perspectives on Psychological Science*, 8(3): 242–247. PMID: 26172966.

Laird, J. E. 2019. *The Soar Cognitive Architecture*. MIT Press. ISBN 9780262538534.

Laird, J. E. 2022. Introduction to Soar. *arXiv:2205.03854*.

Laird, J. E.; Lebiere, C.; and Rosenbloom, P. S. 2017. A Standard Model of the Mind: Toward a Common Computational Framework across Artificial Intelligence, Cognitive Science, Neuroscience, and Robotics. *AI Magazine*, 38(4): 13–26.

Lampinen, A.; Dasgupta, I.; Chan, S.; Mathewson, K.; Tessler, M.; Creswell, A.; McClelland, J.; Wang, J.; and Hill, F. 2022. Can Language Models Learn from Explanations in Context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 537–563. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Lin, B. Y.; Fu, Y.; Yang, K.; Ammanabrolu, P.; Brahman, F.; Huang, S.; Bhagavatula, C.; Choi, Y.; and Ren, X. 2023. SwiftSage: A Generative Agent with Fast and Slow Thinking for Complex Interactive Tasks. *arXiv:2305.17390*.

Long, J. 2023. Large Language Model Guided Tree-of-Thought. *arXiv:2305.08291*.

Longpre, S.; Hou, L.; Vu, T.; Webson, A.; Chung, H. W.; Tay, Y.; Zhou, D.; Le, Q. V.; Zoph, B.; Wei, J.; and Roberts, A. 2023. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. *arXiv:2301.13688*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Welleck, S.; Majumder, B. P.; Gupta, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv:2303.17651*.

Miceli Barone, A. V.; Barez, F.; Cohen, S. B.; and Konstas, I. 2023. The Larger they are, the Harder they Fail: Language Models do not Recognize Identifier Swaps in Python. In *Findings of the Association for Computational Linguistics: ACL 2023*, 272–292. Toronto, Canada: Association for Computational Linguistics.

Miech, A.; Alayrac, J.-B.; Laptev, I.; Sivic, J.; and Zisserman, A. 2021. Thinking Fast and Slow: Efficient Text-to-Visual Retrieval with Transformers. *arXiv:2103.16553*.

Nguyen, T. T.; Nguyen, N. D.; and Nahavandi, S. 2020. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybernetics*, 50(9): 3826–3839.

OpenAI. 2023. GPT-4 Technical Report. *arXiv:2303.08774*.

O'Reilly, R. C.; Hazy, T. E.; and Herd, S. A. 2016. The Leabra Cognitive Architecture: How to Play 20 Principles with Nature. *The Oxford Handbook of Cognitive Science*, 91: 91–116.

Oroojlooy, A.; and Hajinezhad, D. 2023. A Review of Cooperative Multi-Agent Deep Reinforcement Learning. *Applied Intelligence*, 53(11): 13677–13722.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training Language Models to Follow Instructions with Human Feedback. *arXiv:2203.02155*.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction Tuning with GPT-4. *arXiv:2304.03277*.

Raffel, C.; Shazeer, N. M.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv*, abs/1910.10683.

Ritter, F. E.; Tehranchi, F.; and Oury, J. D. 2019. ACT-R: A Cognitive Architecture for Modeling Cognition. *Wiley Interdisciplinary Reviews. Cognitive Science*, 10(3): e1488.

Rohrer, B. 2011. BECCA: Reintegrating AI for Natural World Interaction. *AAAI Spring Symposium: Designing Intelligent Robots*.

Shi, F.; Suzgun, M.; Freitag, M.; Wang, X.; Srivats, S.; Vosoughi, S.; Chung, H. W.; Tay, Y.; Ruder, S.; Zhou, D.; Das, D.; and Wei, J. 2022. Language Models are Multilingual Chain-of-Thought Reasoners. *arXiv:2210.03057*.

Sun, R. 2017. The CLARION Cognitive Architecture: Toward a Comprehensive Theory of the Mind. In Chipman, S. E. F., ed., *The Oxford Handbook of Cognitive Science*, volume 375, 117–133. New York, NY, US: Oxford University Press, xi.

Tang, X.; Zheng, Z.; Li, J.; Meng, F.; Zhu, S.-C.; Liang, Y.; and Zhang, M. 2023. Large Language Models are In-Context Semantic Reasoners Rather than Symbolic Reasoners. *arXiv:2305.14825*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023a. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *arXiv [cs.CL]*.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv:2305.16291*.

Wang, Y.; Ivison, H.; Dasigi, P.; Hessel, J.; Khot, T.; Chandu, K. R.; Wadden, D.; MacMillan, K.; Smith, N. A.; Beltagy, I.; and Hajishirzi, H. 2023b. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. *arXiv:2306.04751*.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2023c. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 13484–13508. Toronto, Canada: Association for Computational Linguistics.

Wason, P. C.; and Evans, J. S. T. B. T. 1974. Dual Processes in Reasoning? *Cognition*, 3(2): 141–154.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv:2201.11903*.

Wu, Z.; Wang, Y.; Ye, J.; and Kong, L. 2023. Self-Adaptive In-Context Learning: An Information Compression Perspective for In-Context Example Selection and Ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1423–1436. Toronto, Canada: Association for Computational Linguistics.

Xu, Z.; Shen, Y.; and Huang, L. 2023. MultiInstruct: Improving Multi-Modal Zero-Shot Learning via Instruction Tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11445–11465. Toronto, Canada: Association for Computational Linguistics.

Ye, S.; Hwang, H.; Yang, S.; Yun, H.; Kim, Y.; and Seo, M. 2023. In-Context Instruction Learning. *arXiv:2302.14691*.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In Vamvoudakis, K. G.; Wan, Y.; Lewis, F. L.; and Cansever, D., eds., *Handbook of Reinforcement Learning and Control*, 321–384. Cham: Springer International Publishing. ISBN 9783030609900.

Zhang, Y.; Feng, S.; and Tan, C. 2022. Active Example Selection for In-Context Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 9134–9148. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic Chain of Thought Prompting in Large Language Models. *arXiv:2210.03493*.