Lifelong Forgetting: A Critical Ingredient of Lifelong Learning, and Its Implementation in the OpenCog Integrative AI Framework

Ben Goertzel
Novamente LLC
Fujian Province Key Lab for Brain-Like Intelligent Systems,
Dept. of Cognitive Science, Xiamen University

Your fingers weave quick minarets ... Speak in secret alphabets ... I light another cigarette ... **Learn to forget, learn to forget**

– Jim Morrison, ”Soul Kitchen”

Of all the aspects differentiating lifelong learning from shorter-term, more specialized learning, perhaps none is more central than forgetting – or, to frame the issue more generally and technically, ”memory access speed deprioritization.” This extended abstract reviews some of the ideas involved in forgetting for lifelong learning systems, and briefly discusses the forgetting mechanisms used in the OpenCog integrative cognitive architecture.

Defining Forgetting In ordinary human discourse, the word ”forget” has multiple shades of meaning. It can refer to the irreversible elimination of a certain knowledge item from memory; or it can mean something milder, as in cases where someone ”forgets” something, but then remembers it shortly after. In the latter case, ”forgetting” means that the knowledge item has been stored in some portion of memory from which access is slow and uncertain.

These various shades of meaning also have relevance to AI systems performing lifelong learning. The advent of larger and cheaper memory stores makes it more and more viable for an AI system to persistently store a large percentage of its experiences. But nevertheless, memories with faster access remain considerably more expensive than those with slower access. So for example, in the case of an AI system that stores many of its thoughts and experiences to disk but keeps only a currently pertinent subset in RAM, the basic problem of ”forgetting” still remains, taking the form of deciding which information to keep in RAM and which to push to the ”back of the mind” on disk.

In general one may define ”forgetting” as the process of storing different memory items in different ways associated with dramatically different access speeds, the outright deletion of items from memory being one extreme. In any lifelong learning system containing a memory store with differentially rapid access, that is much too small to contain the system's whole experience and knowledge base, forgetting will be a critical process.

Notwithstanding its general neglect in the AI field, forgetting is extremely important – basically, it's about learning what is likely to be most important to be able to access in the future, with what level of access speed.

Forgetting in OpenCog Next we very briefly describe the OCP (OCP) Artificial General Intelligence architecture, implemented within the open-source OpenCog AI framework, with a focus on those aspects most relevant to forgetting. Other cognitive architectures that have paid particular attention to forgetting include Pei Wang's NARS (Wang 2006), Stan Franklin's LIDA (Franklin 2006), and many more.

Conceptually founded on the ”patternist” systems theory of intelligence outlined in (Goertzel 2006), OCP combines multiple AI paradigms such as uncertain logic, computational linguistics, evolutionary program learning and connectionist attention allocation in a unified architecture. Cognitive processes embodying these different paradigms interoperate together on a common neural-symbolic knowledge store called the Atomspace. The interaction of these processes is designed to encourage the self-organizing emergence of high-level network structures in the Atomspace, including superposed hierarchical and heterarchical knowledge networks, and a self-model network.

OpenCog has been used for commercial applications in the area of natural language processing and data mining; e.g. see (Goertzel et al. 2006) where OpenCog's PLN reasoning and RelEx language processing are combined to do automated biological hypothesis generation based on information gathered from PubMed abstracts. Most relevantly to the present proposal, has also been used to control virtual agents in virtual worlds (Goertzel and Et Al 2008), using an OpenCog variant called the OpenPetBrain, and humanoid robots (Goertzel 2010). These agents demonstrate a variety of interesting and relevant functionalities including learning new behaviors based on imitation and reinforcement; responding to natural language commands and questions, with appropriate actions and natural language replies; and spontaneous exploration of their world, remembering their experiences and using them to bias future learning and linguistic interaction. In the virtual agent and physical robotics contexts, OpenCog is intended for lifelong learning (although experiments so far have not run more than weeks).

Declarative knowledge representation in OpenCog is handled by a weighted labeled hypergraph called the Atomspace, which consists of multiple types of nodes and links, generally weighted with probabilistic truth values and also attention values (ShortTermImportance (STI) and LongTermImportance (LTI) values, regulating processor and memory use). Equations called Economic Attention Networks (ECAN) (Goertzel et al. 2010) are used to update the STI and LTI values dynamically, where the attention values associated with an Atom are dependent on how useful the Atom is estimated to be for achieving the system's future goals (short or long term, respectively), and by STI and LTI spreading to it along links from other Atoms, roughly similarly to activation spreading in a neural network.

Forgetting in OpenCog And when OpenCog's declarative memory becomes full, how does forgetting happen? The Atoms with the lowest LTI are removed from RAM. Atoms also have a single bit variable called VLTI (V=very), which determines whether, upon removal from RAM, the Atom is saved to disk or just deleted. For instance, recently created speculative concepts that prove useless will generally just be deleted; whereas any Atom that has been significantly important in the past will likely be saved to disk.

When an Atom is saved to disk, what happens to the other Atoms in RAM that link to it? They retain a certain percentage of these links, which point to AtomHandles rather than Atom objects. To follow those links is an expensive operation undertaken only under special circumstances, as it involves loading the target Atoms into RAM. How many of these disk-directed links are retained is determined by the LTI of the linking Atom in RAM, and a system parameter.

Interdependency of Judgments about Forgetting A subtlety is that there are many cases where one has a large set of memory items, so that none of the items individually is particularly important to remember, but so that it's important for the memory to retain some of the memory items in rapid-access memory rather than deleting them all or relegating them all to slow-access memory. This means that judgments about forgetting must be made on the level of memory networks.

To understand this better, consider again a system like OpenCog that retains many memory items in RAM, but relegates many more to disk. In this case, there is a major issue of how the system knows what it has stored on disk. One way of extracting information from the disk store is to use named Atoms such as those corresponding to natural language words. So if the system wants to know more about Australia, for instance, it can search its disk store and see if it previously stored there any Atoms linked to the "Australia" WordNode. Once it has imported some of these Atoms into RAM, it can then import other Atoms linked to those, etc. However, this is a fairly crude method, which doesn't help that much with recall of the system's own novel conceptions (that may not be easily indexed using natural language words or other persistent external references). Suppose a system has learned a novel network of 1000 concepts and their interrelationships, which it thinks may be useful to it in the future, but which it feels it can't afford to retain in RAM. What's its best forgetting strategy?

One solution to this issue is for the system to retain a random (or more judiciously chosen) subsample of the 1000-concept network in RAM. Even if no individual Atom in the network is all that important in itself, retaining some of the network's Atoms in RAM is a valuable thing, as these remaining RAM-resident Atoms can be used to bring the other members of the network back into RAM as appropriate. This phenomenon is accounted for in OpenCog via a special formula that boosts the LTI of an Atom if is is linked to relatively high-LTI Atoms on disk, which have few links to other Atoms in RAM. The quantitative weighting of this formula controls, in our example, how many of the Atoms in the 1000-concept network will remain in RAM (depending also on the LTI dynamics of the rest of the network).

Conclusion We have reviewed some of the general issues involved with forgetting in lifelong learning systems, and summarized how these issues are solved in the OpenCog system. As OpenCog is architected quite differently than the human brain, one expects the brain's approach to resolving the same problem to have many different characteristics, as well as likely some similarities. In general, different lifelong learning systems may handle the subtleties of forgetting differently, but all must address the same issues, such as the interdependency of judgments about forgetting; and these are issues that are largely particular to lifelong learning systems, not arising often in AI systems that are booted up only temporarily for the solution of individual problems.

This work partially funded by Chinese NSF Grant 60975084/F030603

References

Franklin, S. 2006. The lida architecture: Adding new modes of learning to an intelligent, autonomous, software agent. *Int. Conf. on Integrated Design and Process Technology*.

Goertzel, B., and Et Al, C. P. 2008. An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In *Proceedings of the First Conference on Artificial General Intelligence*. IOS Press.

Goertzel, B.; Pinto, H.; Pennachin, C.; and Goertzel, I. F. 2006. Using dependency parsing and probabilistic inference to extract relationships between genes, proteins and malignancies implicit among multiple biomedical research abstracts. In *Proceedings of Bio-NLP 2006*.

Goertzel, B.; Pitt, J.; Ikle, M.; Pennachin, C.; and Liu, R. 2010. Glocal memory: a design principle for artificial brains and minds. *Neurocomputing, Special Issue of Artificial Brain*.

Goertzel, B. 2006. *The Hidden Pattern*. Brown Walker.

Goertzel, B. e. a. 2010. Opencogbot: An integrative architecture for embodied agi. *Proceedings of ICAI-10, Beijing*.

Wang, P. 2006. *Rigid Flexibility: The Logic of Intelligence*. Springer.