Quantitative Comparison for Generative Theories:
Embedding Competence Linguistic Theories in Cognitive Architectures and Bayesian Models

Adrian Brasoveanu & Jakub Dotlačil

BLS 44, UC Berkeley · February 9, 2018
The main goal

Introduce a new framework integrating generative theories, ACT-R models, and Bayesian methods.

i. Generative theories + ACT-R: competence-level generative theories are embedded in performance-level processing ACT-R models
   (Anderson and Lebiere 1998, Lewis and Vasishth 2005 a.o.)
   - this enables us to explicitly and fully model the behavior of human participants in standard experimental tasks
     (lexical decision, forced-choice, self-paced reading, eye-tracking)

This is computationally implemented in a new Python3 library: pyactr, https://github.com/jakdot/pyactr.
(If you use this Python3 library, please cite it as Brasoveanu and Dotlačil (2018, in prep.) and include the github url.)
The main goal

ii. **ACT-R + Bayes**: the ACT-R models are embedded in Bayesian models; we can then fit them to experimental data and do quantitative comparison for qualitative theories

- **pyactr** enables us to easily interface ACT-R models with standard statistical estimation methods implemented in widely-used Python3 libraries

- we use ACT-R models as the likelihood component of full Bayesian models, and fit the ACT-R parameters to experimental data

- upshot: we are able to consider alternative generative grammar theories and quantitatively compare how well they fit experimental data
The main goal

The ability to do quantitative comparison for qualitative generative theories on this scale is unprecedented (as far as we know).

• even in ACT-R, subsymbolic/quantitative parameters are usually set by hand instead of estimated from the data using standard statistical estimation methods

A detailed introduction to the framework will be available soon in Brasoveanu and Dotlačil (2018, in prep.). Today, a case study:

• the lexical decision task in Murray and Forster (2004)
• we model their data with 3 different ACT-R models that differ qualitatively / symbolically or quantitatively / subsymbolically
• we fit these models to data and compare the results
Road map for the talk

• we introduce the lexical decision task and the data we want to model

• we discuss a basic Bayesian log-frequency model for this data; this model
  - highlights the imperfect data fit of the log-frequency assumption
  - and introduces the basic structure of a Bayesian model we will need later

• we introduce a series of 3 ACT-R models of a participant completing the lexical decision task and quantitatively compare them

• these lexical access models are particularly simple – the framework can accommodate much more realistic linguistic theories
  - (if there’s time) we demo an incremental left-corner parser & interpreter (using DRT on the semantics side) with visual and motor interfaces
The lexical decision task in Murray and Forster (2004)

- word frequency: one very robust parameter affecting latencies and accuracies in lexical decision tasks (Whaley, 1978)
- frequency effects have been found in many if not all tasks that involve some kind of lexical processing (Forster, 1990; Monsell, 1991)
- specific functional form: lexical access latency can be well approximated as a log-function of frequency (Howes and Solomon 1951)
- Murray and Forster (2004) studied the role of frequency in detail and identified various issues with the log-frequency model
- their data consisted of collected responses and response times in a lexical decision task using words from 16 frequency bands – see table on the next slide
## The lexical decision task in Murray and Forster (2004)

The 16 word-frequency bands (in tokens per 1 million words) investigated in Murray and Forster (2004), Exp. 1:

<table>
<thead>
<tr>
<th>Frequency range</th>
<th>Mean frequency</th>
<th>Latency (ms)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>315–197</td>
<td>242.0</td>
<td>542</td>
<td>97.22</td>
</tr>
<tr>
<td>100–85</td>
<td>92.8</td>
<td>555</td>
<td>95.56</td>
</tr>
<tr>
<td>60–55</td>
<td>57.7</td>
<td>566</td>
<td>95.56</td>
</tr>
<tr>
<td>42–39</td>
<td>40.5</td>
<td>562</td>
<td>96.3</td>
</tr>
<tr>
<td>32–30</td>
<td>30.6</td>
<td>570</td>
<td>96.11</td>
</tr>
<tr>
<td>24–23</td>
<td>23.4</td>
<td>569</td>
<td>94.26</td>
</tr>
<tr>
<td>19</td>
<td>19.0</td>
<td>577</td>
<td>95</td>
</tr>
<tr>
<td>16</td>
<td>16.0</td>
<td>587</td>
<td>92.41</td>
</tr>
<tr>
<td>14–13</td>
<td>13.4</td>
<td>592</td>
<td>91.67</td>
</tr>
<tr>
<td>12–11</td>
<td>11.5</td>
<td>605</td>
<td>93.52</td>
</tr>
<tr>
<td>10</td>
<td>10.0</td>
<td>603</td>
<td>91.85</td>
</tr>
<tr>
<td>9</td>
<td>9.0</td>
<td>575</td>
<td>93.52</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>620</td>
<td>91.48</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>607</td>
<td>90.93</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>622</td>
<td>84.44</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>674</td>
<td>74.63</td>
</tr>
</tbody>
</table>
Specifying a Bayesian log-frequency model

To get acquainted with the structure of a Bayesian model, let’s specify a simple Bayesian log-frequency model:

```python
log_freq_model = Model()

with log_freq_model:
    # priors
    intercept = Normal(...)  # priors
    slope = Normal(...)  # priors

    # likelihood
    mu = Deterministic(intercept + slope*np.log(freq), ...)  # likelihood
    observed_rt = Normal(mu=mu, observed=rt, ...)  # likelihood

    # sample posterior
    trace = sample(draws=5000, ...)  # sample posterior
```
The predictions of the log-frequency model

Figure: Log-frequency model estimates and observed RTs

Log frequency model: Observed vs. predicted RTs
Frequency effects as practiced memory retrieval

• log-frequency gets middle values right, but underestimates time needed to access words in extreme frequency bands

• our proposal: frequency effects as practiced memory retrieval
  (different from the proposal in Murray and Forster 2004)

Frequency effects as practiced memory retrieval

- practice: repeated presentation of an item
- ACT-R: retrieval from declarative memory is a power function of time elapsed since item presentation
- the power function is used to compute (base) activation and is based on the number of practice trials / ‘rehearsals’ of a word (1) (free parameters enumerated in parentheses)
- activation of an item is in turn used to compute accuracy (2) and latency (3) for retrieval processes

\[ A_i = \log \left( \sum_{k=1}^{n} t_k^{-d} \right) \]  (d: decay)

\[ P_i = \frac{1}{1 + e^{-\frac{A_i - \tau}{s}}} \]  (s: noise, \(\tau\): threshold)

\[ T_i = Fe^{-fA_i} \]  (F:factor, f: exponent)
Frequency effects as practiced memory retrieval

Figure: Activation, retrieval probability and retrieval latency as a function of time (threshold – dotted black line; 5 presentations – red)
Frequency effects as practiced memory retrieval

- for any word, the number of rehearsals that contribute to its activation are determined by its frequency (we ignore other factors in this model)
- we generate a rehearsal / presentation schedule for a 15-year old speaker based on word frequency and the average number of words the 15-year old speaker is estimated to have seen (estimate based on Hart and Risley 1995)
Bayesian model with ACT-R likelihood for RTs

Embed ACT-R models in Bayesian models to link them to data:

```python
lex_decision_with_bayes = Model()

with lex_decision_with_bayes:
    # priors for model parameters
    d = ...
    s = ...
    tau = ...
    F = ...
    f = ...

    # likelihood: RTs are based on the ACT-R model
    pyactr_rt = actrmodel_latency(F, f, d, activation_from_time)
    rt_observed = Normal(mu=pyactr_rt, observed=RT, ...)
    prob_observed = ...
```
Bayesian model with ACT-R likelihood for RTs

- **pyactr_rt** on line 10 invokes an ACT-R model (we’ll discuss these models presently), and runs it to generate lexical latencies for words in the 16 frequency bands.
- The ACT-R model is parametrized by a latency factor $F$, a latency exponent $f$, a decay $d$ and the activation for words in the 16 frequency bands $\text{activation_from_time}$, computed based on their 15-year long rehearsal schedule.
- The 16 reaction time (RT) means from Murray and Forster (2004) are then assumed to be noisy realizations of the ACT-R generated RTs (line 11).
- For simplicity, we model the observed response accuracies directly (line 12), not via an ACT-R model.
ACT-R models

ACT-R models embed competence theories in processing models.

• we have a qualitative/symbolic competence theory of the lexicon: lexical items have various features (their form etc.)
• we have a qualitative performance theory of what human participants actually do in a lexical decision task
  • lexical items are stored in declarative memory and have an activation that is a function of their frequency
  • participants read a form (sequence of characters) on the screen and attempt to retrieve a word with that form
• the qualitative components are implemented in ACT-R as condition-action pairs (production rules) stored in procedural memory
• these rules trigger a cognitive action if the cognitive context / mental state satisfies a range of conditions
Quantitative comparison for qualitative theories

Generative theories + ACT-R + Bayes enable us to do quantitative comparison for qualitative theories:

- we can implement different competence + processing models in ACT-R, and then embed these alternative ACT-R models in a Bayesian model
- we can then estimate their subsymbolic parameters and quantitatively compare these different models
- model comparison with Bayes factors can apply across the board for any kind of hybrid (quantitative & qualitative) model
  (if done responsibly ... Kass and Raftery 1995)
The model consists of 4 central rules:

1. The "**attend word**" rule takes a visual location encoded in the visual location buffer, a.k.a., the visual *where* buffer, and issues a command to the visual *what* buffer to move attention to that visual location.
An ACT-R model of lexical decision (Model 1)

```
lex_decision.productionstring(name="attend word", string=""
  =g>
  state  attend
  =visual_location>
  isa  _visuallocation
  ?visual>
  state  free
  ==> 
  =g>
  state  retrieving
  +visual>
  cmd  move_attention
  screen_pos =visual_location
  ~visual_location>
  ""
"
```
2. The "retrieving" rule takes the visual value/content discovered at that visual location, which is a potential word form, and places a declarative memory request to retrieve a word with that form;

```
lex_decision.productionstring(name="retrieving", string=""
  =g>
  state retrieving
  =visual>
  value =val
  ==> 
  =g>
  state retrieval_done
  +retrieval>
  isa word
  form =val
"""
)```
3. and 4. The "lexeme retrieved" and "no lexeme found" rules take care of the two possible outcomes of the memory retrieval request
- if a word with that form is retrieved from memory ("lexeme retrieved"), a command is issued to the motor module to press the 'J' key
- if no word is retrieved ("no lexeme found"), a command is issued to the motor module to press the 'F' key
An ACT-R model of lexical decision (Model 1)

```plaintext
lex_decision.productionstring(name="lexeme retrieved", string="""

  =g>
  state retrieval_done
  ?retrieval>
  buffer full
  state free
  =>
  =g>
  state done
  +manual>
  cmd press_key
  key J
  """
)
An ACT-R model of lexical decision (Model 1)

```plaintext
lex_decision.productionstring(name="no lexeme found", string="""
    =g>
    state retrieval_done
    ?retrieval>
    buffer empty
    state error
    ==> 
    =g>
    state done
    +manual>
    cmd press_key
    key F
    """
)
An ACT-R model of lexical decision (Model 1)

Running this model, we obtain an output detailing the cognitive process and its temporal trace:

```plaintext
****Environment: {1: {'text': 'elephant', 'position': (320, 180)}}
(0, 'PROCEDURAL', 'RULE SELECTED: attend word')
(0.05, 'PROCEDURAL', 'RULE FIRED: attend word')
(0.0679, 'PROCEDURAL', 'RULE SELECTED: retrieving')
(0.1179, 'PROCEDURAL', 'RULE FIRED: retrieving')
(0.1179, 'retrieval', 'START RETRIEVAL')
(0.1679, 'retrieval', 'RETRIEVED: word(form= elephant)')
(0.1679, 'PROCEDURAL', 'RULE SELECTED: lexeme retrieved')
(0.2179, 'PROCEDURAL', 'RULE FIRED: lexeme retrieved')
(0.2179, 'manual', 'COMMAND: press_key')
(0.4679, 'manual', 'PREPARATION COMPLETE')
(0.5179, 'manual', 'INITIATION COMPLETE')
(0.6179, 'manual', 'KEY PRESSED: J')
```
ACT-R Model 1: fit to data

Figure: Model 1: estimated and observed RTs and probabilities
ACT-R Model 1: fit to data and qualitative limitations

- the plots show Model 1 has a very good fit, both for latency and accuracy
- but Model 1 oversimplifies the process of encoding visually retrieved data
  - it assumes the visual value found at a particular visual location is immediately shuttled to the retrieval buffer
  - but cognition in ACT-R is goal-driven: any important step in a cognitive process should involve the goal or imaginal buffer
    - the imaginal buffer is a goal-like buffer that stores internal ‘snapshots’ of the cognitive state
- the transfer between the visual and the retrieval buffer should be mediated by the imaginal buffer
ACT-R Model 2: adding the \textit{imaginal} buffer

- Bayesian model remains the same, the only part we change is the ACT-R-provided likelihood for latencies
- we modify the procedural core of the ACT-R model
  - we add the imaginal buffer to the model
  - we replace the "\texttt{attend word}" and "\texttt{retrieving}" rules with three rules "\texttt{attend word}'s", "\texttt{encoding word}'s" and "\texttt{retrieving}'s"
  - the new rule "\texttt{encoding word}'s" mediates between "\texttt{attend word}'s" and "\texttt{retrieving}'s"
  - \texttt{encoding} a word form means taking it from the visual buffer and shuttling it to the imaginal buffer
ACT-R Model 2: adding the **imaginal** buffer

```python
lex_decision.set_goal("imaginal")

lex_decision.productionstring(name="attend word", string="'"

    =g>
    state attend
    =visual_location>
    isa _visuallocation
    ?visual>
    state free
    ==> 
    =g>
    state encoding [the only change in this rule]
    +visual>
    cmd move_attention
    screen_pos =visual_location
    ~visual_location>
    ""
```
ACT-R Model 2: adding the **imaginal** buffer

```python
lex_decision.productionstring(name="encoding word", string="""
  =g>
  state  encoding
  =visual>
  value  =val
  ==>  
  =g>
  state  retrieving
  +imaginal>
  isa  word
  form  =val
  """
)```
ACT-R Model 2: adding the **imaginal** buffer

```
lex_decision.productionstring(name="retrieving", string="""
  =g>
state retrieving
=imaginal>  [imaginal instead of visual: the only change in this rule]
isa word
form =val
==> =g>
state retrieval_done
+retrieval>
is word
form =val
"")
```
ACT-R Model 2: adding the **imaginal** buffer

- these modifications are symbolic/discrete/non-quantitative modifications
- but we are able to fit the new model to the same data and quantitatively compare its performance with Model 1 (the no-imaginal-buffer model)
- the left plot on the next slide shows that Model 2 has a very poor fit to the latency data
ACT-R Model 2: fit to data

Figure: Model 2: estimated and observed RTs and probabilities
ACT-R Model 2: adding the **imaginal** buffer

- the encoding step adds **200 ms** to every lexical decision simulation
- **200 ms** is the default ACT-R delay for chunk-encoding into the imaginal buffer
- the predicted latencies for 15 out of the 16 word-frequency bands are greatly overestimated (above the diagonal line)
- Model 2 cannot run faster than about **640 ms**; this is too high to fit high-frequency words, which take about **100 ms** less than that
ACT-R Model 3: **imaginal** buffer with 0 delay

- let’s change a quantitative feature of Model 2 and set the imaginal delay to 0 ms (instead of its default 200 ms value)

```python
1 lex_decision.goals["imaginal"].delay = 0
```

- it is reasonable to assume that various default values for ACT-R subsymbolic parameters should be changed when modeling linguistic phenomena
- natural language comprehension involves fast incremental construction of rich hierarchical representations
- this richness significantly exceeds the complexity of representations needed for other high-level cognitive processes modeled in ACT-R (e.g., arithmetic)
- Model 3 fits very well the mean latencies for all the 16 word-frequency bands
ACT-R Model 3: fit to data

Figure: Model 3: estimated and observed RTs and probabilities
Conclusion

- we have a formally explicit way to connect competence-level theories to experimental data via explicit processing models
- we can formally, explicitly connect qualitative/symbolic/competence-level theory construction (the main business of the generative grammarian) and quantitative/subsymbolic/performance-level data collection and prediction (the main business of the experimental linguist)

For a future occasion – more systematic / formal model comparison:

- we have only done informal quantitative comparisons based on posterior predictions
- but systematic across-the-board model comparison via Bayes factors is possible in this framework
Demo time
An incremental left-corner parser & interpreter (using DRT on the semantics side) with visual and motor interfaces

... applied to cataphora, specifically the conditional:

(4) John won’t eat it if a hamburger is overcooked. (Elbourne 2009, p. 3)

The model provides an end-to-end simulation of a human participant in a self-paced reading task (Just et al. 1982):

• it reads the conditional in (4), which is displayed one word at a time on a virtual screen

• it presses the space bar to move to the next word when the current word is integrated (parsed & interpreted)

• it implements a version of Discourse Representation Theory (DRT; Kamp 1981, Kamp and Reyle 1993) on the semantics side

• it builds the expected tree structures on the syntax side
We are grateful to Donka Farkas, Abel Rodriguez, Matt Wagers and the UCSC S-lab audience (January 2018) for comments and discussion. The usual disclaimers apply.


