Baayen & al. 1997, *JML*
Singulars and plurals in Dutch: evidence for a parallel dual-route model

(1) Storage vs. computation

- When you have to carry out some computation often, maybe you should give yourself a shortcut by just storing the result.
- The longer the computation, the more valuable the shortcut
- Cite findings for reaching movements, arithmetic

(2) Competing models for morphologically complex words

- Pinker & Prince-type model (dual mechanism) says you don't store regularly inflected words no matter how frequent
  - hard to explain frequency effects for regulars
- Butterworth (full listing) says you list all whole words, regular or not
  - hard to explain base-frequency effects (unless they come from concept nodes?)
- Caramazza, Burani, Laudanna (Augmented Addressed Morphology): full listing (direct route), but with parsing route as backup for rare or new words
  - 1980s version is "cascading" dual route: parsing route kicks in only if whole-word route fails
  - 1990s versions suggest that parsing route could also be used if whole word has low frequency but parts have high frequency

(3) Model adopted here: Schreuder & Baayen 1995

- Dual-route race model
- Parsing route requires activating access reps of parts, then activating their lemmas, then checking subcategorization features, then computing meaning of whole word
- Activation flows back down from syntactic/semantic layer to access representations of whole word and constituents
  - → family size and the like can only matter for transparent words
- "Over time, activation feedback tunes the system such that an advantage for the parsing route results for transparent routes, but a disadvantage results for semantically opaque words." (p. 97)

(4) Exp. 1: singular- vs. plural-dominant words

- E.g., *eyes* > *eye*, but (I'm guessing) *steak* > *steaks*
- Schematic predictions of different models

only freq. matters (no parsing)

freq. doesn't matter; but all plurals take a bit longer because you have to parse

mostly just freq. matters, but low-freq. plurals might require parsing --> even slower

sg. is substring of pl. -> sg. gets activation every time pl. accessed -> sg. RT depends on summed freq. of sg. & pl.
pl. whole-word rep. doesn't get activation when sg. is accessed -> only freq of pl. affects its RT

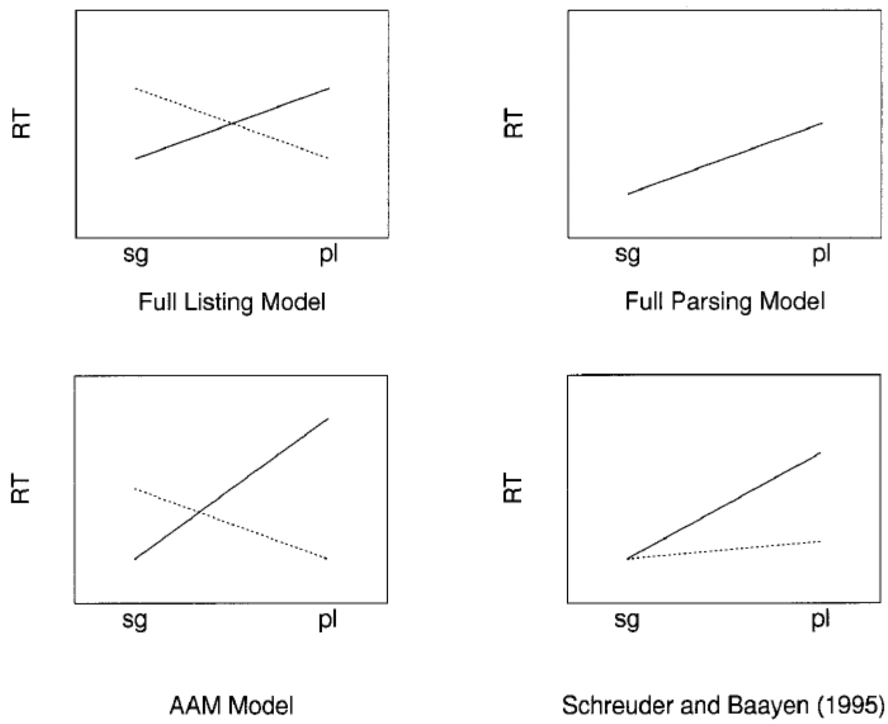
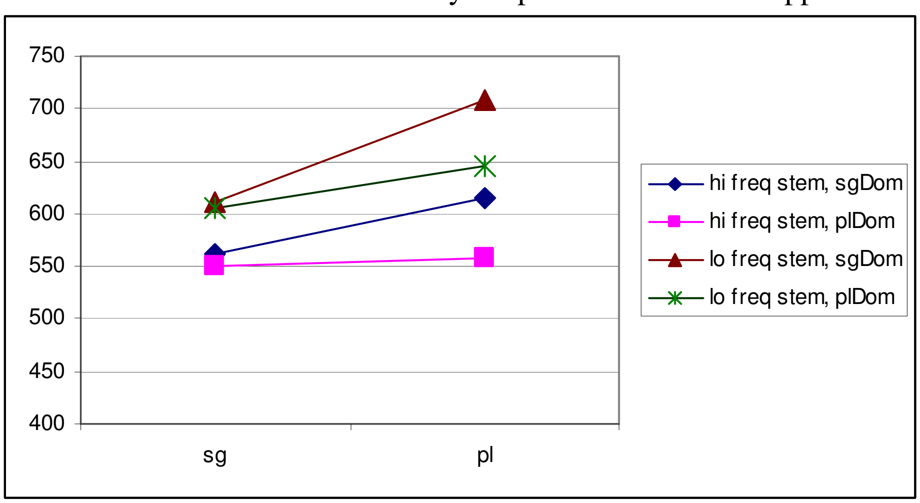


FIG. 1. Predicted pattern of reaction times (RT) for singulars (sg) and plurals (pl) for four models of morphological processing. Dotted lines represent plural-dominant pairs, and solid lines represent singular-dominant pairs.

(p. 98)

(5) Exp. 1, cont'd: Dutch plurals

- *-en, -s* (both productive), *-eren*
- This study just uses *–en*: regular, and homonymous with a verb suffix
- Simple visual lexical decision
- Dominance seems to matter only for plural-dominant—supports Schreuder & Baayen model



(6) Modeling Exp. 1 results

- Resting activation a_ω of word ω proportional to log of frequency of all words that contain it $f_{cum,\omega}$.
- Time required to reach threshold $t_\omega = \frac{1}{1+\log f_{cum,\omega}}$.
- So, for singular lexeme, $t_{cat} = \frac{1}{1+\log(f_{cat}+f_{cats})}$
  - Reaction time should be just $t_{cat} + \varepsilon$, where ε is how long it takes to do the initial mapping to access rep. and also execute response
- But for full-form lexeme of plural, $t_{cats} = \frac{1}{1+\log f_{cats}}$
  - Reaction time should be $t_{cats} + \varepsilon$ or $t_{cat}+\Delta_p$, whichever is shorter (where Δ_p is the time it takes to segment, license, and compose)
- Add some Gaussian noise to all these variables.
- Fit ε and Δ_p to data.
- Result—pretty good match. Parsing route is so expensive (big Δ_p) that it wins only for lowest-freq. plurals.

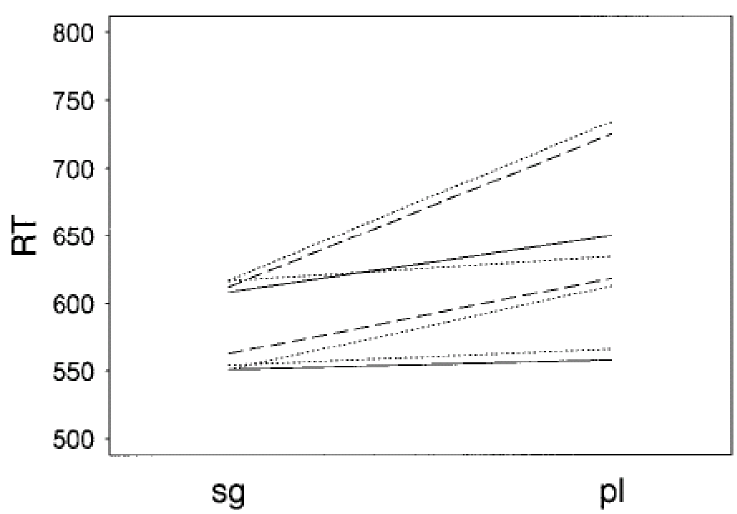


FIG. 2. Modeling results for Experiment 1. Observed reaction times of plural-dominant singulars (sg) and plurals (pl) are represented by solid lines, and observed reaction times of singular-dominant singulars and plurals are represented by dashed lines. The reaction times generated by the model are plotted with dotted lines. (p. 101)

(7) Exp. 2: same idea but hold sg. freq constant and vary pl. freq.

- So sg.-dominant, neutral, pl.-dominant
- Result: taking parameter values from Exp. 1 (but estimating standard deviations for noise anew), qualitatively decent fit.

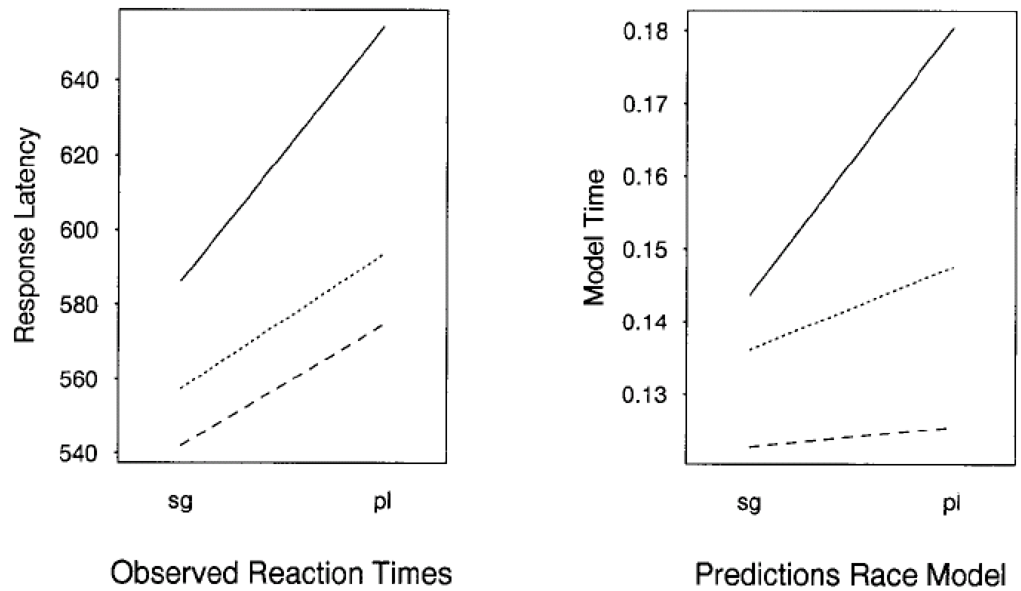


FIG. 3. Observed (left) and predicted (right) patterns of reaction times for singulars (sg) and plurals (pl) in Experiment 2. Solid lines denote singular-dominant nouns, dashed lines denote plural-dominant nouns, and dotted lines denote dominance-neutral nouns.

(p. 107)

(8) Why is parsing so costly?

- *-en* is also a verb suffix (infinitive, plural subject, participle with *ge-*)
- Occurs more often on verbs (64% of CELEX tokens) than on nouns (36%).
- Assume a single access rep. for *–en*, linked to two (or more?) different lemmas.
  - verbal *–en* lemma gets activated before nominal *–en* lemma
  - then a noun base's lemma gets activated
  - subcategorization conflict!
  - deactivate verbal *–en*'s lemma
  - eventually nominal *–en* becomes available and subcategorization succeeds
- → delay in the subcategorization-checking stage of parsing for *–en* nouns.

(9) Exp. 3: will parsing be cheaper for verbal *–en*?

- Unfortunately, didn't quite copy the design of Exp. 1 to estimate Δ_p
- Instead, just compare the "*-en* disadvantage" in nouns and verbs—it is indeed much bigger in nouns:

TABLE 3

MEAN LATENCIES (IN MILLISECONDS) AND PERCENTAGES OF ERRORS FOR -EN NOUN AND VERB SINGULARS AND PLURALS IN EXPERIMENT 3

Word category	Singular	Plural
Noun	545 (2)	611 (6)
Verb	603 (7)	612 (10)

(p. 109)

(10) What this means for us

"Together, our results suggest that many noun plurals are stored in order to avoid the time-costly resolution of the subcategorization conflict that arises when the *–en* suffix is attached to nouns." (p. 94)

- If we adopt this model, we make predictions about which complex words will prefer the whole-word route and which the parsing route...
- ...but this depends on Δ_p, which is affected by homonymy
  - The less-frequent version of a homonymous affix will be costly to parse, and produce more storage
- Would we expect anything similar in production?