Automatic Least-Effort Contextual Learning

Lucas E. Morales {lucasem}
Kevin Ellis {ellisk}
Joshua B. Tenenbaum {jbt}

Abstract

Machine learning typically becomes less effective as a subject domain is broadened. Humans, on the other hand, are good at learning in arbitrarily broad domains, and exhibit the ability to specialize in many domains without facing critical consequences from working at such large scales. We hypothesize that this ability emerges, in part, from automatically structured knowledge, where local relations between knowledge artifacts correspond to contextual relevance. We introduce the construction and utilization of this knowledge structure as *contextual learning*, and consider the problem of implementing a contextual learner. We propose a solution to this problem with a framework for automatic learning at various scales of generality inspired by patterns found in social wealth, natural languages, protein interactions, and many other natural phenomena. This framework implements a knowledge abstraction, so many existing machine learning algorithms can be augmented to interface with it allowing knowledge to be automatically structured behind-the-scenes. We experimentally verified our hypothesis by applying our framework to a program induction algorithm tasked with string transformation problems, and by demonstrating the enhanced ability for the program inductor to learn and solve complex problems in a broad domain when augmented to utilize our framework.

1 Introduction

Consider a mathematician attempting to prove a theorem. In order to come up with the proof, the mathematician needs to utilize and build on top of theorems that already exist. However, good mathematicians don't consider using every single theorem that they know — they instead focus their efforts on a handful of theorems they expect to be relevant. This *context* makes searching for a proof tractable. Despite knowing thousands of theorems, a mathematician might focus on only a dozen that seem relevant to the task at hand. Upon successfully proving the theorem, the mathematician will add it to her tool belt, utilizing it whenever a relevant circumstance arises.

This example illustrates *contextual learning* — learning that is conditioned on a relevant subset of knowledge called the context. Not many learning mechanisms exhibit this behavior, but there are some that are capable of being adapted to become contextual. Such learning mechanisms must be *compositional* — they must both consume and construct knowledge, allowing for rapid acquisition and generalization of knowledge when confronted with new complex tasks [1]. To adapt compositional learners to be contextual, these interactions must not be immediately free to use all knowledge: a particular context must motivate relevant knowledge artifacts to be readily available while setting others further apart.

Our approach is motivated by the observation that many compositional learners scale poorly to broad domains. We propose an answer to this challenge of enabling compositional learning mechanisms to become

scalable contextual learners by designing a knowledge abstraction that provides an interface for learning mechanisms while automatically structuring knowledge behind-the-scenes. Our solution is general enough to act as a unified framework on which distinct mechanisms can share knowledge and learn in tandem, so we refer to it as a component of cognitive architecture.

To contextualize knowledge in an abstract manner, we develop a component of cognitive architecture called the SKN framework (**S**cale-Free **K**nowledge **N**etwork). Knowledge is represented as distinct artifacts of information and their relations in the structure of a connected network. The network is constructed by preferential "least effort" attachment [2, 3], where a new knowledge artifact joins the network with relations to the contextual knowledge from which it was learned. This network is scale-free — it conforms to a mathematical pattern similar to that of Zipf's Law and Pareto distributions [4, 5], where the degrees of connectivity for nodes in the network follow a class of power law distributions — a design which inherently encodes a metric of utility in its structure. The resulting system enables automatic contextualization of any learned knowledge, provides effective compositionality for learning mechanisms which rely on knowledge artifacts, and supports multi-mechanism learning with a unified context.

We apply this system to a compositional program inductor tasked with learning string transformations. Our results show that our approach to automatically structuring knowledge allows us to achieve rapid learning of rich models without sacrificing ability to scale to broad domains.

2 Related work

In the discipline of artificial intelligence, much effort is placed on designing accurate learning mechanisms. These learning mechanisms are either designed for a particular domain, or designed to specialize given consistent domain-specific input. Special-purpose learning mechanisms include visual object recognition systems based on the human visual cortex [6], handwritten character identification and generation [7], visual feature modification of images [8], and sound texture perception and synthesis [9]. General-purpose learning mechanisms include hierarchical Bayesian methods [10], program learning [11, 12], inductive logic programming [13, 14], and generative adversarial networks [15].

Each of these mechanisms are remarkable, but lack the generality that is apparent in human cognition. Even the general-purpose mechanisms, while capable of being applied to many domains, are only practical when restricted to a single domain, as a specialized instantiation of the mechanism. The need for specialization makes general-purpose mechanisms unappealing in practice, because a special-purpose mechanism could outperform it. This stems, in part, from a problem of *knowledge*, for which learning mechanisms have some internal representation. Those representations are mechanism-specific, and are generally used without hierarchy. Making all knowledge flat in this manner yields systems that are only performant when confined to a single domain and flawed as a model of cognition due to the reduced capability of both handling very complex problems and specializing in multiple domains.

Theories for cognitive architecture have pursued solving these problems. ACT* [16] is a system which utilizes memory according to a degree of activation, where activation spreads to favor information most related to the immediate context. ACT* relates items of memory with a matrix pairing the strength of connection between any two items, which restricts memory capacity with a flat representation of knowledge and is therefore lacking in the same vein as the mechanisms mentioned above. Soar [17] is another system of cognitive architecture which traverses memory by pattern-matching stored production rules. Soar has

unstructured sets of memory and learns new production rules with implicit generalization about the context it informs. Soar does not have implicit relationships between items of knowledge like ACT* — it has limited explicit relationships about context as a hierarchical data structure of subgoals, problem spaces, states, and operators. Soar relies on generating new production rules efficiently to reduce the complexity of problem solving.

Preferential "least-effort" attachment and resulting power-law distributions have been reported heavily over the last century, from urn models and Yule processes describing the statistics of the evolution of simple mathematical systems, to Zipf's Law describing patterns in social wealth and natural language, to the Matthew effect describing analogous patterns in psychosocial processes, to the emergence of scale-free networks and its presence in protein interactions, linguistics, citations, and many other systems [2, 4, 18–23].

We model non-flat collections of knowledge artifacts using a scale-free network based on intuitive traits of cognition:

- *Growth*, the learning and discovery of new knowledge artifacts expanding a knowledge-base.
- *Least-effort attachment*, the construction of relations between a new knowledge artifact and its antecedents. These relations are not random, but instead associate according to relevance.

Presence of these two traits in systems have been shown effective in yielding scale-free networks. While the flat representations described earlier in this section yield an average graph distance of $\langle d \rangle = 1$ (corresponding to the immediate access to every item in the network and an obvious scale to the network), scale-free networks yield an average distance of $\langle d \rangle \sim \lg \lg N$ for network size N. Even with massive network sizes it's still easy to travel from one point of the network to another. The lack of scale of the network refers to the variety of generally-useful (high degree) and highly-specialized (low degree) items within the same network, a feature which inherently encodes a metric of utility in the network structure.

The goal of SKN is to augment learning mechanisms around *contextual* knowledge in a manner that reduces the intractability of hard problems by implicitly relating knowledge in a non-flat structure, enabling compositionality at arbitrary scale. In terms of the Marr-Poggio levels of analysis [24], SKN serves as a tool on the algorithmic level of abstraction, not as a description of hardware-level structure — analogous to the World Wide Web operating at a higher level over the Internet.

3 Scale-Free Knowledge Network

The SKN framework implements a knowledge abstraction, on par with typical non-hierarchical sets or addressed memory. We refer to a small motivated subnet of the knowledge network as the *context*. We provide a simple interface for interacting with knowledge where all interactions either retrieve information related to the current context or adjust the context within the knowledge network, outlined in Table 1.

A knowledge artifact $\mathcal{I}$ is a structure containing a unique identifier, a symbolic tag denoting which mechanism can understand the item's content, and the arbitrary content itself. Only the $\textsc{Add}(\mathcal{I})$ and $\textsc{Orient}(\mathcal{I})$ methods have the potential to augment the active context. The $\textsc{Add}(\mathcal{I})$ method automatically adds a new item of knowledge to the network, and adjusts the context to include the new item, illustrated in Figure 1.

The context itself is sized according to a pre-determined minimum size $K_{\min}$ and the maximum degree

GET() $\{\mathcal{I}\}$	Yields the set of items in the active context
EXPLORE() $\{\mathcal{I}\}$	Yields the set of items within one edge of the active context
ORIENT($\mathcal{I}$)	Shifts context to focus on the given item
ADD($\mathcal{I}$)	Adds an item to the knowledge network

Table 1: Methods on an instance of the knowledge network.

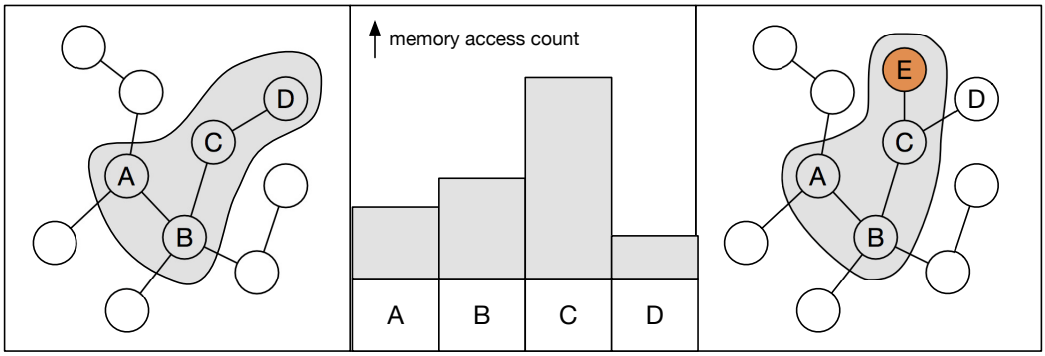

Figure 1: The first pane shows a knowledge network, and a shaded region representing the active context. The second pane shows the memory accesses to particular knowledge artifacts leading up to the addition of a new knowledge artifact. The third pane shows the new knowledge network, with the additional knowledge artifact.

of a node in a scale-free network based on network size N, with degree exponent γ satisfying $2 < \gamma < 3$:

$$k_{\max} \sim N^{\frac{1}{\gamma - 1}}$$

$$K = \max(K_{\min}, k_{\max})$$

We equate the context size K to $k_{\max}$ wherever possible in order to support the scale-free property. When we generate a new context after calls to either ORIENT($\mathcal{I}$) or ADD($\mathcal{I}$), we use a growth procedure as follows: we first uniformly sample γ to probabilistically determine a context size K and initialize the context to the selected node. Then we add the selected node's neighbors to the context in order preferring the higher-degree nodes, select the neighbor with the highest recent access count, and repeat until we reach size K. This process adds relevant nodes while preferring to generalize.

In order to enable the emergence of the scale-free property, the automatic attachment follows a least-effort procedure where the new node's connections are determined using popularity-based probabilistic subset selection. The selection technique we implement is equivalent to a non-initial iteration of the Indian Buffet Process (IBP) with parameter $\alpha = 0$ [25]. For every node $\mathcal{I}_j \in \{\mathcal{I}_i\}_{i=1}^{k}$ in the context, we have m_j, a count of accesses to $\mathcal{I}_j$ since the last context-switch. Additionally, we define $M = \sum_{i=1}^{k} m_i$ as the total count of knowledge artifact accesses since the last context-switch. The new node attaches to node $\mathcal{I}_j$ with probability proportional to its popularity, according to Bernoulli$\left(\frac{m_j}{M}\right)$. If no connections are made for the new node, a single connection is forced using an iteration of the Chinese Restaurant Process (CRP) with parameter $\alpha = 0$ and an initial configuration of k tables associated with nodes in the context where the occupancy of each table is the count of memory accesses to its corresponding node. This yields the same probability of selecting a node as the IBP, but guarantees exactly one connection.

While this attachment procedure follows a least-effort scheme, emergent properties of the network (such as the degree exponent γ) are ultimately determined by how a learning mechanism interacts with knowledge, and no theoretical guarantee of the scale-free property can be made. A guarantee of the scale-free property typically comes from attachments being made to nodes according to their degree of connectivity. This is not anticipated to be problematic for learning mechanisms which interact with knowledge artifacts according to their utility, because the tasks that a mechanism faces should tend to utilize certain concepts more than others, naturally resulting in memory accesses that are proportional to their high utility and thus correlated with high degree of connectivity in the network.

It follows from this specification that the requirements of a learning mechanism in order to utilize SKN effectively are as follows:

- The mechanism must be able to learn a number of knowledge artifacts, which consist of information that is technically independent of other artifacts. Technical independence here refers to the mechanism's capability of utilizing an artifact without needing any other particular artifact.
- The mechanism must have the capacity to consume knowledge artifacts, leveraging them to learn new richer models.
- The mechanism must have a quantifiable notion of *usage* of any particular knowledge artifact in its context.

The first two requirements describe a variant of compositional capability. There are reasons to break away from these requirements without changing the specification of SKN, which are discussed in Section 5.

4 Experiments

To experiment with SKN, consider the class of problems for which this system would excel: complex domains where modular knowledge improves the efficiency of completing relevant tasks. There are many learning mechanisms which are suitable for experimentation, such as inductive logic programming (ILP) systems or algorithms which perform search on a grammar. Distinct knowledge artifacts for ILP systems could be sets of predicates, and for grammar search systems they could be language artifacts which construct a mixture model on which search is performed. In this section, we apply SKN to a problem of learning effective grammars. The grammar learner consumes an initial grammar and solves tasks to construct a more effective grammar, yielding a learning mechanism which fits the requirements shown in the previous section. We augment the grammar learner to utilize SKN and task it with solving string transformation tasks. We demonstrate that SKN improves the effectiveness of the baseline grammar learner.

4.1 Materials

4.1.1 Learning mechanism

We augment an algorithm which performs search on a learned expressive grammar, the EC algorithm [12]. The augmentation defines the grammar based on learned language artifacts stored *in the context*, rather than artifacts stored in a small bounded set constructed dynamically on every task set. We use SKN iteratively, referring to each iteration as a *phase* and the complete ordered set of phases as the *curriculum*. Within each

phase lies a set of tasks used with a number of iterations of the EC algorithm. The augmented algorithm interacts with the knowledge network at the beginning and end of each phase.

The initial grammar employed at the beginning of each phase is defined using the set of combinators in the current context:

$$C := \bigcup \textsc{Get}() = \bigcup_{i=1}^{k} \mathcal{I}_i$$

At the end of each phase, we must determine whether the context should be adjusted and whether to add a new knowledge artifact. We refer to the new set of combinators constructed by the grammar learner as C^*.

Consider the set of combinators beyond the context:

$$C' := \bigcup \textsc{Explore}()$$

If the most probable combinator c_i in C^* is also in C', we adjust the context according to the knowledge artifact which contains c_i:

$$\exists \mathcal{I}.\mathcal{I} \in \textsc{Explore}().c_i \in \mathcal{I}.\textsc{Orient}(\mathcal{I})$$

We now recompute C' with the new state of the knowledge network. We check the P most probable combinators in C^*, referred to as $C^P \subseteq C^*$, for presence in C', and add a new knowledge artifact with every combinator not in C':

$$\left(\mathcal{I} := C^P \setminus C'\right).|\mathcal{I}| > 0.\textsc{Add}(\mathcal{I})$$

The resulting system, parameterized by P, is such that the current context may appropriately be identified as domain-specific knowledge, where a knowledge artifact is a set of combinators which are used as primitives for a grammar defined on the context. Figure 2 illustrates a knowledge network that arises from this system.

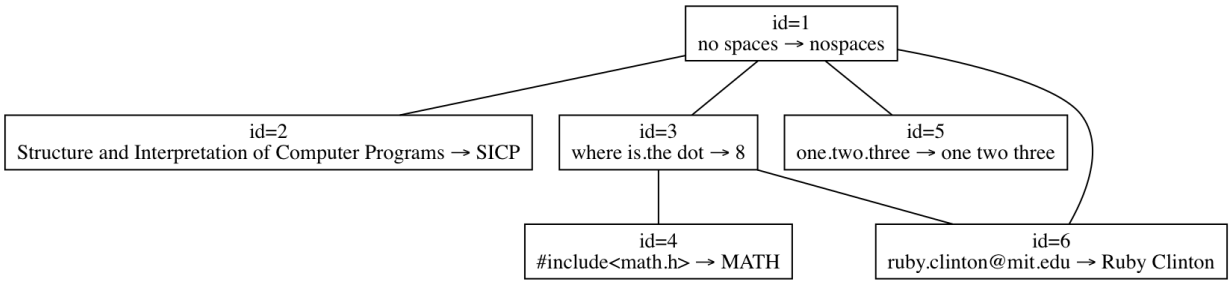

Figure 2: A knowledge network after learning programs with the EC algorithm. For demonstration purposes, each node displays the id corresponding to the phase at which the artifact was learned, and an example of a task from that phase. Actual knowledge artifacts contain learned combinators, not the content shown here. Older artifacts, towards the top, have high connectivity because they contain concepts that are useful.

4.1.2 Task domain

We test the augmented EC algorithm in the domain of string transformation [26, 27], chosen because it has potential for common underlying structure between complex tasks. Microsoft Excel's FlashFill implements program induction for string transformations, but has not been demonstrated to utilize composition over previously learned programs. To handle this task domain, we provide our augmented EC algorithm with the primitives shown in Table 2. Examples of string transformation tasks are shown in Table 3.

<ascii chars>...	char
zero	int
empty	str
string_of_char	char → str
string_of_int	int → str
upper	str → str
lower	str → str
capitalize	str → str
concat_on_spaces	str → str
replace_substr_first	str → str → str
replace_substr_all	str → str → str
incr	int → int
decr	int → int
word_count	str → int
char_count	str → int
find_char	char → str → int
substr	int → int → str → str
replace	str → int → int → str → str
nth	int → str → str
fnth	(str → str) → int → str → str
feach	(str → str) → str → str
is	str → str → bool
filter_words	(str → bool) → str → str

Table 2: Primitives for string transformation given to the EC algorithm, and their associated types.

Task	*Subdomain*	*Input*	*Output*
1	1,3	no spaces	nospaces
2	1	Marin Lorentzen	M L
3	1	Marin Lorentzen	ML
4	1	Words that start With Caps	Words With Caps
5	1	Structure and Interpretation of Computer Programs	SICP
6	2,3	ruby	r
7	2,3	where is.the dot	8
8	2	where is<the angle	8
9	2,3	discard after.dot	discard after
10	2	drop first two chars	op first two chars
11	2	before angle<discard	discard
12	2	discard>after angle	discard
13	2	#include <os.h>	os.h
14	2	#include <os.h>	os
15	2	#include <os.h>	OS
16	3	discard@after at	discard
17	3	ruby.clinton	ruby clinton
18	3	ruby clinton	Ruby Clinton
19	3	ruby.clinton	Ruby Clinton
20	3	ruby.clinton@mit.edu	Ruby Clinton

Table 3: Complete list of string transformation tasks that we teach the EC algorithm to solve compositionally. Not all subdomains have more than one phasic task set. Some tasks are repeated to enable non-contextual learning to successfully complete all phasic task sets.

The string transformation tasks were first separated into three distinct subdomains, each of which contained tasks of varying complexity that respectively culminated in tasks 5, 15, and 20 shown in Table 3. We then separated the tasks within each subdomain into smaller sets of tasks which would be supplied to the augmented EC algorithm in a single phase. We refer to these as *phasic task sets*. The structure of these phasic task sets is much like a school curriculum — they start simple and get more complex as foundational

Task	*Solution*
1	`replace-substr-all empty (string-of-char <space>)`
2	`feach (substr zero (incr zero))`
3	`λa.replace-substr-all empty (string-of-char <space>)` `(feach (substr zero (incr zero)) a)`
4	`filter-words (λa.is a (cap a))`

5	`λa.replace-substr-all empty (string-of-char <space>)`
	`(feach (substr zero (incr zero))`
	`(filter-words (λb.is b (cap b)) a))`

Table 4: Solutions for a particular phasic task set made by both the specialized and contextual grammars, displayed in lambda calculus.

concepts are learned. See Table 4 for an example of a phasic task set and solutions to its tasks.

4.2 Method

We compare primitive, specialized, and contextual grammars in terms of total clock-time performance, EC-iteration clock-time performance, and solution likelihood. Each task has multiple examples, where between one and two examples are used for training while any remaining examples are used for cross-validation. If cross-validation dictates that a solution is a false positive, or if no solution was successfully enumerated, the task is marked as a failure.

For the primitive grammar results, we perform one iteration of the EC algorithm on each phasic task set. This demonstrates the strength of our set of primitives with minimal composition. We perform two analyses with specialized grammars: *per-phase* and *full-domain*. For the per-phase specialized grammar, we independently perform many iterations of the EC algorithm on each phasic task set. This will tailor the learned grammar to the tasks at hand, while not relying on learned grammars from any other phasic task set. For the full-domain specialized grammar, we pool together every task into one massive task set, and run many iterations of the EC algorithm on the that task set. This will demonstrate that the task domain is broad and yields ineffective learning when attempted at full scale. For the contextual grammar, we first learn according to the curriculum, then reiterate the curriculum and use results from the reiteration. We reiterate because we are interested in the contextual grammar's results after it has already learned the material and automatically formed a SKN. In other words, we are interested in the *recall* of contextual knowledge. This will yield a large — but not comprehensive — grammar corresponding to whatever artifacts are in the SKN's context during the particular reiterated phase.

The total clock-time measurement is simply the time spent on each phase. The EC-iteration clock-time measurement is the time it took the final iteration of the EC algorithm to enumerate particular solutions for each task, which represents the expressiveness of the grammar that the EC algorithm ultimately utilized. The likelihood measurement effectively represents the probability of an expression given the grammar in the final iteration of the EC algorithm [12]. All results use the same parameters for the EC algorithm except for the number of iterations: The primitive grammar uses only one iteration, the specialized grammars use five iterations, and the contextual grammar uses five iterations when initially learning from the curriculum and only one iteration during recall when we take measurements.

4.3 Results and discussion

To demonstrate that SKN provides automatic compositionality, we anticipate the contextual grammar to out-perform all other grammars in terms of total speed and to perform at least as well as all other grammars in the solve speeds measured during the final iteration of the EC algorithm, and we anticipate the solution likelihoods with the contextual grammar to be about the same as with per-phase specialized grammar. The faster total speed is expected because the contextual grammar is running only one iteration of the EC algorithm — we can get away with this because the contextual grammar should start each phase with an expressive grammar already prepared, whereas the specialized grammars must spend more iterations to learn such a grammar. The at-least-as-fast speeds in the final iteration are expected because the contextual grammar would have, during a previous phase, already performed the EC algorithm on the same tasks for the same number of iterations as the per-phase specialized grammar. This does not imply that such measurements are meaningless, because the contextual grammar must rely on an expressive contexts that do not contain too many irrelevant combinators which would increase the enumeration time for EC. The similar solution likelihood is expected because the contextual grammar should find the same solutions as the per-phase specialized grammar. Additionally, we anticipate the full-domain specialized grammar to perform poorly in all measurements because the domain is sufficiently broad to deem a single unified grammar poorly expressive for many tasks. Our results are consistent with these expectations.

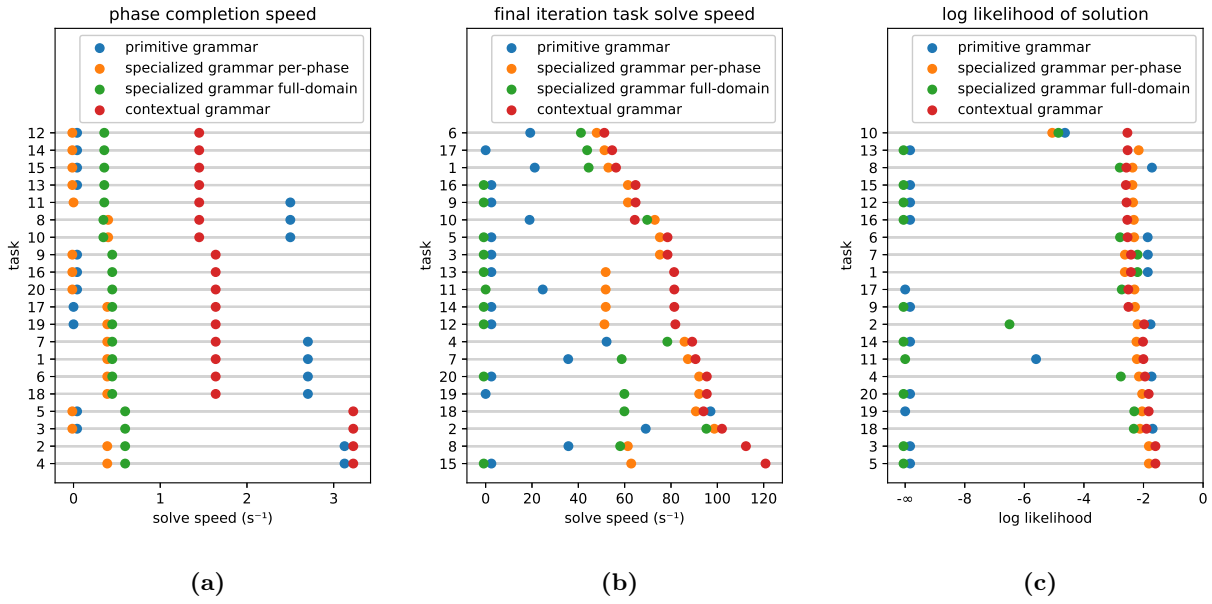

Figure 3: Tasks are automatically ordered based on performance in each metric. (a) The speed at which phases were completed, shown for each task. (b) The speed at which tasks were solved in the final iteration of the EC algorithm. (c) The log likelihood of solutions for tasks conditioned on the grammar.

The comparison of phase completion speeds is shown in Figure 3a. Because the primitive and contextual grammars used only one iteration of the EC algorithm, they are much faster than either of the specialized grammars.

The comparison of solution speeds in the final iteration of the EC algorithm is shown in Figure 3b. The performance of the contextual grammar is almost always superior to every other grammar. The shortcomings of the primitive grammar are due to its lack of compositional learning. Though it can solve every task we provided, the per-phase specialized grammar occasionally lacks the performance of the contextual grammar because it does not also utilize learned grammars from previous phases that are contextually relevant. The full-domain specialized grammar is unable to solve every task due to the large number of tasks it faced, especially given the constraints on the search depth and the number of iterations of the EC algorithm.

The comparison of solution likelihoods is shown in Figure 3c. The per-phase specialized and contextual grammars perform very similarly due to the similarities in their respective learned combinators. The full-domain specialized grammar generally performed worse than either the per-phase specialized or contextual grammars, though it is superior to the primitive grammar with few exceptions.

5 Conclusion and future work

We introduce the problem of contextual learning, and observe the costly oversight of the lack of attention to knowledge structure in many existing learning mechanisms. We propose a solution that enables compositional learners to become scalable contextual learners. Our solution features a novel method of automatically structuring knowledge inspired by patterns found in natural processes, and is designed to interface well with existing learning mechanisms. This makes it an ideal candidate for utilization with any mechanisms that satisfy the requirements described in Section 3, whether to help scale a mechanism to broad domains or to use as a unified learning framework for many mechanisms. We applied SKN to one such mechanism, the EC algorithm, and demonstrated that SKN is effective at enhancing learning in broader domains.

The SKN system as experimented on in this paper is missing an important element of *context localization*. We believe a major step in improving SKN's effectiveness is to introduce an artificial neural network which maps perceptual input to items in the context. An online machine learning algorithm such as the multiclass perceptron could iteratively expand its memory as new knowledge artifacts are learned by other mechanisms, and make calls to $\textsc{Orient}(\mathcal{I})$ where deemed effective upon beginning a new phase.

The SKN system can be utilized in many different ways. By connecting many learning mechanisms to the same knowledge network, interoperability between distinct learning mechanisms such as for vision and sound can result in proximal — and perhaps even shared — knowledge artifacts representing the same *multi-modal* symbol being perceived by different mechanisms. Because learning mechanisms may treat knowledge artifacts as first-class objects that can be observed or created, the mechanisms do not necessarily have to be restricted to *learners* — the introduction of *actors* as mechanisms would yield a production system conditioned on contextual knowledge and immediate perceptional information. Supplemented with support for direct communication between mechanisms, collections of mechanisms can be structured and would be more fittingly referred to as *agents* [28]. The SKN system, in conjunction with these agents, should be tested as a model of cognitive architecture.

References

[1] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2016). Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*.

[2] Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439), 509-512.

[3] i Cancho, R. F., & SolÃĺ, R. V. (2003) Least effort and the origins of scaling in human language. In *Proceedings of the National Academy of Sciences* 100(3), 788-791.

[4] Zipf, G. K. (1949). Human Behaviour and the Principle of Least-Effort.

[5] Barabási, A. L. (2016). *Network science*. Cambridge University Press.

[6] Serre, T., Oliva, A., & Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. In *Proceedings of the National Academy of Sciences*, 104(15), 6424-6429.

[7] Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 1332-1338.

[8] Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems* (pp. 2539-2547).

[9] McDermott, J. H., & Simoncelli, E. P. (2011). Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis. *Neuron*, 71(5), 926-940.

[10] Tenenbaum, J. B., & Griffiths, T. L. (2001). The rational basis of representativeness. In *Proceedings of the 23rd annual conference of the Cognitive Science Society* (p. 103641).

[11] Liang, P., Jordan, M. I., & Klein, D. (2010). Learning programs: a hierarchical Bayesian approach. *International Conference on Machine Learning*: 639âĂŞ646

[12] Dechter, E., Malmaud, J., Adams, R. P., & Tenenbaum, J. B. (2013). Bootstrap Learning via Modular Concept Discovery. In *IJCAI*.

[13] Lavrac, N., & Dzeroski, S. (1994). Inductive Logic Programming. In *WLP* (pp. 146-160).

[14] Muggleton, S. H., Lin, D., & Tamaddoni-Nezhad, A. (2015). Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning, 100*(1), 49-73.

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672-2680).

[16] Anderson, J. R. (1983). *The architecture of cognition*. Psychology Press.

[17] Newell, A. (1994). *Unified theories of cognition*. Harvard University Press.

[18] Udny Yule, G. (1925). A mathematical theory of evolution, based on the conclusions of Dr. JC Willis, FRS. *Philosophical Transactions of the Royal Society of London Series B*, 213, 21-87.

[19] Merton, R. K. (1968). The Matthew effect in science. *Science, 159*(3810), 56-63.

[20] Jeong, H., Mason, S. P., BarabÃąsi, A. L., & Oltvai, Z. N. (2001). Lethality and centrality in protein networks. *Nature, 411*(6833), 41-42.

[21] i Cancho, R. F., & SolÃĺ, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London B: Biological Sciences, 268*(1482), 2261-2265.

[22] de Solla Price, D. J. (2002). The pattern of bibliographic references indicates the nature of the scientific research front. *Social Networks: Critical Concepts in Sociology, 4*, 328.

[23] Clauset A., Shalizi, C. R., & Newman M. E. J. (2009) Power-law distributions in empirical data. *SIAM Review* 51(4), 661-703 (arXiv:0706.1062, doi:10.1137/070710111)

[24] Marr, D., & Poggio, T. (1976). From understanding computation to understanding neural circuitry.

[25] Griffiths, T. L., & Ghahramani, Z. (2011). The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr), 1185-1224.

[26] Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices* (Vol. 46, No. 1, pp. 317-330). ACM.

[27] Lin, D., Dechter, E., Ellis, K., Tenenbaum, J., & Muggleton, S. (2014). Bias reformulation for one-shot function induction. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence* (pp. 525-530). IOS Press.

[28] Minsky, M. (1988). *Society of mind*. Simon and Schuster.