An Expanded Set of Declarative Memory Functionalities in PyACTUp, a Python Implementation of ACT-UP's Accountable Modeling

Yuxue C. Yang (chery@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Don Morrison (dfm2@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Andrea Stocco (stocco@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Mark Orr (mo6xj@virginia.edu)
Biocomplexity Institute, University of Virginia,
P.O. Box 400298, Charlottesville, VA 22904 US

Christian Lebiere (cl@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Keywords: Cognitive Architecture; Accountable Modeling; ACT-R; ACT-UP; Python; Agile Development

Introduction

ACT-R (Anderson, 2007) is the most influential cognitive architecture in psychology and neuroscience (Kotseruba and Tsotsos, 2018). To enforce its commitment on *end-to-end* modeling (that is, the constraint that the modeler should consider *all aspects* of a task), ACT-R is purposely designed to paradigmatically discourage the developer from taking shortcuts, which is often frustrating.

In 2010, Reitter and Lebiere proposed the alternative paradigm of *accountable modeling*. Instead of discouraging developers from taking shortcuts, it encourages them to explicitly specify which components of an architecture they plan to use and how they account for aspects that are not modeled using the architecture (e.g., through parameter estimation). As a proof of concept, they designed ACT-UP, a modular Lisp implementation of ACT-R's declarative memory system that can be used independently of the other components. Despite its successes, ACT-UP was limited by (a) being written in Lisp, (b) covering only a limited set of functions; and (c) inheriting ACT-R's traditional assumption that memories are equal in terms of importance.

Here, we present PyACTUp, a Python implementation of ACT-UP that takes advantage of Python's modern language design, and extensive graphic and scientific libraries, and further extends the set of declarative memory functionalities, including spreading activation and emotional components. Both of these core functionalities have received widespread attention in contemporary models, and are now implemented without any reference to ACT-R's other first-level structures (buffers or productions). Github: https://cher_yang@bitbucket.org/cher_yang/pyactup2.git

General PyACTUp Architecture

PyACTUp closely follows Reitter & Lebiere's (2010) original implementation of ACT-UP, doing away with buffers and procedural knowledge and simplifying the process of modeling memory encoding and retrieval, so that modelers are able to focus on essential cognitive phenomena instead of being stopped by programming difficulties.

The most fundamental functions of PyACTUp are learning and retrieving memories. As pieces of information are *learned*, they are stored in declarative memory as *chunks*. The memory contains one or more *slot-value* pairs, and can be *retrieved* by specifying all or a subset of identifying *cues,* also in the form of slot-value pairs. A new declarative memory is created as an instance of the `Memory` class object, i.e. `dm = Memory()`. Chunks are represented using Python's built-in dictionary type, and are learned and retrieved using Python's keyword-argument syntax to manage slot-value pairs. For example, the following line:

```
dm.learn(guy="Ringo",role="Drummer")
```

creates a chunk for Ringo Starr of The Beatles. To retrieve him, one can type

```
dm.retrieve(role="Drummer")
```

As in ACT-R, chunks in a `Memory` structure are retrieved based on their *activation*, a scalar meta-quantity that reflects frequency (the retrieval probability odds) and the recency (decays over time). Depending on the number of memories available in a `Memory` structure, it is possible to enable *blending* and retrieve *blended* memories instead of a specific chunk.

Implementation of Spreading Activation

In ACT-R, a chunk's activation is made of a *base-level* (the part that decays over time) and a *spreading* component, which increases a chunk's activation in proportion to its association to other chunks. Because spreading activation originates in ACT-R buffers, ACT-UP and its original Python counterpart originally excluded it from their available functionalities. However, since spreading activation is important to capture phenomena like the fan effect (Anderson, 1974) and working memory (Daily et al 2001), it was introduced as new functionality in this version of PyACTUp. To avoid the use of buffers, activation is spread through a specific function and the use, again of the

keyword-argument syntax, with an argument that defines the source. For example,

```
dm.spread(role="drummer")
```

spreads activation from "Drummer" to other chunks in `dm`. Following the ACT-R standard algorithm, the associative strength between two chunks *i* and *j* is calculated, by default, as inversely proportional to the logarithm of the fan of *i*, i.e. the number of times *i* appears in other chunks other than *j*. Modelers, however, are given flexibility to define their own associative strength functions. The value returned by these functions is then multiplied by the weight.

Differential Importance of Memories

ACT-R and ACT-UP share the assumption that all memories are equal in terms of importance, and therefore they decay at the same rate. This assumption was recognized as a limitation by Anderson himself (2007, chapter 3). Researchers modeling effects of emotion, for example, have repeatedly suggested that affect and emotion alter how memorable certain events are by providing an activation bias or boost (Juvina, Larue, & Hough, 2018, Fum & Stocco, 2004; Cochran et al., 2006).

Without taking a stance in the debate, we decided to provide a way for developers to add their own activation bias terms through an additional importance term, which is linearly additive to base-level and spreading activation. Here, we implement the highest level of conceptual idea of importance. The degree of importance is attached to the nature of memory and to some extent decides the needs of retrieval subsequently. The importance is set directly through learning. For example,

```
dm.learn(guy="Disco",role="Manager",
importance=5)
```

To observe the effects of importance on memory retrieval, we created a model to simulate the retrieval process of memory with various degrees of importance. Highly important memory is set to values from 0 to 3 while normal memory is set to a randomly distributed value from 0 to 2.

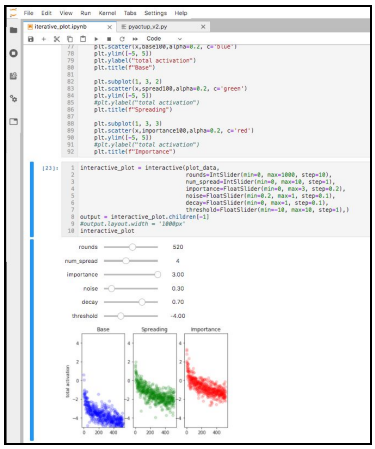

Fig 1. *Interactive Model shows the change of base-level activation over time (blue); base-level + spreading (green); and base-level + spreading + importance (red).*

Interaction Example

Fig 1 provides an example of the code in Jupyter Notebook window. This shows how PyACTUp can be used to quickly run an interactive model in Python. By adjusting parameters, the model shows dynamic memory encoding and retrieving outputs.

Summary

The current version of PyACTUp inherits the simplification from the last version and extends its functionality to include the spreading activation and importance term. We use the new PyACTUp to model spreading activation effects and the retrieval discrepancy between important memory and trivial ones. Admittedly, PyACTUp is only a subset of the ACT-R architecture, so it does have some limitations in modeling human cognition. How to keep PyACTUp simple, flexible and applicable is a trade-off problem for the framework designer. Further researchers could explore other useful functions and expand the current PyACTUp to a more comprehensive, accountable and modular implementation of the ACT-R cognitive modeling architecture.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6*(4), 451–474

Anderson, J. R. (2007). How can the human mind occur in the physical universe? *Oxford University Press*.

Chown, E., Cochran, R. E, & Lee, F. J. (2006). Modeling emotion: Arousal's impact on memory. *Proceedings of the Annual Meeting of the Cognitive Science Society, 28.*

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*(3), 315-353.

Fum, D., & Stocco, A. (2004). Memory, emotion, and rationality: An ACT-R interpretation for gambling task results. In C. D. Schunn, M. C. Lovett, C. Lebiere & P. Munro (Eds.) *Proceedings of the 6th International Conference on Cognitive Modeling.* Mahwah, New Jersey: Lawrence Erlbaum Associates, pp. 106–111.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research, 48*, 4-24.

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, 1-78.

Reitter, D., & Lebiere, C. (2010). Accountable modeling in ACT-UP, a scalable, rapid-prototyping ACT-R implementation. In *Proceedings of the 10th International Conference on Cognitive Modeling, ICCM 2010* (pp. 199-204)