Spontaneous Retrieval from Long-Term Memory for a Cognitive Architecture

Justin Li and **John Laird**
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109-2121
{justinnh, laird}@umich.edu

Abstract

This paper presents the first functional evaluation of spontaneous, uncued retrieval from long-term memory in a cognitive architecture. The key insight is that current deliberate cued retrieval mechanisms require the agent to have knowledge of when and what to retrieve — knowledge that may be missing or incorrect. Spontaneous uncued retrieval eliminates these requirements through automatic retrievals that use the agent's problem solving context as a heuristic for relevance, thus supplementing deliberate cued retrieval. Using constraints derived from this insight, we sketch the space of spontaneous retrieval mechanisms and describe an implementation of spontaneous retrieval in Soar together with an agent that takes advantage of that mechanism. Empirical evidence is provided in the Missing Link word-puzzle domain, where agents using spontaneous retrieval out-perform agents without that capability, leading us to conclude that spontaneous retrieval can be a useful mechanism and is worth further exploration.

Introduction

In almost every cognitive architecture, there is a distinction between short-term (working) memory and long-term declarative memory, and the mechanisms that transfer knowledge between the two have been a topic of study since cognitive architectures existed (Raaijmakers and Shiffrin 1981). Although many long-term memory mechanisms have been explored, one remains curiously unexamined: that of a mechanism which automatically (*spontaneously*) retrieves knowledge from long-term memory to short-term memory.

This is surprising, as human involuntary memory has been recognized since the first systematic studies of memory (Ebbinghaus 1913), and has continued to receive attention up to the present day (Kvavilashvili and Mandler 2004; Hintzman 2011). Its utility for artificial agents, however, has not been evaluated. This is likely due to the success of deliberate retrievals, and agent designers' preference for mechanisms whose behavior they can specify precisely. The inherent unpredictability of spontaneous retrievals, both in when the retrieval occurs and in what memory is returned, may make it difficult to design agents that can effectively use such a mechanism without compromising task performance. Furthermore, it is difficult to imagine the circumstances under which spontaneous retrieval would out-perform an agent with top-down control of its memory.

This paper takes the first step in exploring how spontaneous retrieval can be effectively used in artificial agents. The key insight is that deliberate retrieval has certain knowledge requirements — namely the knowledge of when to search memory and what to search for — that spontaneous retrieval does not. We therefore hypothesize that spontaneous retrieval could be well-suited for agents in domains where this knowledge is unavailable or otherwise incorrect. This insight is crucial in defining a space of possible spontaneous retrieval mechanisms, as well as in deriving the design principles necessary for an agent to take advantage of these retrievals. Results from the Missing Link word puzzle support our hypothesis, and further hint at the conditions under which spontaneous retrieval can be beneficial.

Background: Memory Mechanisms in Cognitive Architectures

The goal of cognitive architecture research is to explore and understand the computational mechanisms that support general intelligence. Although there are a number of cognitive architectures in use, the majority share a similar core design; here the focus is on ACT-R (Anderson 2007) and Soar (Laird 2012), two widely used architectures.

All knowledge is represented as an edge-labeled directed graph in ACT-R and Soar. *Working memory* holds the knowledge that is immediately relevant to the agent's current situation, while *long-term memory* contains other general knowledge about the world; a single piece of knowledge in either memory is called a *memory element*. Working memory has specialized *buffers* that contain information about the agent's perception and motor output; an input-processing-output step is called a *decision cycle*. These buffers may be changed by *procedural rules*, which match on and modify working memory.

In addition to input and output buffers, there are also buffers for long-term memory access, where rules can initiate *retrievals*. Such retrievals require that the agent create *cues* that describe the desired features; long-term memory then finds all memory elements that match the cue, and returns a single element (a graph vertex together with all of its outgoing edges) as determined by some *bias*. In ACT-R and

Soar, this bias is called *activation*, and is the sum of several factors. One factor is an element's *base-level activation*, which decreases (*decays*) over time but increases (*is boosted*) with every access (every retrieval or storage of the element). Another factor is activation that has *spread* from neighboring elements, allowing the bias to take the current context of the agent into account. The higher the total activation, the more likely an element is selected for retrieval (assuming it matches the cue); the element is recreated in the long-term memory buffer, where it can be matched on and modified by rules. Similarly, the agent can also *store* a working memory element into long-term memory through this buffer.

Long-term memory retrieval must be *deliberately* initiated: the retrieval buffer never changes without some rule firing. In contrast, a retrieval that is spontaneous would change the buffer without action on the part of the agent; this is known as *buffer stuffing* by the ACT-R community. One such mechanism has been implemented in ACT-R's declarative memory, but has never been used. This is partially due to the lack of documentation in the manual, and partially because it is easier to create cognitive models that have top-down control of memory retrievals (Lebiere 2014).

Outside of cognitive architecture research, the case-based reasoning (CBR) community has also considered the problem of when to search for relevant cases (Riesbeck and Schank 1989). The focus of that work, however, remains on a useful definition of relevance, while ignoring when relevant cases should be searched for; in fact, most CBR agents begin by being given a case, for which spontaneity has no role (Aamodt and Plaza 1994). Our work has similarities with work on spontaneous analogies (Pickett and Aha 2013), although "spontaneity" in that work is only vaguely defined and focuses more on perceptual features. As a result, the work reported here is the first application of a spontaneous retrieval mechanism that we know of.

Theoretical Benefits of Spontaneous Retrieval

To design a spontaneous retrieval mechanism, it is necessary to first consider the role it plays in an intelligent agent. The goal of all long-term memory retrievals, deliberate or spontaneous, is to find knowledge that is relevant to the agent's current task; in other words, it is a mechanism for *knowledge search* (Newell 1990). This search is necessary because knowledge in long-term memory may not always be well organized and immediately accessible. This is especially true for a long-lived agent that acquires knowledge during its lifetime and must use that knowledge for multiple goals. In this context, the cues created by the agent in a deliberate retrieval provide heuristic guidance for what knowledge might be relevant. Since the search is deliberate, however, the agent's rules dictate not only what to search for, but also *when* to perform this search. This raises the question of how the agent copes when it lacks this search-guidance knowledge, or when the heuristics are incorrect. It would be beneficial to have a mechanism that provides a more general heuristic that is robust to the agent's lack of search knowledge, even if it is not as accurate as is possible with a deliberate cue.

Consider an agent in the real world, where it perceives an abundance of objects every perceptual cycle. The agent may have additional knowledge about many of these objects, but for the majority of objects, this knowledge does not immediately benefit the agent. Moreover, the agent may not know which objects are important, and therefore cannot focus its search on these objects, if any of the objects are important at all. Without this search-guidance knowledge and with only deliberate retrieval, the agent could only retrieve additional information by brute force. Spontaneous retrieval could help in this situation by using subsymbolic information in the agent to heuristically provide it with knowledge.

This analysis has several implications for the design of the mechanism. These implications are further discussed in the next section, but the most important one is that the mechanism cannot depend on the agent's procedural knowledge to initiate the search nor to provide a cue to guide search, since it must be robust to the agent's lack of knowledge of both when to search and what to search for. The mechanism must therefore be automatic or *spontaneous* and *uncued* — hence the focus of this paper on spontaneous retrieval. In particular, we claim that spontaneous retrieval can, under the right circumstances, allow an agent to overcome its lack of knowledge of:

C1 *when* to search long-term memory

C2 what to search for in long-term memory, if the *cue* is unknown

C3 what to search for in long-term memory, if the *relation* of the cue to the desired knowledge is unknown

The Space of Spontaneous Retrieval Mechanisms

The problem of missing or incorrect heuristics for knowledge search presents additional constraints on spontaneous retrieval beyond the need for it to be automatic and uncued. Within these constraints and guided by the theoretical benefits, there is a space for variations in implementation.

The first consideration is the conditions under which a spontaneous retrieval should occur, which must take into account its interaction with deliberate retrieval. The simplest choice is to provide an additional buffer into which spontaneously retrieved elements can be placed, so that deliberate retrievals are not overwritten. This requires the agent to decide which of the two retrievals are more important, or otherwise integrate these two sources of information. On the other hand, having a single buffer for both retrievals better reflects how both mechanisms serve the same purpose in knowledge search, allowing the agent to be agnostic as to the source of the retrieval. In such an approach, spontaneous retrievals occur only when no other retrievals are deliberately initiated by the agent.

A different consideration is that the retrieved element should be relevant to the agent's current situation. This is a requirement shared with deliberate retrieval, and if activation is used as a proxy for relevance, the solution of spreading activation also applies to spontaneous retrieval. There is, however, no standard algorithm for spreading activation; in particular, the *source* of the spread may be different. In ACT-R, spreading activation originates from elements in

working memory, increasing the likelihood that its immediate neighbors will be retrieved. Note that the activation due to spreading is separate from the activation due to base-level, so the resulting retrieval bias may lack the historical context of previous retrievals. A different spreading mechanism could directly affect base-level, by treating the event as another retrieval or storage. More nuanced spreading mechanisms are also possible, where there is a limit to the distance to which activation spreads, or where the change in retrieval bias decreases as the affected element is further away from the source. Finally, if the knowledge graph includes cycles, there may be differences in whether spreading affects the same element multiple times; in this case, mechanisms like inhibition may be necessary to prevent runaway positive feedback loops (Lebiere and Best 2009).

Although non-activation-based mechanisms are also possible — one can cast spontaneous retrieval as learning a mapping from agent state to a single long-term memory element — they are not well studied in literature.

A related but separate design decision is the selection of the element to be retrieved. Several possibilities exist, drawing inspiration from action selection in reinforcement learning: to always select the most highly activated element; to be epsilon-greedy, where the most highly activated element is selected except for some small probability of a random selection; or to use a softmax function where the probability of an element being retrieved is proportional to its activation value. Again, these options also exist for deliberate cued retrieval, except that with spontaneous retrieval, there is no hard constraint for the element to match a cue; in fact, uncued retrievals are a natural extension of partial matching mechanisms that are available in ACT-R and in Soar's episodic memory.

Implementation In Soar

For our implementation of spontaneous retrieval in Soar, we chose to maximize the similarities between deliberate and spontaneous retrieval, and to highlight the unique aspects of spontaneous retrieval. Spontaneous retrieval therefore selects the most highly activated element and uses the same buffer as deliberate retrieval, and only occurs when no deliberate retrievals are in progress. Spreading activation occurs whenever an element is stored or retrieved, and is treated as another access to the affected elements. Specifically, both the parents and the children of the source node, and the parents and children of those neighbors, and so on, are all affected, up to some parameterized distance d.

There are three constraints on spontaneous retrieval to prevent positive feedback loops. First, activation is not allowed to spread in a cycle, so each element could only be activated once per spread. Second, spontaneous retrieval can only return elements that are not already in working memory, with the exception that, if the element that was previously spontaneously retrieved remains the most highly activated one, it is kept in working memory and no other element replaces it. Finally, spontaneously retrieved elements do not receive a boost in base-level activation; this prevents the element from continually having the highest activation.

In terms of computational efficiency, calculating the effects of spreading activation is expensive, and has been an ongoing research problem for the cognitive architecture community (Douglass and Myers 2010; Chen, Petrovic, and Clark 2014). The goal of this research is to investigate the functionality that spontaneous retrieval affords, so our implementation of spreading activation is straightforward and unoptimized. The functionality of our implementation of spontaneous retrieval does not depend on the underlying implementation details of spreading activation. Specifically, Soar's long-term memory is implemented as a SQLite database, which uses indices to allow efficient queries; in particular, all long-term memory elements are indexed by their activation value for fast deliberate retrieval (Derbinsky, Laird, and Smith 2010). For spontaneous retrieval, an element is selected by iterating through the index until an element is found that is not already in working memory. This allows spontaneous retrieval to take advantage of any efficiency improvements for spreading activation, and data are presented below to show the costs of these two processes separately.

Evaluation in the Missing Link Domain

In order to evaluate the benefits of spontaneous retrieval, we use the Missing Link word puzzle as a domain. In this puzzle, the agent is given three words (*stems*) as the clue (for example, `fall`, `fort`, and `time`), and must provide a fourth word (the *link*; in this case, `night`) that forms compound words or phrases with all three stems (`nightfall`, `fortnight`, `nighttime`). The puzzles used in this evaluation are gathered from the Unix word list, using compound words that can be completely divided into two shorter words; a total of 550 compound words formed by 195 stems are used, with each stem being used in an average of 2.8 compound words. In the single-link representation (explained below), the compound words and stems form a connected bipartite graph of diameter 12.

It should be noted that this puzzle is the same one used in the Remote Associates Test (Mednick 1962), and it was chosen for some of the same reasons. In addition to the large amount of search necessary to find the solution, the puzzle also allows the agent's knowledge and the important features of the environment to be transparently manipulated. Search guidance in this domain depends on the connections between the stems and their compound words, which in turn depends on the source of knowledge. A generic dictionary may provide only the component stems of a compound word, while a knowledge base optimized for the Missing Link puzzle would have connections directly between the stems. While the existence of the latter connections may make the puzzle trivial, spontaneous retrieval may help with less specialized representations. To model this variation in knowledge, three different representations are tested in the basic domain (they are also depicted in Figure 1):

- The *single*-link representation only has links from the compound words to its prefixes and suffixes.
- The *double*-link representation also has links from the stems to all their compound words.
- The *direct*-link representation also has links between the prefixes and suffixes of any compound word.

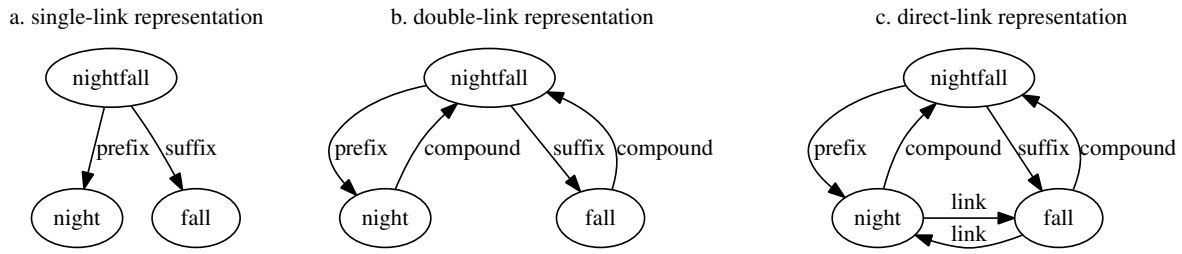

Figure 1: Different representations of knowledge in the Missing Link domain. The nodes are labeled for convenience only; in reality, the string that represents the word (eg. `night`) is a child of the node. For simplicity, other compound words that also use these stems are not shown.

Two additional variations of the Missing Link domain are used. In the first variation, Subset Missing Link, the agent is additionally presented with *distractors*, other words that do not form compound words with the missing link. In the above example, the additional words `foot` and `man` might also be presented to the agent. The addition of distractors models how only a subset of features in the environment may be relevant to the agent's goals.

The second variation modifies the Missing Link domain on a multi-puzzle level. In this Probabilistic Missing Link domain, each puzzle has only some probability of being solvable (that is, that a link exists between the three stems). As with the first variation, this models how only a subset of features are relevant, except in the temporal dimension.

These three domain variations correspond to three cases where spontaneous retrieval is likely to be beneficial. In the Missing Link domain with the single-link representation, the agent does not know the relation of the stem to its compound word (ie. whether it's a prefix or a suffix), and must therefore search memory with both (C3). For Subset Missing Link, the agent does not know which portion of its percepts are important and should be used as the cue to search memory (C2), while for Probabilistic Missing Link, the agent does not have knowledge of whether searching at this time will be beneficial, or if it will just consume resources (C1).

Since the agent can always use brute-force search to find the solution, the metric for this evaluation is not the number of puzzles that the agent solves, but the amount of computation needed to do so, in both decision cycles and real time.

Agent Design

In each puzzle, the domain presents the stems as strings. Both agents with and without spontaneous retrieval must first retrieve the long-term memory representations of the stems. To correctly solve a Missing Link puzzle, an agent without spontaneous retrieval must retrieve all compound words that contain each stem, as well as the other stem that forms those compound words, before finally checking if a missing link exists that is shared by the compound words of all three stems. The differences in representation only determine how quickly this can be done. With the single-link representation, the agent must do all of these steps. With the double-link representation, the agent does not need to retrieve the compound words. With the direct-link representation, the agent can directly check for a missing link. This is true in the two domain variations as well: the stems must be expanded to detect which three have a missing link, just as they must be expanded to determine whether a solution exists for any three stems.

A different search process is possible with spontaneous retrieval. When the memory element representing each stem is retrieved, activation spreads to its compound words and to the other prefix/suffix. Since the missing link is the only word which receives three activation boosts, it has the highest activation and is the first element to be spontaneously retrieved. The agent can then verify that the spontaneously retrieved word forms compound words with all three stems. The benefit of spontaneous retrieval is therefore that the stem–compound-word–stem connections do not need to be deliberately explored. Alternately, spontaneous retrieval can be thought of as using spreading activation to specify a search criteria based on the *graph topology* around the cue elements, one that is not constrained to return elements that must be direct neighbors of the cue.

Deliberate and spontaneous retrievals should not be thought of as separate strategies, however, but as two mechanisms working towards the same goal. Spontaneous retrieval is returning elements that the agent could deliberately retrieve, given the correct sequence of retrievals with the correct cues — since in this domain deliberate retrieval is only used for brute-force search, spontaneous retrieval allows the agent to skip ahead in its reasoning. However, there are also cases where spontaneous retrieval may mislead the agent; for the Subset Missing Link domain in particular, the retrieval of the distractors could be interleaved with the retrieval of the real stems, allowing time for the activation of the missing link to decay. This may cause a more-recently-boosted element to be returned by spontaneous retrieval instead, which would fail the verification. In this case, the agent must resort to the deliberate strategy, at least until a different element is spontaneous retrieved.

It should be noted that the spontaneity of spontaneous retrieval is not strictly necessary for this domain. Since the puzzles are episodic, one can imagine the agent deliberately retrieving the solution when a new puzzle is presented, but using a mechanism that returns that most highly activated

element. This agent would do better than a spontaneous retrieval agent, since it is taking advantage of the episodic nature of the environment. To demonstrate the benefits of spontaneous retrieval, however, the agent uses the deliberate, brute-force search, but takes advantage of any spontaneously retrieved knowledge; that is, spontaneous retrieval provides usable knowledge without being asked to do so. The focus is on spontaneous retrieval complementing deliberate retrieval, especially when brute force becomes more expensive in the Subset and Probabilistic Missing Link variations.

For this reason, agents must be designed so that they can move from deliberate retrievals to spontaneous retrievals and back, and be able to integrate information from both. This can be achieved by conditioning the processing of retrieved memory elements not by the mechanism that retrieved it, but by the information that it represents. For example, in the Missing Link domain, if the retrieved element is a compound word that contains a stem, it should be stored so that the other prefix/suffix can be retrieved later; on the other hand, if the retrieved element is not connected to the stems, it may be the missing link, and it should be verified as the answer. The key is that these behaviors depend only on the agent state and how the retrieval should be used, but not on whether the retrieval was deliberate or spontaneous. Rules that process retrievals only match on the features of the retrieved element, and not on which mechanism retrieved the element or why it is retrieved; in fact, neither mechanism provides the latter. By separating the processing of information from the its source, the agent can effectively use both the knowledge that it knows it needs as well as any spontaneously retrieved shortcuts.

Results

All results in this section are averaged over 100 puzzles. Since each puzzle is independent, the agent is reset between puzzles to negate any recency and frequency effects of activation. The knowledge of compound words and their stems is loaded into the agent's long-term memory before each puzzle. Spreading activation is limited to a distance of two, the distance necessary to reach the solution word (from a stem to its compound words, then from the compound words to the missing link); we briefly discuss the effects of alternate settings of this parameter in the conclusion. All agents are written as deliberate agents, with the processing of retrieved elements then modified to be agnostic to the retrieval mechanism. In fact, the "deliberate" and "spontaneous" agents have the exact same rules, with the only difference being that the architecture provides spontaneous retrievals for the "spontaneous" agent. No effort was made to space deliberate retrievals so that spontaneous retrievals could occur.

Efficiency of Spontaneous Retrieval

To evaluate the efficiency of spontaneous retrieval, we instrumented Soar to measure the amount of time spent calculating the effects of spreading activation, versus that of selecting the most highly activated element in long-term memory. Averaged over 100 puzzles, an agent with the single-link representation takes an average of 350.7 milliseconds to solve a puzzle, of which 335.2 milliseconds is spent updating activation values, and only 0.85 milliseconds for spontaneous retrieval; this sub-millisecond cost over multiple decision cycles is inconsequential given that cognitive architectures limit each individual decision cycle to 50 milliseconds.

Since spreading activation also affects the outcome of deliberate retrievals, it is included in the results below for both the deliberate and spontaneous agents.

Missing Link

The purpose of this experiment is to demonstrate that spontaneous retrieval can overcome a lack of knowledge of how percepts relate to the desired knowledge. A brute-force search would require the agent to iterate through all possible relations in the worst case, although there are only two possible relations (ie. prefix or suffix) in this domain. The average number of decision cycles and amount of real time needed to solve a puzzle by different agents are shown in the table below.

Knowledge	**Decision Cycles**		**Real Time (msec)**	
Rep.	Delib.	Spon.	Delib.	Spon.
Single	56.1	24.5	2631.6	350.7
Double	60.9	60.9	1167.7	1173.1
Direct	12.0	12.0	267.8	268.3

For the single-link representation, the spontaneous retrieval agent takes half the number of decisions to complete the task, and an even smaller proportion of real time. As hypothesized during the discussion of agent design, this is because activation spreads to the missing link, which spontaneous retrieval then puts into working memory. This eliminates many decision cycles during which the deliberate approach is exhaustively expanding the stems and the compound words. The difference in real time is more dramatic because of the cost of spreading activation — since there are fewer retrievals, there are fewer boosts to base-level activation and therefore fewer spreads. This difference would be smaller with a more efficient spreading activation algorithm.

Spontaneous retrieval has little effect on either metric for the other two representations, although the source of this lack of differentiation is different. For the direct-link representation, the missing link could be immediately determined after the stem elements are retrieved, removing the need to search long-term memory. For the double-link representation, the majority of agent processing is from retrieving the other prefix/suffix of the compound words; there are no gaps between these uses of long-term memory, leaving no opportunity for spontaneous retrievals to occur. This raises questions about balancing deliberate and spontaneous retrievals, as well as whether spontaneous retrievals should be placed in a different buffer; these issues are addressed in the conclusion. Regardless, although spontaneous retrievals has no benefits when these knowledge representations are used, it also incurs minimal cost.

The results across all three knowledge representations agree with the original insight that spontaneous retrieval supplements deliberate retrieval when the agent lacks search-guidance knowledge. The more optimized the knowledge base for a task — as more connections are added between

stems and their links — the more effective deliberate retrieval becomes, and the less spontaneous retrieval can offer. On the other hand, spontaneous retrieval can provide the agent with relevant knowledge when brute-force search is necessary, greatly reducing processing time. Since the single-link representation best highlights the difference between deliberate and spontaneous retrievals, it is the only representation used for the remaining results.

Subset Missing Link

The presence of distractors in the Subset Missing Link domain means that the agent must retrieve more compound words before the missing link can be identified. In terms of search, this increases the number of initial states (memory elements), thereby increasing the size of the search "frontier." To be explicit, distractors are presented in addition to the three stem words, so the agent is presented with five clue words for a puzzle with two distractors. Results from this domain variation are shown in the table below.

No. of Distractors	**Decision Cycles**		**Real Time (msec)**	
	Delib.	Spon.	Delib.	Spon.
0	56.1	24.5	2631.6	350.7
1	65.1	26.6	3048.6	341.4
2	70.9	28.2	5752.1	319.4
3	80.8	32.8	3679.1	613.5
4	84.5	40.0	17591.5	840.0

Although both agents with and without spontaneous retrieval require more resources to deal with distractors, the agent with spontaneous retrieval requires less additional resources to do so. For the spontaneous retrieval agent, this growth is due to the extra decision cycles necessary to retrieve the long-term memory representations of the distractors. The deliberate retrieval agent, however, also needs to search for all the compound words for those distractors, hence the larger increase. These results suggest that, for domains where the relevant percepts are not obvious, search-guidance knowledge is doubly important, as search grows exponentially with the number of irrelevant percepts. This is true with spontaneous retrieval as well, but using activation as a heuristic reduces the exponential.

Despite this increase in resource consumption, spontaneous retrieval allows the agent to sidestep the selection of a cue for deliberate search, and continues to reduce the amount of computation necessary for the agent to solve the puzzle.

Probabilistic Missing Link

The goal of this experiment is to show that spontaneous retrieval can overcome the lack of knowledge of when to retrieve from long-term memory. Unlike the previous experiments, here the role of spontaneously retrieved elements is not only to provide the correct answer, but also to be a heuristic for whether an answer exists at all. This additional assumption is reflected in how the agent with spontaneous retrieval does not continue to retrieve words until the solution is found. Instead, when a new puzzle is presented to the agent, it simply attempts to verify whether the first spontaneously retrieved element is the missing link. The results of two different verification procedures are shown: the more costly method (V1) retrieves all compound words of the potential solution, while the more efficient method (V2) directly checks for compound words formed by the potential solution and the stems. For the agent without spontaneous retrieval, no such heuristic for solvability is available, and it must therefore deliberately retrieve all compound words before giving up.

Prob. Solvable	**Decision Cycles**		**Real Time (msec)**		
	Delib.	Spon. V1	Delib.	Spon. V1	Spon. V2
1.0	56.1	24.5	2631.6	350.7	100.5
0.9	56.1	25.9	2602.9	442.4	159.6
0.8	55.5	27.6	2464.0	628.4	165.4
0.7	56.0	29.8	2555.8	744.8	178.0
0.6	55.9	30.5	2459.8	777.9	179.9
0.5	53.9	31.9	2099.2	861.9	186.5
0.4	51.4	35.3	1677.9	1135.3	218.3
0.3	50.3	38.1	1574.2	1450.2	247.0
0.2	49.5	40.0	1400.6	1633.9	263.4
0.1	49.0	41.2	1324.1	1768.4	264.3
0.0	47.8	42.7	1099.5	1858.6	316.8

The results for the single-link agents are shown in the table above. Although the agent with spontaneous retrieval can often identify solvable puzzles in fewer decisions than agents without spontaneous retrieval, whether it can do so in less real time depends on the verification method. The deliberate agent gives up sooner on unsolvable puzzles, since those stems tend to have fewer compound words (as it would otherwise increase the probability that a missing link exists). No such trend exists for the spontaneously retrieved element: a word that has the most neighbors would have the highest activation, regardless of whether the puzzle is solvable. This is exacerbated by the quadratic time necessary to complete the naive method of iterating through all compound words (V1), which only takes linear time for the more efficient method (V2). As spontaneous retrieval is less and less likely to return the solution, the verification of the solution could take more time than iterating through the compound words, leading to the tradeoff seen in the results for the naive verification method.

Assuming the cost of verification is low, these results show that spontaneous retrieval can cheaply provide the agent with knowledge, even if the agent is uncertain whether that knowledge exists.

Discussion

In all three variations of the Missing Link domain, the use of spontaneous retrieval leads to agents that more quickly complete their tasks. Spontaneous retrieval provides a shortcut through brute-force search; allows the agent to efficiently ignore distractors; and indicates whether relevant knowledge exists in long-term memory. Both the spontaneity and the spreading activation bias are necessary for these benefits: spreading activation reduces the time needed for search, while the spontaneity allows the agent to search less frequently. Together, these results provide evidence for our hypothesis that spontaneous retrieval can supplement deliberate retrieval

as a heuristic to overcome a lack of knowledge of when to search and what to search for.

Since spontaneous retrieval is a new mechanism, there are many questions that have not been explored in this paper. A high level question is how the agent should decide whether to use deliberate retrieval or to rely on spontaneous retrieval. This is an exploration-exploitation tradeoff: deliberate retrieval can be exploited if the search-guidance knowledge is correct, while spontaneous retrieval — a cue-less, unguided exploration of long-term memory — can mitigate incorrect search knowledge and provide shortcuts to relevant knowledge, but may also consume resources for verification. This formulation of the problem suggests that reinforcement learning techniques can be applied, to generally learn the utility of each mechanism, or to fine-tune whether spontaneous retrieval should be used in a particular context. Alternately, spontaneously retrieved elements could be placed in a separate buffer so that it is always available; however, the integration of information from the different mechanisms would become more difficult.

Another major area of research is the uncued selection of a long-term memory element. Several different algorithms for spreading activation were discussed in the agent design section, while other definitions of "relevance" may bring to mind algorithms from case-based reasoning or analogical reasoning (Riesbeck and Schank 1989; Pickett and Aha 2013). These choices are likely to also be influenced by the structure of the knowledge base, such as the average in- and out-degrees or the diameter of the knowledge graph, as well as the distance to a relevant memory element.

In a preliminary investigation of these issues, we varied the distance over which activation spreads, in the single-link Missing Link domain. While decreasing the limit merely reduced performance to the level of deliberate retrieval, increasing the limit increased the number of decision cycles and the amount of real time needed by a spontaneous retrieval agent. The real-time growth is mostly caused by spreading, as the activation of exponentially more long-term memory elements must be boosted. This led to unwanted elements having higher activation than the desired elements, leading to the retrieval of the former and forcing the agent to attempt verification before failing, hence the growth in the number of decision cycles. It is possible that this problem could be sidestepped by using other models of spreading activation, such as one where the boost decreases over distance.

Finally, further research is needed to identify domains where agents lack search-guidance knowledge and to determine whether spontaneous retrieval can help in those cases. The results presented here suggest environmental properties that may limit the effectiveness of spontaneous retrieval, such as the degree to which long-term memory is optimized for a task, the presence of distractors, and the cost of verification. In goal management, for example, where spontaneous retrieval may be applicable (Li and Laird 2013), these factors translate into properties such as the complexity of the description of the goal and the amount of other knowledge that the agent has. These factors should be taken into account when deciding whether to supplement deliberate retrieval with spontaneous retrieval.

Acknowledgments

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 7(1):39–59.

Anderson, J. R. 2007. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Chen, Y.; Petrovic, M.; and Clark, M. H. 2014. SemMemDB: In-database knowledge activation. In *Proceedings of the 27th International Florida Artificial Intelligence Research Society Conference*, 18–23.

Derbinsky, N.; Laird, J. E.; and Smith, B. 2010. Towards efficiently supporting large symbolic declarative memories. In *Proceedings of the 10th International Conference on Cognitive Modeling*, 49–54.

Douglass, S. A., and Myers, C. W. 2010. Concurrent knowledge activation calculation in large declarative memories. In *Proceedings of the 10th International Conference on Cognitive Modeling*, 55–60.

Ebbinghaus, H. 1913. *Über das Gedächtnis (Memory: A Contribution to Experimental Psychology)*. Columbia University.

Hintzman, D. L. 2011. Research strategy in the study of memory: Fads, fallacies, and the search for the "coordinates of truth". *Perspectives on Psychological Science* 6(3):253–271.

Kvavilashvili, L., and Mandler, G. 2004. Out of one's mind: A study of involuntary semantic memories. *Cognitive Psychology* 48(1):47–94.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. MIT Press.

Lebiere, C., and Best, B. 2009. Balancing long-term reinforcement and short-term inhibition. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Lebiere, C. 2014. Personal communication, August 27th, 2014.

Li, J., and Laird, J. E. 2013. The computational problem of prospective memory retrieval. In *Proceedings of the 12th International Conference on Cognitive Modeling*, 155–160.

Mednick, S. A. 1962. The associative basis of the creative process. *Psychological Review* 69(3):220–232.

Newell, A. 1990. *Unified Theories of Cognition*. Harvard University Press.

Pickett, M., and Aha, D. W. 2013. Spontaneous analogy by piggybacking on a perceptual system. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, 3229–3234.

Raaijmakers, J. G. W., and Shiffrin, R. M. 1981. Search of associative memory. *Psychological Review* 88(2):93–134.

Riesbeck, C. K., and Schank, R. C. S. 1989. *Inside Case-Based Reasoning*. Psychology Press.