easyNet

Worksheet B

Exercises on the Dual-Route Cascaded Model (DRC)

Exercise B.1: Steps on the nonlexical route

We know that the nonlexical route of the DRC can read pseudowords in a plausible way and give potentially correct pronunciations of words. Now we're going to look at the workings of the nonlexical route directly. Start a new copy of the DRC. Our goal is to see each of the stages in nonlexical processing. We start with looking at the input that the nonlexical route receives.

Turn on graphing for the **Letters** (state → layers/by_time_grouped) layer that feeds the nonlexical route and **Most Active Letters** (state → layers/by_time), the first layer on the nonlexical route**.**

Switch to the **Word_Naming** trial if this was not selected by default.

Run **hunt** as a stimulus. Compare the two graphs. Notice, the left-to-right inclusion of letters on the nonlexical route. This contrasts with the lexical route, where all the letters are fed in parallel on the lexical route to the orthographic lexicon. This left-to-right inclusion of the letters is the first cause of the similar staging of phonemes arriving in sequence to the Output Buffer.

Run **aave** as a stimulus. In the letters plot, notice that top-down feedback activates the H/0 from the word *have*. However, only the most active /0 letter, viz. A/0, is ever fed to the nonlexical route.

Activate plotting for the **GPC Graphemes** (state → layers/wtstring_layer2) and **GPC Phonemes** (state → layers/by_time) layers, and trash the letters plot.

Re-run **hunt**.

Notice that the plot of graphemes is quite different from the other layers. This is not only a different graph, but the underlying representation is of strings that are tagged with weights, rather than nodes with particular activities. The activities that are in the most-active-letters layer were turned into a string so that individual grapheme detection rules could be applied across multiple positions without duplication (as in the original DRC).

All of the graphemes in 'hunt' are single-letter graphemes, identified when the relevant letter is received. You will see that in the GPC phonemes layer that these convert to the expected phonemes (*V* is the short u (/ʌ/) sound).

With thanks to:

The Leverhulme Trust

easyNet

Worksheet B

Exercise B.2: The complexities of parsing graphemes

Note, in **hunt**, though, that the 'u' grapheme has a marker [bm] that indicates that the grapheme has been marked as being in either a beginning or middle position, rather than the end. Let's continue to examine the graphemic parsing process; for now, turn off the GPC Phonemes graph.

The 'u' grapheme has a different regular pronunciation when it appears at the end of the stimulus. Run **flu** as a stimulus. Observe that 'u' is interpreted as beginning/middle until the blank/terminator '"' symbol is entered into parsing, and then u[e] ([e] is end) is selected.

Run **psops** as a stimulus. Notice that 'ps' is treated as a single grapheme 'ps[b]' at the beginning, but two separate graphemes, 'p' and 's' separately at the end. This indicates that the position affects the identification of the grapheme, rather than the grapheme being identified position-free, and the position affecting its pronunciation.

Run **ace** as a stimulus. Notice the 'a?e' (a-something-e or 'a-magic-e') grapheme. See that the first letter is reparsed once the third letter of the grapheme is received on the nonlexical route. That is, two more letters are received before the ultimate parse of 'a' is reach (as part of a?e). One of the parses of the c is 'c(e)[bm]'. The '(e)' indicates a contextual element that could be available for further parsing; this can more clearly be seen with the pseudoword **sced**.

Run **mave** as a stimulus. Why is the "m" grapheme so much less active than the others? Confirm your interpretation with a well-aimed lesion.

Run **itsch** as a stimulus. When is the parsing of 't' complete (how long does it take)?

Exercise B.3: The consequences of changing parses: whammies

Now we'll look at the phonemes that come from these changing graphemic parses. So, re-activate the **GPC Phonemes** graph, and also graph the **Output Buffer**.

Run **shed** as a stimulus. Notice the change from 's' to 'sh' as the letter 'h' comes in. Look at the consequences at the phoneme layer: the 's' is replaced by 'S' (/ʃ/). Notice that S/0 is not, however, the phoneme that controls the RT.

Compare with **sped**. Note that s/0 is also not the phoneme that controls the RT. There is no whammy cost for this early digraph, because it is resolved too soon.

Also look at **crash**. What happens for late digraphs in words?

Doaph is a stimulus from Rastle & Coltheart (1998). What happens here?

With thanks to:

The Leverhulme Trust