15_lexical_decision_model_running_model_understanding_ouput

April 12, 2021

Model up to this point:

[1]:
```
import pyactr as actr

environment = actr.Environment(focus_position=(0,0))
lex_decision = actr.ACTRModel(
    environment=environment,
    automatic_visual_search=False,
    motor_prepared=True
)
```

[2]:
```
actr.chunktype("goal", "state")
actr.chunktype("word", "form")

dm = lex_decision.decmem

for string in {"elephant", "dog", "crocodile"}:
    dm.add(actr.makechunk(typename="word", form=string))

g = lex_decision.goal

g.add(actr.makechunk(nameofchunk="beginning",
                     typename="goal",
                     state="start"))
```

[3]:
```
lex_decision.productionstring(name="find word", string="""
    =g>
    isa     goal
    state   start
    ?visual_location>
    buffer  empty
    ==>
    =g>
    isa     goal
```

```
    state   attend
    +visual_location>
    isa _visuallocation
    screen_x closest
""")

lex_decision.productionstring(name="attend word", string="""
    =g>
    isa     goal
    state   attend
    =visual_location>
    isa    _visuallocation
    ?visual>
    state   free
    ==>
    =g>
    isa     goal
    state   retrieving
    +visual>
    isa     _visual
    cmd     move_attention
    screen_pos =visual_location
    ~visual_location>
""")

lex_decision.productionstring(name="retrieving", string="""
    =g>
    isa     goal
    state   retrieving
    =visual>
    isa     _visual
    value   =val
    ==>
    =g>
    isa     goal
    state   retrieval_done
    +retrieval>
    isa     word
    form    =val
""")

lex_decision.productionstring(name="lexeme retrieved", string="""
    =g>
    isa     goal
    state   retrieval_done
    ?retrieval>
    buffer  full
```

```
    state   free
    ==>
    =g>
    isa     goal
    state   done
    +manual>
    isa     _manual
    cmd     press_key
    key     J
""")

lex_decision.productionstring(name="no lexeme found", string="""
    =g>
    isa     goal
    state   retrieval_done
    ?retrieval>
    buffer  empty
    state   error
    ==>
    =g>
    isa     goal
    state   done
    +manual>
    isa     _manual
    cmd     press_key
    key     F
""");
```

0.1 Running the lexical decision model and understanding the output

Before we run the simulation of the model, we have to specify the set of stimuli (character strings) that should appear on the screen.

- we use a dictionary for that

Below, we specify that our first – and only – stimulus is the word *elephant*, which should be displayed on the screen starting at pixel $\langle 320, 180 \rangle$.

[4]:
```
word = {1: {'text': 'elephant', 'position': (320, 180)}}
```

We are now ready to initialize the simulation:

[5]:
```
lex_dec_sim = lex_decision.simulation(
    realtime=False,
    gui=False,
    environment_process=environment.environment_process,
    stimuli=word,
    triggers='',
```

```
    times=1)
```

- the first parameter, namely `realtime` (line 2 above), states that the simulation should not appear in real time
  - setting this parameter to `True` ensures that the simulation will take the same amount of real time as the model predicts
  - otherwise, the simulation is executed as fast as the processing speed of the computer allows it
  - this does not affect the actual model and its predictions in any way, it only affects the way the simulation is played out
- the second parameter `gui` (line 3) specifies whether a graphical user interface should be started in a separate window to represent the environment, i.e., the virtual screen on which the stimuli are displayed
  - this option is switched off here, but feel free to switch it on by setting `gui` to `True`
- the third argument (line 4) states what environment process should appear in our environment
  - you can create your own, but there is one predefined in the `Environment` class that displays stimuli from the list above one at a time on the virtual screen
- the stimulus list to be displayed in the environment is specified by the fourth parameter (line 5)
- the penultimate parameter is `triggers` (line 6), which specifies the triggers that the process should respond to
  - we do not care about any triggers here, so we leave that list empty
- the final parameter is `times` (line 7), which specifies that each stimulus should be displayed for 1 s max

The simulation can now be run:

[6]:
```
lex_dec_sim.run()
```

```
(0, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0, 'PROCEDURAL', 'RULE SELECTED: find word')
****Environment: {1: {'text': 'elephant', 'position': (320, 180)}}
(0.05, 'PROCEDURAL', 'RULE FIRED: find word')
(0.05, 'g', 'MODIFIED')
(0.05, 'visual_location', 'CLEARED')
(0.05, 'visual_location', "ENCODED LOCATION:'_visuallocation(color= None,
screen_x= 320, screen_y= 180, value= None)'")
(0.05, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.05, 'PROCEDURAL', 'RULE SELECTED: attend word')
(0.1, 'PROCEDURAL', 'RULE FIRED: attend word')
(0.1, 'g', 'MODIFIED')
(0.1, 'visual_location', 'CLEARED')
(0.1, 'visual', 'PREPARATION TO SHIFT VISUAL ATTENTION STARTED')
```

```
(0.1, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.1, 'PROCEDURAL', 'NO RULE FOUND')
(0.1163, 'visual', 'CLEARED')
(0.1163, 'visual', "ENCODED VIS OBJECT:'_visual(cmd= move_attention, color= ,
screen_pos= _visuallocation(color= None, screen_x= 320, screen_y= 180, value=
None), value= elephant)'")
(0.1163, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.1163, 'PROCEDURAL', 'RULE SELECTED: retrieving')
(0.1663, 'PROCEDURAL', 'RULE FIRED: retrieving')
(0.1663, 'g', 'MODIFIED')
(0.1663, 'retrieval', 'START RETRIEVAL')
(0.1663, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.1663, 'PROCEDURAL', 'NO RULE FOUND')
(0.1916, 'visual', 'PREPARATION TO SHIFT VISUAL ATTENTION COMPLETED')
(0.1916, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.1916, 'PROCEDURAL', 'NO RULE FOUND')
(0.2163, 'retrieval', 'CLEARED')
(0.2163, 'retrieval', 'RETRIEVED: word(form= elephant)')
(0.2163, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.2163, 'PROCEDURAL', 'RULE SELECTED: lexeme retrieved')
(0.2663, 'PROCEDURAL', 'RULE FIRED: lexeme retrieved')
(0.2663, 'g', 'MODIFIED')
(0.2663, 'manual', 'COMMAND: press_key')
(0.2663, 'manual', 'PREPARATION COMPLETE')
(0.2663, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.2663, 'PROCEDURAL', 'NO RULE FOUND')
(0.3162, 'visual', 'SHIFT COMPLETE TO POSITION: [320, 180]')
(0.3162, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.3162, 'PROCEDURAL', 'NO RULE FOUND')
(0.3163, 'manual', 'INITIATION COMPLETE')
(0.3163, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.3163, 'PROCEDURAL', 'NO RULE FOUND')
(0.4163, 'manual', 'KEY PRESSED: J')
(0.4163, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.4163, 'PROCEDURAL', 'NO RULE FOUND')
(0.5663, 'manual', 'MOVEMENT FINISHED')
(0.5663, 'PROCEDURAL', 'CONFLICT RESOLUTION')
(0.5663, 'PROCEDURAL', 'NO RULE FOUND')
```

- we see that it should take a bit more than 400 ms to find a stimulus, decide whether it is a word and press the right key – see the event `'KEY PRESSED: J'` above
- this is slightly faster than the $500 - 600$ ms usually found in lexical decision tasks, but it is a consequence of our inadequate modeling of memory retrieval; consult the papers below for more details about human reaction time in lexical decision tasks:
  - Forster, Kenneth I. 1990. Lexical processing. In *Language: An invitation to cognitive science*, ed. Daniel Osherson and Howard Lasnik, 95–131. Cambridge, MA: MIT Press.
  - Murray, Wayne S, and Kenneth I. Forster. 2004. Serial mechanisms in lexical access: the rank hypothesis. *Psychological Review* 111:721

- while visual and motor processes are fairly realistically modeled, we assume retrieval always takes 50 ms regardless of the specific features of the word we're trying to retrieve and the cognitive state in which retrieval happens
- we will address this when we introduce the sub-symbolic components of ACT-R in future notebooks, which we will then use to model lexical decision tasks much more realistically

The remainder of this notebook is dedicated to discussing the visual and manual processes that are chronicled in the output of the simulation in the trace above.

0.1.1 Visual processes in our lexical decision model

Traditionally, visual attention is equated to (keeping track of) the focus position of the eyes:

- Just, Marcel A., and Patricia A. Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological Review* 87:329–354
- Just, Marcel A., Patricia A. Carpenter, and Jacqueline D. Woolley. 1982. Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General* 111:228–238
- understanding which word one attends to is tantamount to identifying which word the eyes are focused on

But this identification of the unobservable cognitive state (attention) and overt behavior (eye focus position) is an overly simplified model.

- for example, it is known that when people read, some words, especially high-frequency ones, are processed without ever receiving eye focus
  - Schilling, Hildur EH, Keith Rayner, and James I Chumbley. 1998. Comparing naming, lexical decision, and eye fixation times: Word frequency effects and individual differences. *Memory and Cognition* 26:1270–1281
  - Rayner, Keith. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124:372–422

The EMMA model (Salvucci, Dario D. 2001. An integrated model of eye movements and visual encoding. *Cognitive Systems Research* 1:201–220) incorporated in ACT-R and implemented in `pyactr` captures this by disassociating eye focus and attention:

- the two processes are related but not identical
- a shift of attention to a visual object, for example, the command `move_attention` on line 15 above triggers:
  - an immediate attempt to encode the object as an internal representation, and at the same time,
  - eye movement
- but the two processes proceed independently of each other

We first discuss the process of encoding a visual object.

The time t_{enc} needed to encode an object is modeled using a Gamma distribution (a generalization of the exponential distribution) with mean T_{enc} and standard deviation one third of the mean.

- $t_{enc} \sim Gamma(\mu = T_{enc}, \sigma = T_{enc}/3)$
- $T_{enc} = K \cdot (-\log f) \cdot e^{kd}$, where:
  - f is the (normalized) frequency of the object (word) being encoded

  - d is the distance between the current focal point of the eyes and the object to be encoded measured in degrees of visual angle (d is the eccentricity of the object relative to the current eye position)
    * that is, the parameter T_{enc} is crucially parametrized by the distance d between the current eye focus position and the position of the target object
  - k is a free parameter, scaling the effect of distance (set to 1 by default)
  - K is a free parameter, scaling the encoding time itself (set to 0.01 by default)

In the trace of the simulation, the time point of encoding a visual object is signaled by the event `ENCODED VIS OBJECT`.

Note about parametrizing Gamma distributions

Gamma distributions are usually parametrized in terms of a shape α and a rate β or scale $\frac{1}{\beta}$. We can convert our non-standard parametrization into the standard one(s) as follows: shape $\alpha = (\frac{\mu}{\sigma})^2$ and rate $\beta = \frac{\mu}{\sigma^2}$ (equivalently: scale $\frac{1}{\beta} = \frac{\sigma^2}{\mu}$).

Let us turn now to discussing the eye movement process.

The time needed for eye movement to the new object is split into two sub-processes:

- preparation, and
- execution

The preparation is modeled once again as a Gamma distribution with mean 135 ms and a standard deviation of 45 ms (yet again, the standard deviation is one third of the mean).

The execution, which follows the preparation, is also modeled as a Gamma distribution with:

- a mean of 70 ms + 2 ms for every degree of visual angle between the current eye position and the targeted visual object, and
- a standard deviation that is one third of the mean

It is only at the end of the execution sub-process that the eyes focus on the new position.

- thus, the whole process of eye movement takes around 200 ms ($\approx 135 + 70$), which corresponds to average saccade latencies reported in previous studies (see, e.g., Fuchs, Albert. 1971. The saccadic system. *The control of eye movements* 343–362).

In our simulation:

- the event `PREPARATION TO SHIFT VISUAL ATTENTION COMPLETED` signals the end of the preparation phase
- the end of the execution phase is signaled by `SHIFT COMPLETE TO POSITION [320, 180]`

It is only at this point that the eyes focus on the new location, but the internal representation of the object has already been encoded:

- the word has already been retrieved from memory by this point, as indicated by the earlier event `RETRIEVED: word(form= elephant)`

How do the two processes of visual encoding and eye movement interact?

- one possibility is that encoding is done before the end of the preparation phase
  - this is actually the case in our trace above

  - when this happens, the planned eye movement can be canceled, but only if the cognitive processes following visual encoding are fast enough to
    * cancel the eye shift, or
    * request a new eye-focus position before the end of the preparation phase
- the second possibility is that visual encoding is finished only during the execution phase of the eye movement process
  - in this case, eye movement cannot be stopped anymore and the eye shift is actually carried out
- the third and final possibility is that visual encoding is still not done after the eyes shift to a new position
  - in this case, visual encoding is restarted
  - since the eyes have moved closer to the position of the object we're trying to encode, the time necessary for visual encoding is now decreased

To understand how the restarted visual encoding time is decreased, consider what the new encoding time would have been if this had been an initial visual encoding.

- we would have a random draw t'_{enc} from a Gamma distribution centered at a new mean T'_{enc} because the distance between the object and the new position of the eyes has now changed to d'
  - $t'_{enc} \sim Gamma(\mu = T'_{enc}, \sigma = T'_{enc}/3)$
  - $T'_{enc} = K \cdot (-\log f) \cdot e^{kd'}$

But instead of taking the full t'_{enc} time to do the visual encoding, we will scale that down by the amount of time we already spent during our initial encoding attempt:

- we will look at the initial expected encoding time t_{enc} and at the time $t_{completed}$ that we actually spent encoding
  - necessarily, $t_{completed} < t_{enc}$
- we can say that we have already completed a percentage of the visual encoding process, and that percentage is $\frac{t_{completed}}{t_{enc}}$
- the new processing time should be the remaining percentage that we have not completed yet, i.e.:
  - $\frac{t_{enc} - t_{completed}}{t_{enc}}$, or equivalently
  - $1 - \frac{t_{completed}}{t_{enc}}$
- thus, instead of saying that the new encoding time is the full t'_{enc}, we will only need the percentage of it that is the same as the percentage of incomplete processing we had left after our first encoding attempt
  - visual reencoding time: $\left(1 - \frac{t_{completed}}{t_{enc}}\right) \cdot t'_{enc}$

0.1.2 Manual processes in our lexical decision model

Similarly to the vision process, the motor process is split into several sub-phases when carrying out a command:

- the preparation phase
- the initiation phase
- the actual key press
- finishing the movement (returning to the original position)

As in the case of the visual module, cognitive processes can interrupt a movement, but only during the preparation phase.

The time needed to carry out every phase is dependent on several variables:

- is this the first movement or not? if a key was pressed before, was it pressed with the same hand or not?
  - answers to these questions influence the amount of time the preparation phase takes
- is the key to be pressed on the home row or not?
  - the answer to this question influences the amount of time the actual movement requires, as well as the preparation phase

[ ]: