

# Instructions for Running Über-Reader 1B Simulations

© Erik D. Reichle (5 September 2025)

## 1. Introduction

This document provides an extremely abbreviated set of instructions for running simulations using the *Über-Reader* model of reading (Reichle, 2021, *Computational Models of Reading: A Handbook*, Oxford University Press) and its two core components, the *Multiple-Trace Memory model (MTMM)* of word identification and the *Pro-Parser* model of sentence processing. Because the *E-Z Reader* model of eye-movement control in reading (Reichle et al., 2009, 2012) is the basic framework in which the MTMM and Pro-Parser are embedded to produce *Über-Reader*, the code for E-Z Reader and *Über-Reader* are very similar. For that reason, I recommend that you download and become familiar with E-Z Reader before running simulations using *Über-Reader* or its components.

The version of *Über-Reader* that I'm sharing is a reimplemented version of the model described in Chapter 7 of Reichle (2021). The model was reimplemented to make running simulations more efficient and to make the code more transparent, as described below in the Appendix. If you haven't already done so, review the description of the model provided by Reichle (2021) because it will make understanding the code and the task of running simulations much easier. In the sections that follow, I provide very brief instructions for setting up and running simulations using the MTMM, Pro-Parser, and *Über-Reader* models. Please let me know if you have any difficulties and I'll obviously do my best to assist. Also note that I intend to write more detailed instructions (similar to those for E-Z Reader) and some point in the near future, when time permits.

## 2. MTMM Simulations

This model was used to complete Simulation 1 as reported by both Reichle (2021) and the Appendix, below.

The main class, *MTMM.java*, defines variables that include model parameters, toggles to display various types of simulation output, and other information required to run simulations. These variables are labeled and fairly well commented in the code, so their roles should be fairly easy to discern. However, here are a few key things to keep in mind:

*corpusFileName* contains all of the words/sentences used in running simulations. For example, "Corpus-ELP-40411.txt" is a .txt file containing the 40,411 words from the ELP project (Balota et al., 2007).

*lexiconFileName* contains all of the lexical information required to populate the

model's lexicon. For example, "Lexicon-ELP-40411.txt" contains all of the lexical information for the 40,411 words of the English Lexicon Project (Balota et al., 2007).

*offset* is a particularly important variable because it specifies where the words will be fixated during a simulation. For example, setting the value of *offset* equal to 7 will cause the fixation to be seven character spaces to the left of the beginning of each word.

*NWords* specifies the number of words in the model's lexicon. The value of this variable therefore equal to the number of words contained in *lexiconInputFile*.

Note that running simulations like Simulation 1 (see Appendix) are slow. For example, using a 2.3 GHz Intel Xeon W processor, each simulated subject required slightly more than 28 minutes to identify each of the 40,411 words in *corpusInputFile*. Because of this, my advice is to set *NSubjects* to a small value (e.g., = 5) the first time you run a simulation using MTMM.

### **3. Pro-Parser Simulations**

This model was used to complete Simulation 2 as reported by Reichle (2021) and the Appendix, below.

The main class, *ProParser.java*, contains several variables that can be specified, with the key one being *corpusFileName*. This file contains the corpus of sentences that are to be parsed. For example, "Corpus-SRC98.txt" contains the 48 sentences of the Schilling et al. (1998) corpus. Because this model is not stochastic, the output of each simulation will be identical.

### **4. Uber-Reader Simulations**

This model was used to complete Simulations 3 and 4 as reported by Reichle (2021) and the Appendix, below.

The main class, *UberReader.java*, contains a large number of variables including model parameters, toggles to display various types of simulation output, and other information required to run simulations. These variables are labeled and the code is well commented, so the variables' roles should be easy to discern. However, here are a few key things to keep in mind:

The various input files are the same as those used with the MTMM (see above), as are many of the parameters and simulation toggles. Despite of the fact that this implementation of Über-Reader is about 78% faster than the version described by Reichle (2021), the simulations still require a lot of time to complete (see Footnote 1 of Appendix, below). For that reason, you're advised to set *NSubjects* to a small value (e.g., = 5) when running your first simulation.

Also, as per E-Z Reader, a few of the toggles that are used to print simulation output to a file (e.g., *displayStates*) will generate a huge amount of information, so use a small value of *NSubjects* when exploring these different options. In my experience, the most useful output is provided by *displayFrequencyClassMeans* and *displayMeanDistributions*. Also, *displayCorpus* can be extremely useful for checking that any sentence corpus (as provided by the file “corpusFileName.txt”) has been formatted correctly prior to running a simulation.

Again, I understand that these instructions are extremely abbreviated and for that I apologize. I will, however, continue to refine them as time permits.

Good luck!

## Appendix: Replication of Simulations 1-4 (Reichle, 2021)

These simulations were completed using a reimplemented version of the *Über-Reader* model of reading (Reichle, 2021) and its two key components, the *Multiple-Trace Memory model (MTMM)* of word identification and the *Pro-Parser* production. This reimplemented version of *Über-Reader* was intended to: (1) speed up the running of simulations and (2) make the code more transparent and thus easier to understand and modify. As such, these updated *Über-Reader*, *MTMM*, and *Pro-Parser* models have been designated as versions 1B. To guarantee that the previous and current implementations are functionally equivalent, I've replicated Simulations 1-4 from Reichle (2021), as reported below. Finally, to benchmark how much faster the current version of *Über-Reader* is relative to its predecessor, the latter required approximately 165 seconds for each simulated subject to read the 48 sentences from the Schilling et al. (1998) corpus, whereas the current model required only 37 seconds<sup>1</sup>. This represents an approximate 78% reduction in the time to run simulations – a non-trivial savings.

Each of the simulations reported below were completed using the same parameter values as reported in Table 7.5 of Reichle (2021), but with the exception of Simulation 2, also used more simulated subjects to ensure that the results are indeed reliable.

### Simulation 1

This simulation examined the *MTMM*'s capacity to identify 40,411 1-22 letter words from the English Lexicon Project corpus (Balota et al., 2007). The simulation was completed using  $N = 20$  simulated subjects. Each statistical subject required approximately 28 minutes to complete using a 2.3 GHz Intel Xeon W processor. Note that, although the simulation replicates Simulation 1 from Reichle (2021), the new results are a bit less erratic than the latter because there were four times as many simulated subjects.

Table 1 shows the model's overall performance in recalling orthographic, phonological, semantic, and syntactic information as a function of the fixation position, as well as two indices of the time required to identify words: the initial stage of activated word-specific memory traces,  $t(\text{activate})$ , and the time required to fully identify the word. The results shown in Table 1 are almost identical to those shown in Table 7.6 of Reichle (2021).

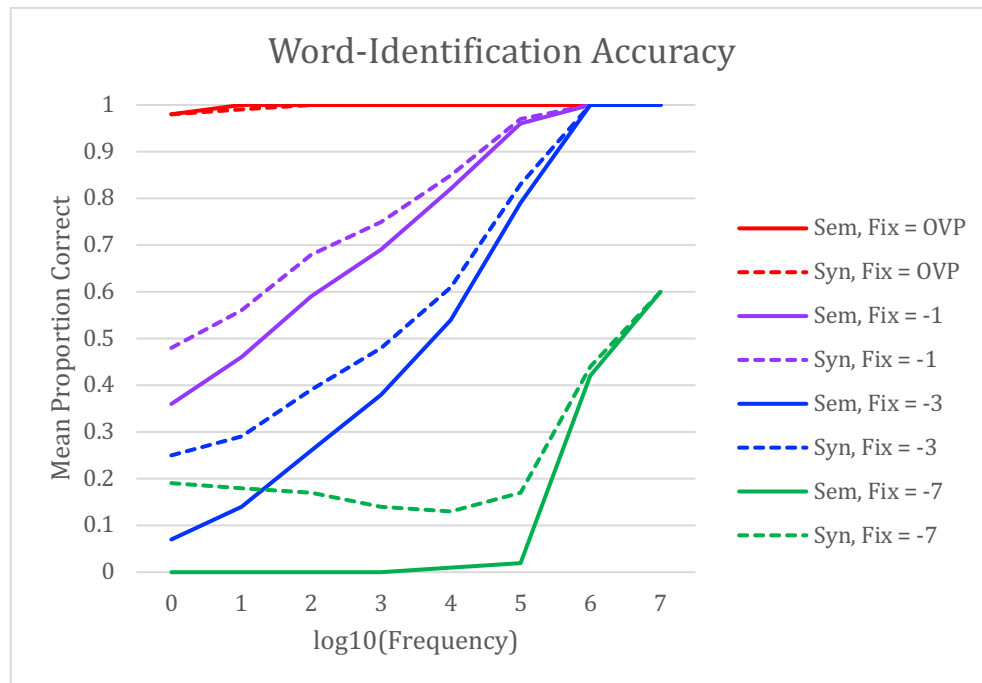
---

<sup>1</sup> These simulations were completed using the same 2.3 GHz Intel Xeon W processor.

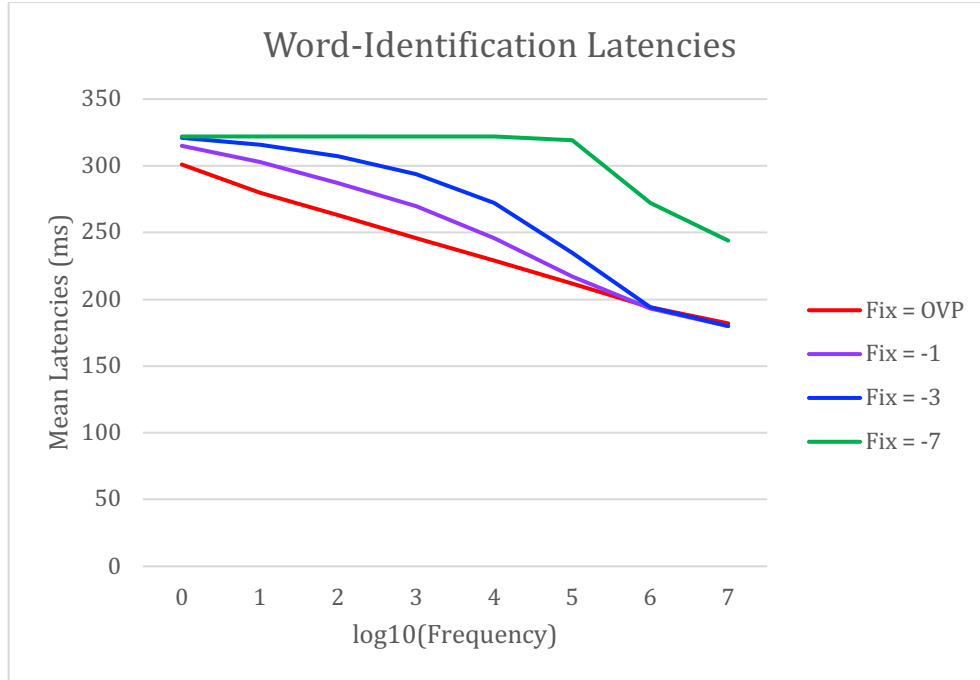
*Table 1.* Simulated mean word-identification accuracies and latencies as a function of fixation position.

Fixation Position	Recall Accuracy				Word Identification Latencies (ms)	
	Ortho	Phono	Sem	Syn	t(activate)	RT
OVP	1.00	0.96	1.00	1.00	173	260
-1	0.61	0.60	0.61	0.68	196	283
-3	0.29	0.29	0.29	0.41	215	302
-7	0	0	0	0.16	235	322

The mean word-identification accuracies and latencies are also shown as a function of eight  $\log_{10}$  frequency classes in Figures 1 and 2, below. These results are again nearly identical to those shown in Figure 7.6 of Reichle (2021) but are slightly less erratic due to the four-fold number of simulated subjects.



*Figure 1.* Simulated word identification accuracies as a function of word-frequency class and fixation position. *Notes:* “Sem” = semantic information; “Syn” = syntactic information; “Fix = OVP” = fixation position is optimal viewing position; “Fix = -1” = fixation position is 1 character space to the left of the word beginning; “Fix -3” = fixation position is 3 character spaces to the left of the word beginning; “Fix = -7” = fixation position is 7 character spaces to the left of the word beginning.



*Figure 2.* Simulated word identification latencies (ms) as a function of word-frequency class and fixation position. *Notes:* “Sem” = semantic information; “Syn” = syntactic information; “Fix = OVP” = fixation position is optimal viewing position; “Fix = -1” = fixation position is 1 character space to the left of the word beginning; “Fix -3” fixation position is 3 character spaces to the left of the word beginning; “Fix = -7” = fixation position is 7 character spaces to the left of the word beginning.

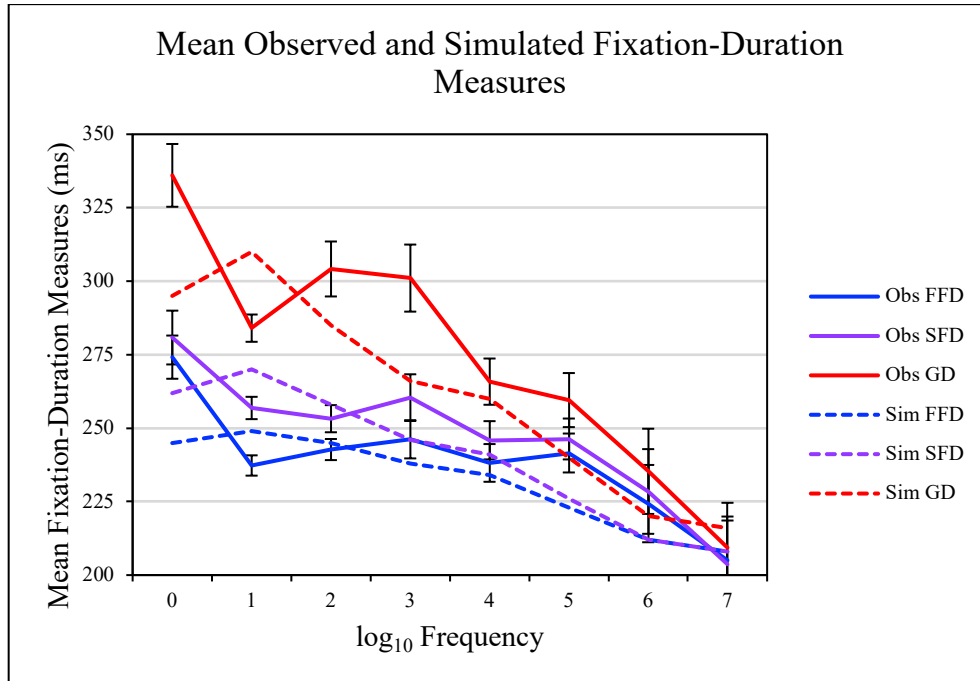
## Simulation 2

This simulation was completed using  $N = 1$  simulated subject because the Pro-Parser model is not stochastic. The results exactly replicated those reported by Reichle (2021), with the model giving plausible parses for 32 of 48 (75%) of the Schilling et al. (1998) sentences (see Table 7.7 of Reichle, 2021).

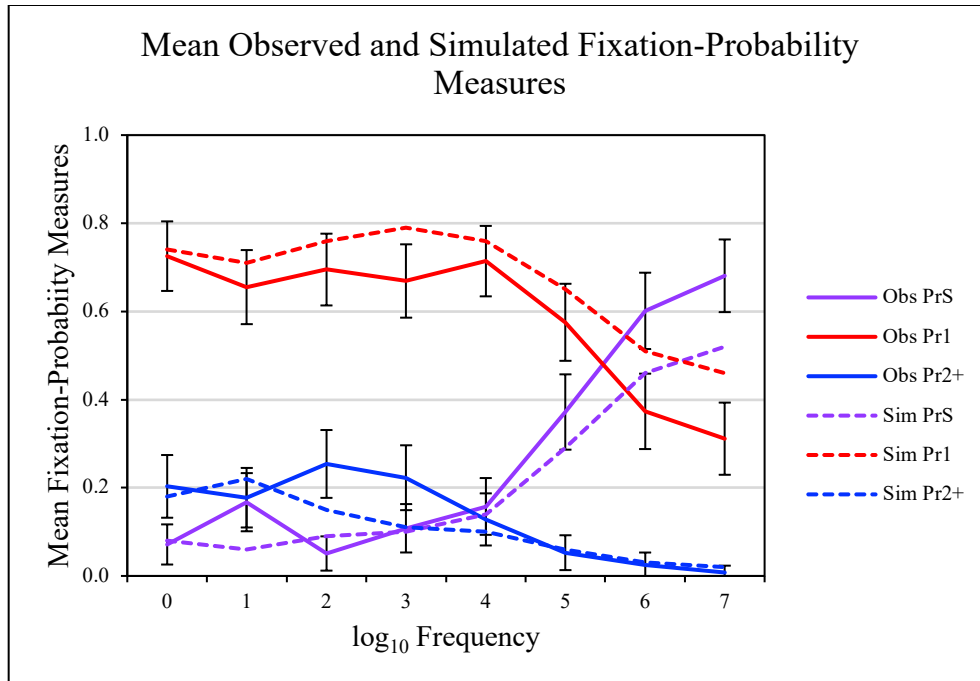
## Simulation 3

This simulation was completed using Über-Reader to simulate the reading and recall of the 48 sentences from Schilling et al. (1998). As such, the simulation generates predictions about various eye-movement measures and the recall of propositions from the individual sentences. The simulation was completed using  $N = 1,000$  simulated subjects, or ten times the number used by Reichle (2021). The RMSD for the simulation was equal to 0.33. This value is slightly larger than the value of 0.29 reported by Reichle (2021), probably because the grid-search procedure that was used to estimate optimal parameter values selected for RMSD

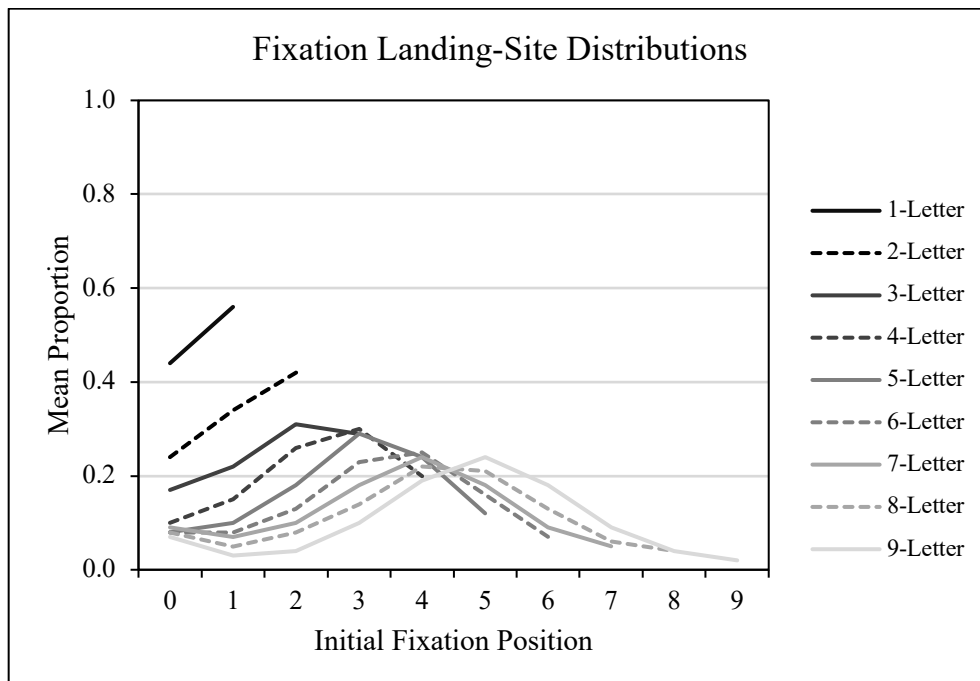
that were estimated to be smaller than they actually were due to chance. Figures 3 and 4 below shows the observed and simulated fixation-duration and fixation-probability measures (cf., Reichle, 2021, Figure 7.7). Figures 5-7 below respectively show the simulated fixation landing-site distributions, refixation-probability distributions, and IOVP effects (cf., Reichle, 2021, Figure 7.8). All of the results reported below are very similar to those reported by Reichle (2021), indicating again that the new implementation of the model is functionally equivalent to the previous implementation.



*Figure 3.* Observed and simulated fixation-duration measures as a function of word-frequency class. *Notes:* “Obs” = observed; “Sim” = simulated; “FFD” = first-fixation duration; “SFD” = single-fixation duration; “GD” = gaze duration.



*Figure 4.* Observed and simulated fixation-probability measures as a function of word-frequency class. *Notes:* “Obs” = observed; “Sim” = simulated; “PrS” = probability of skipping; “Pr1” = probability of making single fixation; “Pr2” = probability of making two or more fixations.



*Figure 5.* Simulated fixation landing-site distributions for 1-9 letter words. *Note:*



The value of 0 on the x-axis represents the blank space immediately to the left of a given word.

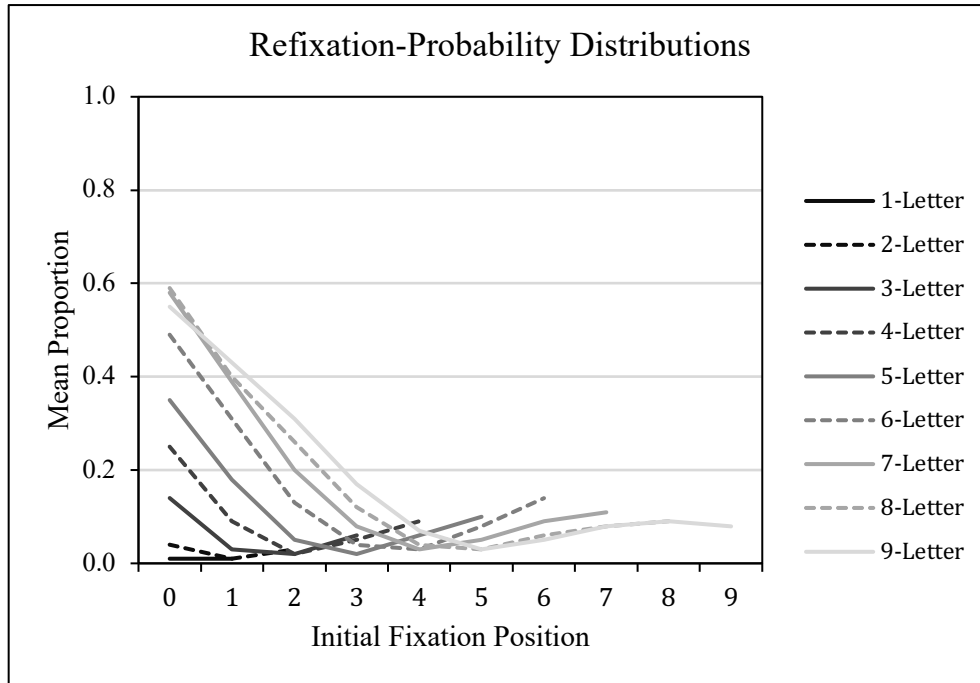


Figure 6. Simulated refixation-probability distributions for 1-9 letter words. *Note:* The value of 0 on the x-axis represents the blank space immediately to the left of a given word.

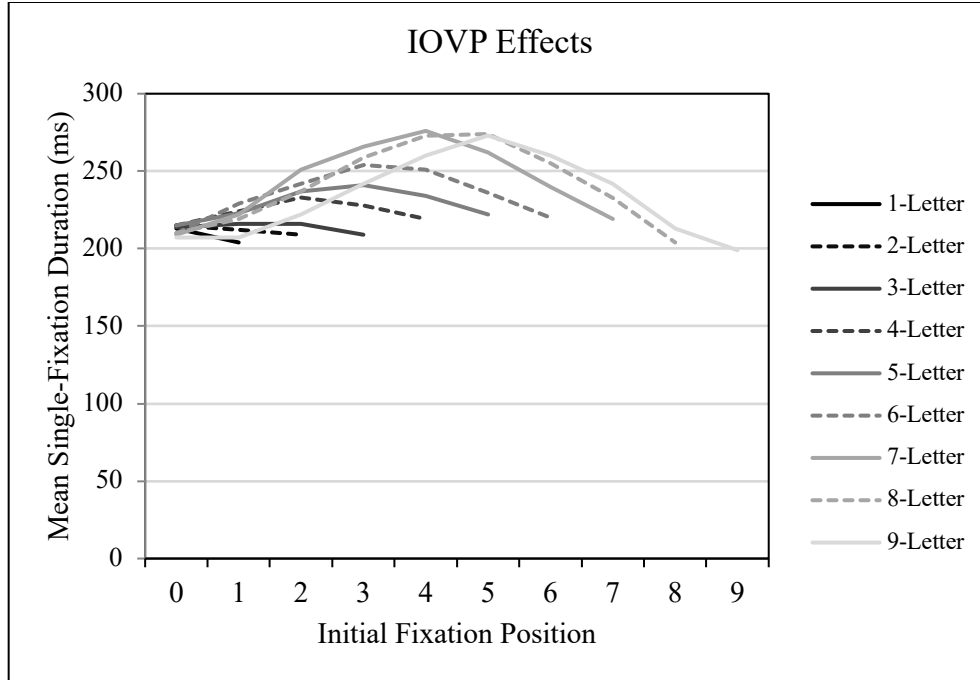


Figure 7. Simulation inverted optimal viewing position (*IOVP*) effects for 1-9 letter words. *Note:* The value of 0 on the x-axis represents the blank space immediately to the left of a given word.

Finally, sentence recall was scored using three different metrics: (1) the correlation,  $r$ , between the pattern of semantic features that correspond to a given sentence and the pattern of features that are recalled; (2) the proportion of a sentence’s semantic features that are recalled; and (3) the proportion of semantic features recalled that were not part of the actual sentence. The mean values of these three metrics are shown in Table 2 for both the current simulation and the simulation reported by Reichle (2021). Comparison of the two sets of values indicates that they are quite similar except for the current model predicts a smaller proportion of incorrectly recalled features.

Table 2. Simulated sentence recall using three metrics for Simulation 3 and Reichle’s (2021) Simulation 3.

Recall Metric	Simulation 3	Reichle’s (2021) Simulation 3
$r$	0.48	0.56
Proportion of Recalled Correct Semantic Features	0.69	0.84
Proportion of Recalled Incorrect Semantic Features	0.09	0.86

## Simulation 4

This simulation was also completed using Über-Reader,  $N = 200$  simulated subjects, or twice the number used by Reichle (2021). The simulation examined the model's capacity to recall the semantic features from the following 3-sentence toy discourse: *The black cat chased the mouse into the corner. It slept after finishing the meal. A typical day in the life of a cat.* The model's performance was evaluated using the same three measures used in Simulation 3<sup>2</sup>. Table 3 shows how the current model compared on these three metrics relative to what was reported by Reichle (2021). Similarly, Table 4 shows one example to illustrate how the sentences of the discourse were parsed, and what was recalled. These latter results are similar to what was reported by Reichle (2021) in Table 7.8 with the except that the new model again predicts the recall of a smaller proportion of incorrect semantic features.

*Table 3.* Simulated sentence recall using three metrics for Simulation 3 and Reichle's (2021) Simulation 3.

Recall Metric	Simulation 3	Reichle's (2021) Simulation 3
$r$	0.25	0.33
Proportion of Recalled Correct Semantic Features	0.61	0.49
Proportion of Recalled Incorrect Semantic Features	0.18	0.51

*Table 4.* Example of the phrase structures and proportional content recalled from the 3-sentence discourse.

Sentence #	Information Type	Example
1	Sentence	The black cat chased the mouse into the corner.
	Phrase Structure	[[The black cat] <sub>NP</sub> [[chased [the mouse [into [the corner.] <sub>NP</sub> ] <sub>PP</sub> ] <sub>NP</sub> ]] <sub>VP</sub> ] <sub>S</sub>
	Propositions	[cat, chased, mouse, into, corner]
2	Sentence	It slept after finishing the meal.
	Phrase Structure	[[It] <sub>NP</sub> [slept [after [.] <sub>NP</sub> ] <sub>PP</sub> [finishing [the meal.] <sub>NP</sub> ] <sub>VP</sub> ]] <sub>S</sub>
	Propositions	[it, slept, after, finishing, meal]
3	Sentence	A typical day in the life of a cat.
	Phrase Structure	[[A day [in [the life [of [a cat.] <sub>NP</sub> ] <sub>PP</sub> ] <sub>NP</sub> ] <sub>PP</sub> ] <sub>NP</sub> ] <sub>S</sub>
	Propositions	[day, in, life, of, cat]

<sup>2</sup> Reichle (2021) reported a fourth recall measure: the proportion of recalled features that were not from a given sentence but were from one of other two sentences in the discourse. This fourth measure was not calculated in this simulation.