Augmenting Cognitive Architectures with Large Language Models

Himanshu Joshi[1], Volkan Ustun[2]
[1]Wireless R&D Qualcomm Research, Qualcomm Technologies Inc
[2]University of Southern California, Institute for Creative Technologies
himanshu@qti.qualcomm.com, ustun@ict.usc.edu

Abstract

A particular fusion of generative models and cognitive architectures is discussed with the help of the Soar and Sigma cognitive architectures. After a brief introduction to cognitive architecture concepts and Large Language Models as exemplar generative AI models, one approach towards their fusion is discussed. This is then analyzed with a summary of potential benefits and extensions needed to existing cognitive architectures that is closest to the proposal.

Introduction

Generative AI models such as Large Language Models (LLM), Vision Transformer (ViT) (Dosovitskiy, et al., 2020), etc. models have captured much interest in the recent years. Several proposals have been made towards leveraging strengths of generative AI fused with other models such as LSTMs (Lester, Al-Rfou, & Constant, 2021), cognitive architectures (Wray, Kirk, & Laird, 2021), or planning and reflection with LLMs (Park, et al., 2023) to list a few. The approach of fusing planning and reflection with an LLM yielded good results for the tasks in simulated environment. This has fueled more interest in the question of whether LLMs can be leveraged in cognitive architectures in a principled fashion with the aim of yielding an agent that can utilize the strengths of each approach in different situations yielding an integrated agent that is greater than the sum of its parts. This work proposes one possible fusion between LLMs and cognitive architectures. The attempt here is to propose a creative fusion that connects ideas from both disciplines to yield an approach that yields a result that is potentially better than when either approach – cognitive architectures or LLMs – is used in isolation. The rest of the paper is organized as follows: first cognitive architectures are discussed summarizing key aspects relevant in this context, this is followed by a discussion on generative AI with a focus on LLMs. LLMs are presented in cognitive architecture terminology. Subsequently a fusion of LLMs with cognitive architectures is discussed in the context on a particular kind of task: interactive task learning (Kirk J., 2019). This is then analyzed with a discussion on the potential merits and issues this approach yields. Finally, conclusion and future work are discussed.

Cognitive Architectures

Cognitive architectures model fixed structures and processes underlying human intelligence. Cognitive architectures have a rich history and are a result of Newell's strategy (Newell, 1973) combined with early work on human problem solving. The key premise is that the problem of general human intelligence is difficult because humans act in a variety of diverse environments and that this is made possible because of the ability of the human mind to combine a variety of knowledge sources dynamically on demand (Newell, 1990). The focus then – as a research strategy – is to model the common cognitive architecture that enables the combination of large knowledge sources.

Cognitive architectures can be characterized in terms of the structures – memories – that hold the agent's beliefs, goals, knowledge etc., the representations for these and the processes that act on them (Langley, Laird, & Rogers, 2009). Here, we briefly discuss relevant aspects of cognitive architectures from the perspective of Soar and Sigma like architectures. Very briefly, memories can be either: long term memory, which is further categorized as declarative (knowledge capturing assertions of the agent), procedural (encoding knowledge about how a certain task is performed), episodic (encoding knowledge about history/episodes), and semantic (encoding facts the agent believes), and short-term working memory, which holds knowledge relevant to the agent in its current situation. Knowledge is typically represented in terms of predicates and patterns over predicates. Predicates and predicate patterns can be purely symbolic – as in Soar (Laird, 2012) –

or have sub-symbolic aspects as in Sigma (Rosenbloom, Demski, & Ustun, 2016). In Sigma's case predicates can designate learnable functions. In addition to predicates, there are operators in both Sigma and Soar. Operators represent actions that the agent can take. Central to the processes that act on this knowledge arranged/organized across various memories is the cognitive cycle – eponymously named after the human cognitive cycle. The cognitive cycle represents about 50ms of human mental activity and involves four phases: integrating new perception, elaborating current state, selecting the next action to take – by selection of an operator – to be finally followed by effecting any changes in the working memory (learning) as well as those requiring any output via the motor system. In Sigma operator selection is aided by numerical metadata – in the form of utilities and in Soar operator selection is aided by numeric preferences that can be learnt (as a case of reinforcement learning). The cognitive cycle implements a parallel to serial bottleneck where parallel processing – of multiple rule firings (Soar) or message passing (Sigma) – is followed by deliberate selection of an operator analogous to the human cognitive cycle. Knowledge is organized according to the Problem Space Computation Model (PSCM) in Soar and Sigma. The PSCM is defined as a goal, associated state in the working memory and available operators – all relevant to a particular problem domain. Processing in this setting can be divided into:

- reactive: the 'mindless' aspects of cognition that represent the activity within a cognitive cycle,
- deliberative: the 'mindful' aspects of cognition that represent a series or sequence of decision cycles, and,
- reflective: the reflective or meta-aspects of cognition where the agent examines its own state and makes modification to its internal state.

This tri-level processing model is supported by the decision cycle via the impasse mechanism, where, upon failure to select an operator in the operator selection phase, an impasse results bringing to bear more knowledge by creating a sub-goal to the current goal and a sub-state to the current state.

Learning occurs at multiple levels. Procedural learning entails learning rules that prevent future impasses by creating a knowledge fragment with preconditions that led to the impasse and predicate change – as the action part of the rule – that resolved the impasse. Soar supports chunking but Sigma does not yet support chunking. Sigma's cognitive cycle is based in graphical models – modified factor graphs – and the elaboration phase involves a form of message passing. Learning in this context involves updating the factors with posterior after message passing. Sigma has demonstrated learning of acoustic models, language models, various forms of deep learning such as feedforward multilayer perceptrons, recurrent neural networks (Rosenbloom, Demski, & Ustun, 2017). When the factors in a factor graph (Kschischang, Frey, & Loeliger, 2001) – such as the kind of network Sigma's processing and knowledge are grounded in – are fully differentiable, factor graphs reduce to deep networks and message passing with suitable modifications for regularization can yield learning similar to gradient descent with backprop. Sigma has also shown learning of fixed word embeddings using random projections.

The rest of this work assumes an architecture that is similar in spirit to Soar and Sigma, with a tri-level processing that supports PSCM and a cognitive cycle that is grounded in a form of graphical models that is similar to Sigma and supports chunking like Soar.

Large Language Models

Large Language Models (LLMs) have gained tremendous popularity over the last few years due to their ability to be versatile problem solvers in across several natural language (NL) tasks and even beyond NL domain. LLMs are trained on a very large dataset and require significant investment of resources to train. LLMs are made of layers of stacked transformer models (Vaswani, et al., 2017) which operate on the concept of 'attention' i.e. each word in the input sequence determines how much influence or 'attention' to pay to the other words in the input sequence. Attention involves calculating three intermediate quantities – the query, key, and value vectors – for each word in the input sequence. Each word is 'embedded' in a low dimensional space and then input to the transformer stack. At the top of the transformer stack are classifier 'heads' that generate a distribution over the predicted next word. To begin with, the input vocabulary is transformed into sub-word units called 'tokens' using some method such as byte pair encoding (BPE) (Shibata, et al., 1999). This helps the model handle out of vocabulary words so that any word the model may encounter in the future can be tokenized. These tokens are embedded. Training then involves using gradient descent to predict a (set of) target word(s) – such as the next word(s), or a masked word(s) – in the context of the last several tokens in the input sequence. Training then learns the model parameters along with embeddings for the input tokens.

In the context of previously discussed concepts of different memories in the cognitive architecture, we can think of the LLM transformer stack to be a form of declarative memory clubbed with a procedural 'classifier head.' The lower layers in the transformer stack learn lexical features, the middle layers learn syntactic features and the top layers near the head learn context sensitive semantic features of the target token – in the context of the input tokens – while the classifier head can be understood as predicate rules that act to choose the next word ('choose'

here means generating a distribution over the target token). The predicted tokens can be sampled using a sampling technique and then the sampled tokens can be converted to words using a decoding strategy such as top-k decoder. Once an LLM is trained on a large dataset, there are several ways to use it in a downstream specialized task:

- finetune (Peters, et al., 2018): The LLM is optionally "frozen" and a new classifier head is trained tailored to the new task specific dataset, and,
- prompting (Reynolds & McDonell, 2021): when the LLM is provided with an input prompt, it generates an output sequence of tokens that when converted to words appears very coherent and meaningful. There are several forms of prompting methods, including analogical prompting, templatized prompting etc.

Prompting is very popular because the model is frozen after initial training and not subsequently updated. Prompting templates were initially hand tuned but some work has attempted to search good task specific prompts (Shin, Razeghi, Logan IV, Wallace, & Singh, 2020). Here a more relevant approach is that of soft prompt training (Lester, Al-Rfou, & Constant, 2021), (Liu, et al.) where task specific 'soft tokens' – i.e., tokens that were not in the original token space of the model when it was trained – are inserted in the prompt of the model. The soft tokens are encoded using an LSTM and then inserted with the other tokens in the prompt. Then the soft tokens are learnt in a supervised fashion on a per task basis. Once training results in soft token embeddings, the LSTM is no longer needed, and the soft token embeddings can be used in the same fashion by inserting them in the task prompts using the same template. The advantage of this method is two-fold: the LLM is frozen and does not need to be updated, the task specific tokens can be saved, and new tokens initialized and trained for a new task. This results in two benefits: firstly the amount of training data required is lower, and secondly, the number of parameters trained is far lower than what would be required if the LLM was being fine-tuned with no loss in performance for very large LLM sizes (Lester, Al-Rfou, & Constant, 2021).

LLM Usage in Cognitive Architecture

There are multiple ways in which LLMs can be used in cognitive architectures – as a model of the world, as a reasoning agent that can select actions when prompted with the current state of the agent etc. Here one potential method of LLM integration is proposed. While the LLM itself may not be trained in a cognitively plausible fashion, the integration of the LLM with cognitive architecture is attempted in a cognitively plausible manner.

To begin with, it is assumed here that the LLM itself is not updated as this is prohibitively costly. The core vision here is that an LLM can be used as a prompt-able declarative memory in a Sigma/Soar like impasse driven architecture where the architecture can prompt the LLM with a task specific prompt to extract knowledge from the LLM coupled with task specific operators with learnable continuous embeddings that are inserted in the LLM prompt based on the agent's goals, knowledge of the task, contents of the working memory – that include the current situation – and the current operators that are proposed.

The cognitive cycle supports the ability to learn this continuous representation by using an algorithm similar to Sigma's message passing algorithm. An impasse can be triggered by proposing an operator to impasse which will then create a substate with the subgoal to bring knowledge from LLM to bear on current situation. The task specific operator embeddings in each such substate can be initialized from parent state's corresponding embeddings in conjunction with lexical embeddings that will be used to prompt the model in subsequent step. Prompting the LLM involves inserting these task specific learnable soft tokens as described in previous section. Multiple prompts can be generated based on the goal and various ways in which the soft tokens can be embedded with prompt text.

The prompts themselves can be stored as task specific or general knowledge and several prompt templates can be selected to be used in parallel i.e., in a reactive manner in the elaboration phase of the model. Several relevant templates have been identified in (Wray, Kirk, & Laird, 2021) in the context of Soar's interactive task learning problem formulation. Knowledge obtained from the LLM can be problematic due to several reasons – LLMs hallucinate (McKenna, et al., 2023), and their output is not always reliable (Wray, Kirk, & Laird, 2021). Once multiple responses are retrieved in parallel, one response can be selected by combining knowledge from working memory – that elaborates the current situation – and curated knowledge from curated long-term memories such as episodic memory, semantic memory etc. In Sigma, this can potentially be implemented as a simple classifier that scores the responses.

During the selection phase of the cognitive cycle learning updates the continuous soft token embeddings and when the impasse resolves, the learnt soft tokens embeddings represent a description of the knowledge that was required from the LLM to resolve the impasse as a function of the current state of the agent and its goals for every task operator. Having a labeled dataset which can propagate a signal back from the LLM to the soft token embeddings will help. However, it is important to note the soft token embeddings are learnt not just from the signal from the LLM but from the classifier that scores the responses and potentially accounts for operator utilities derived from task specific knowledge. If the output of the LLM is not usable because the agent does not have actions available – either because it does not know how to perform the suggested

action, or its current state indicates the suggested action is impossible – another impasse can be created to resolve this.

When a new action operator is created by the agent because it does not know how to perform an action suggested by the LLM, the associated soft token embedding with it will be learnt by further querying the LLM to break the complex operator action into a set of simplified actions until an action or set of actions are found that the agent can perform. The operator specific embedding that was learnt for all action operators can be used to determine semantic closeness of actions and the agent can try to substitute such actions. Unlike semantic embeddings for words that indicate semantic similarity in the embedded space, these operator embeddings will be a function not just of the words but the current state of the agent as well. When action operators are available to perform the action, the embeddings can aid – in conjunction with other numerical metadata such as utilities – in planning required to generate a policy over them. When the impasse that led to querying the LLM is resolved, the top state shall have potentially created (action) operators or updates to predicates with associated embeddings and these can be used to update the predicate/operator embeddings in the top state. These embeddings can be subsequently used as task and state specific embeddings and brought to bear either to prevent future impasse in a potentially different situation.

The work closest in approach to the proposed scheme is (Kirk, Lindes, & Peter, 2023) where the authors propose and evaluate a Soar based framework (STARS) to query and use knowledge from an LLM in the ITL task set. STARS stands for Search Tree, Analyze, Repair, and Select corresponding to the phases of the framework where LLM is prompted with hierarchical tree templatized prompts and a beam search is performed to narrow the responses to the most probable set of responses. These are analyzed and evaluated for their usefulness in the current situation and the best one is selected by querying the LLM again. The key difference here is that the whole scheme is working with discrete token prompts derived from discrete words. As discussed previously and based on results from (Lester, Al-Rfou, & Constant, 2021), more task specific data is needed when working in the discrete prompt domain without soft tokens. In this context, this would mean the ST, A, and R phases have more work to do. The hierarchical tree-based templates are simulating structural properties of the task domain as encoded in the English language. The interspersing of soft tokens with state prompt using templates in the scheme proposed in this paper corresponds to this ST phase. Analysis will take place similar to what is proposed in STARS but aided by availability of embeddings. Finally, repair will take place via an impasse mechanism in this proposal. Selection of the next action is left to the cognitive architecture in this proposal. In STARS evaluation, the 'S' selection phase – where the action is selected by the LLM itself – did not improve the agent's task completion performance in the task that was evaluated. In the scheme proposed here, action selection is done by the cognitive architecture with the aid of embeddings, utilities on operators, and the contents of the working memory i.e., the LLM is used to elicit knowledge in a reactive manner only and captured in the task specific operator embeddings which are subsequently used reactively (generator selection score based on utilities on operators as well as the embedding) as well as deliberatively (operator selection). It is unclear how the STARS phases work in the context of Soar's tri-level control. Furthermore, it is unclear whether the beam search involved in the STARS is cognitively plausible and to what extent Soar's cognitive cycle supports handling probabilistic processing.

The idea to use embeddings to aid search is not new, neither is the idea to use LLM's to aid in planning (learning a policy over actions), or reflection (impasse processing). What is new here is the integration of LLM in a cognitive setting with soft tokens on task and state specific operators that can be used to prompt and extract knowledge from the LLM. In the p-tuning work where soft tokens were introduced, they experimented with a few prompt insertion templates. A cognitive architecture can improve upon this manual search of finding suitable prompt insertion template by bringing to bear its mechanisms and knowledge from other memories – such as episodic or semantic – to guide this search potentially improving upon the back and forth required with the LLM i.e., the data required to learn the soft token embeddings.

To evaluate the proposal presented here, the ITL domain seems the most natural. Sigma is the most natural candidate to consider for evaluating. Sigma's decision cycle is both mixed (symbolic+subsymbolic, including capable of neural processing) and hybrid (discrete+continuous) as required by proposal. Sigma's cognitive cycle does not yet support embeddings on operators and this is an extension that will have to be added.

Conclusion and Future Work

A method to augment cognitive architectures with generative LLM memory was proposed. The integration assumes a cognitive cycle that is capable of simultaneously processing symbolic and sub-symbolic information. Various aspects of the integration have been independently demonstrated in Sigma or Soar but some extensions to Sigma will have to be made to support the proposal.

Acknowledgments

The authors would like to thank Taesang Yoo and June Namgoong of Wireless R&D, Qualcomm Research for several fruitful discussions on LLMs and cognitive architectures.

References

Dosovitskiy, A., Beyer, L., Kolsenikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., & Houlsby, N. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint*, arXiv:2010.11929.

Joshi, H., Rosenbloom, P. S., & Ustun, V. 2014. Isolated word recognition in the Sigma cognitive architecture. *Biologically Inspired Cognitive Architectures*(10), 1-9.

Kirk, J., 2019. Learning Hierarchical Compositional Task Definitions through Online Situated Interactive Language Instruction. PhD dissertation, Department of Computer Science, University of Michigan, Ann Arbor, MI.

Kirk, J. R., Lindes, P., & Wray, R. 2023. Improving Knowledge Extraction from LLMs for Robotic Task Learning through Agent Analysis . arXiv preprint, arXiv:2306.06770 [cs.AI]. Ithaca, NY: Cornell University Library.

Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory, 47*, 498-519.

Laird. 2012. *The Soar Cognitive Architecture.* Cambridge, MA: MIT Press.

Langley, P., Laird, J. E., & Rogers, S. 2009. Langley, Pat, John E. Laird, and Seth Rogers. *Cognitive Systems Research, 10*(2), 141-160.

Lester, B., Al-Rfou, R., & Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. arXiv preprint, arXiv:2104.08691 [cs.CL]. Ithaca, NY: Cornell University Library.

McKenna, N., Li, T., Cheng, L., Hosseini, M. J., Johnson, M., & Steedman, M. 2023. Sources of Hallucination by Large Language Models on Inference Tasks. arXiv preprint, arXiv:2305.14552 [cs.CL]. Ithaca, NY: Cornell University Library.

Newell. 1973. You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. (W. G. Chase, Ed.) *Visual Information processing*.

Newell. 1978. Harpy, production systems and human cognition. *Perception and production of fluent speech*, 299-380.

Newell. 1990. *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. arXiv preprint arXiv:2304.03442 [cs.HC]. Ithaca, NY: Cornell University Library.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. 2018. Deep contextualized word representations. *CoRR abs/1802.05365*.

Raschka, S. 2023 Understanding Large Language Models -- A Transformative Reading List. https://sebastianraschka.com/blog/2023/llm-reading-list.html. Accessed: 2023-08-01.

Reynolds, L., & McDonell, K. 2021. rompt programming for large language models: Beyond the few-shot paradigm. . *CHI Conference on Human Factors in Computing Systems Extended Abstracts of the 2021* , pp 1-7.

Rosenbloom. 2012a. Deconstructing reinforcement learning in Sigma. *Proceedings of the 5th Conference on Artificial General Intelligence*, (pp. 262-271).

Rosenbloom. 2013. The Sigma cognitive architecture and system. *AISB Quarterly, 136*, 4-13.

Rosenbloom, P. S., Demski, A., & Ustun, V. 2016. The Sigma cognitive architecture and system: Towards functionally elegant grand unification. *Journal of Artificial General Intelligence*, 7, 1-103.

Rosenbloom, P. S., Demski, A., & Ustun, V. 2016. The Sigma cognitive architecture and system: Towards functionally elegant grand unification. *Journal of Artificial General Intelligence*, 7, 1-103.

Shibata, Y., Kida, T., Fukamachi, S., Takeda, M., Shinohara, A., Shinohara, T., & Arikawa, S., 1999. Byte Pair encoding: A text compression scheme that accelerates pattern matching. https://www.researchgate.net/profile/Takeshi-Shinohara/publication/2310624_Byte_Pair_Encoding_A_Text_Compression_Scheme_That_Accelerates_Pattern_Matching/links/02e7e522f8ea00c318000000/Byte-Pair-Encoding-A-Text-Compression-Scheme-That-Accelerates-Pattern-Matching.pdf. Accessed: 2023-08-01.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., & Singh, S. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint, arXiv:2010.15980 [cs.CL]. Ithaca, NY: Cornell University Library.

Wray, R. E., Kirk, J. R., & Laird, J. E. 2021. Language Models as a Knowledge Source for Cognitive Agents. arXiv preprint, arXiv:2109.08270 [cs.AI]. Ithaca, NY: Cornell University Library.