RESEARCH · LANGUAGE

A Reading Is a Program over a Sequence

Deriving sentence understanding from meaning pairs, with no language model in the loop

Stefan Ragland

Dominion Labs Research & Development

Dominion Labs

research@dmnlabs.org

January 21, 2025

Abstract

We ask whether a system can *derive* a procedure for reading a sentence — mapping a string of words to its meaning — rather than being handed a hand-written parser or a statistical language model. Treating a reading as a program over a token sequence (a cursor advancing over words, writing to a few registers under branch conditions), we induce the individual read-instructions from a handful of before/after demonstrations, then synthesize the branching procedure that composes them purely from sentence/meaning example pairs. From *five* taught sentences the system derived a six-step procedure and then correctly read *7 of 7 held-out sentences whose every content word was new*, with *zero language-model calls* at any point (measured, not assumed). The synthesizer searched 251,487 candidate procedures; four fit all five examples, and all four agreed on every held-out sentence. A corrupted-supervision control yielded no procedure within the search bound. The result is evidence that sentence understanding at this scope can be obtained as program synthesis from examples, not as language modeling.

1 Reading as program synthesis

The dominant way to map text to meaning is to train a large statistical model on a massive corpus. The alternative examined here treats a single sentence-reading as a short *program*: a left-to-right cursor over tokens that writes a subject, a predicate, and a polarity into registers and emits a structured meaning. The scientific question is whether such a program can be *derived from examples* rather than authored by a human or approximated by a trained network. Program synthesis from input/output examples is well studied [1,2]; se-

mantic parsing learns sentence-to-meaning maps from supervision [3]; and systematic-generalization benchmarks show where statistical sequence models fail to generalize compositionally [4].

A deterministic extractor — a fixed set of patterns — is the system being *read to*, not learning to read: every new construction needs a human to write another pattern. Our premise is to derive the next pattern from data instead.

> *A reading is a program over a sequence. Given word-class as data and a small instruction set demonstrated by example, the map from a sentence to its meaning can be induced as a branching procedure from sentence/meaning pairs alone — and it generalizes to unseen sentences whose content words are entirely new, with no language model in the loop.*

2 The reading machine

A cursor scans the words of a sentence left to right. A few registers hold the emerging meaning: who the sentence is about, what is predicated of it, and whether it affirms or denies. A short, fixed instruction set operates the machine — bind the current word as the subject; bind it as the object; mark the reading negative; advance without recording; and emit the finished meaning — and every instruction advances the cursor. A small closed-class lexicon marks certain words as function words (copulas, determiners, negators) and publishes, for the word under the cursor, only *which class* it is. This class label is treated as data the world provides, exactly like "which block is smaller" in a stacking puzzle.

This is the honest boundary of the result: what is *supplied* is six words classed as copula, determiner, or negator; what is *derived* is which class matters, where, in what order, and what to do about it.

3 Inducing instructions, synthesizing the procedure

For each instruction, a few demonstrations record the machine's observable state before and after the instruction fires; from these, one rule is induced per register the instruction writes (a rule may only conclude about terms its own body binds). This yields executable operators whose effects are learned, not coded. Given the learned operators and the conditions the machine can branch on, a synthesizer then searches for the shortest branch-table procedure ("under condition X, run instruction Y") that reproduces the correct meaning on *every* training pair. No grammar, no parse trace, and no rule about determiners is given — only sentences and what they mean. Because the model-call count is enforced to zero and measured, "no model was used" is a measurement rather than an assurance: a model may propose a situation, but the world supplies the outcome.

4 Results

4.1 Held-out reading

Trained on five sentences (three affirmations and two negations, with both bare and determiner-led subjects), the derived procedure read seven held-out sentences whose every content word was new — four affirmations and three negations — correctly in all seven cases, emitting the exact (subject, predicate, polarity) meaning.

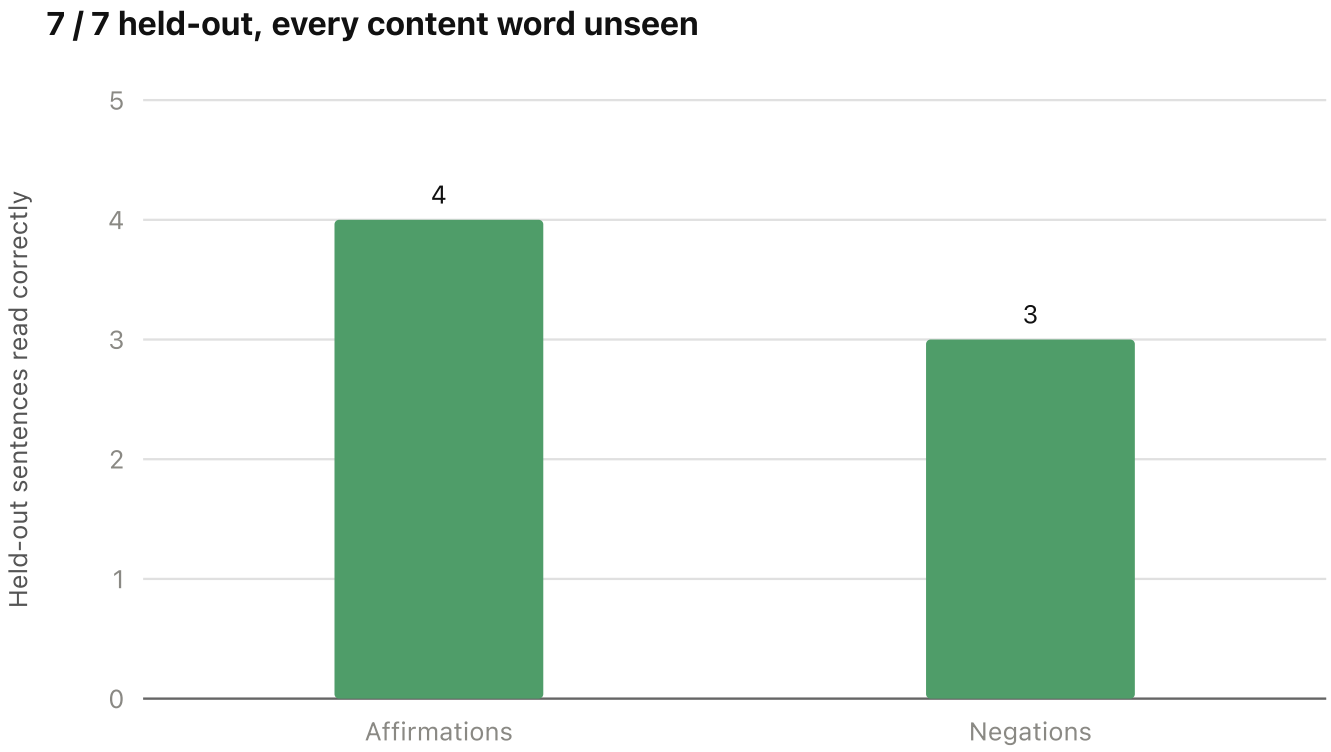

Figure 1. Held-out sentences read correctly, by construction: affirmations 4/4, negations 3/3 — 7/7 in total. Negation is the discriminating construction: no word-class-blind policy satisfies it, so reading it correctly requires the derived branch on the negator class.

Held-out sentence	Emitted meaning
“an ostrich is a bird”	(ostrich, bird, affirms) ✓
“mercury is a metal”	(mercury, metal, affirms) ✓
“a whale is not a fish”	(whale, fish, denies) ✓
“the ledger is not balanced”	(ledger, balanced, denies) ✓

Table 1. A sample of the held-out sentences and the meanings the derived procedure emitted. None of these content words appeared in the five training sentences.

4.2 Under-determination, made explicit

Fitting the training pairs is easy; the load-bearing claim is transfer. The synthesizer searched **251,487** candidate procedures and returned **four** that fit all five training pairs. Rather than hide this under-determination, we test it: all four procedures *agreed* on all seven held-out sentences. Multiple hypotheses, one held-out behavior.

4.3 No model in the reading loop

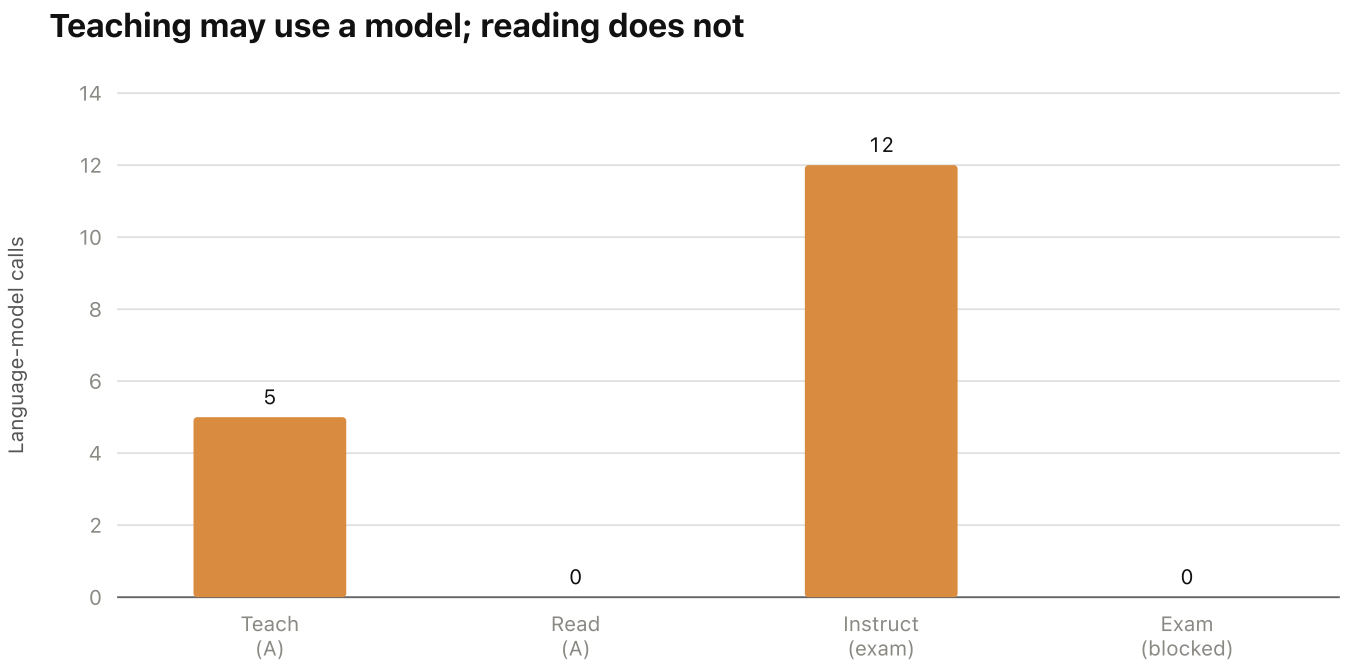

Figure 2. Language-model calls, measured. In a variant where a model acts as a one-time teacher of the five meanings, teaching costs five model calls and the subsequent derivation and reading cost zero. In a separate curriculum exam, instruction costs twelve calls and the exam itself costs zero, with zero blocked reaches — words learned 0/18 → 18/18, sentences 12/12, and six never-shown transfer sentences 6/6.

A corrupted-supervision control — deliberately wrong meanings — yielded *no* procedure of six steps or fewer that fit all five examples within the search bound (an honest bounded-search negative, not a universal impossibility claim). The capability is therefore not an artifact of a search that always finds *something*.

4.4 Coverage at scale

Extending the same machine with additional function-word classes and a taught open-class vocabulary, a scaled run read 11,530 of 12,082 offered sentences (95.4%; 12 unreadable) across seven grammatical constructions, with a taught vocabulary of 1,326 words.

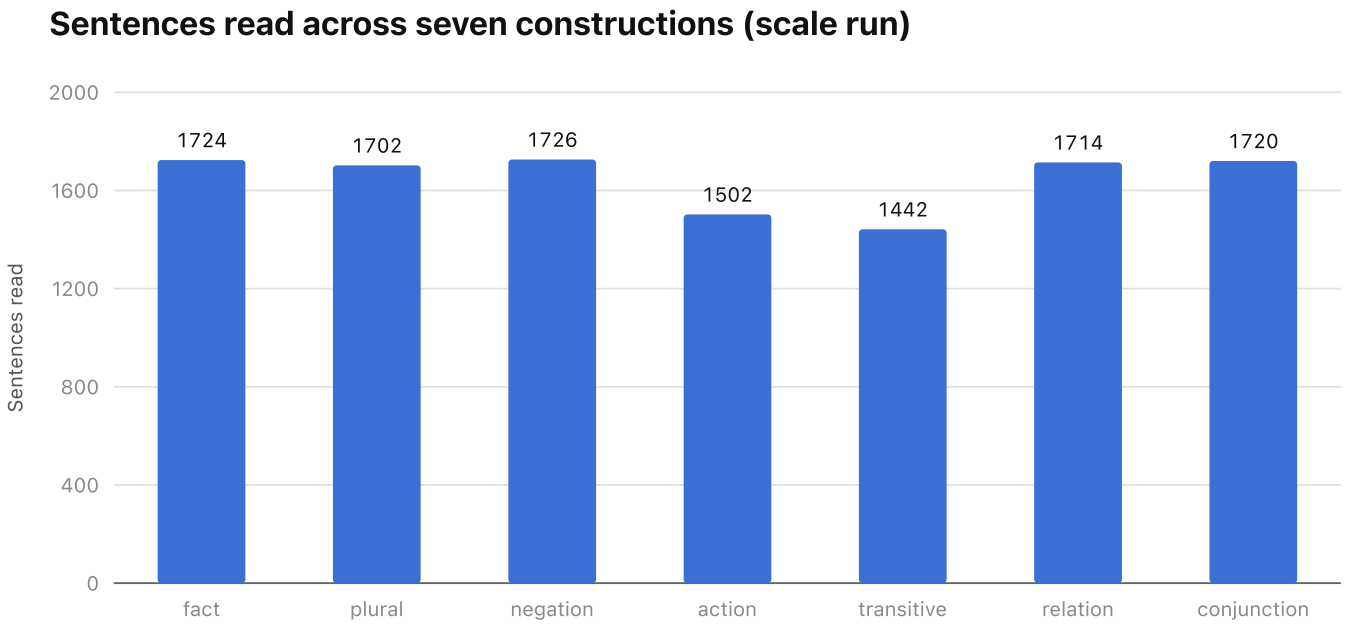

Figure 3. Sentences read by construction in the scaled run (fact, plural, negation, action, transitive, relation, conjunction). This run's headline is *coverage with a taught vocabulary*: its 1,011 model calls are instruction-time vocabulary teaching, not reading calls. The strict zero-calls-during-reading claim is the small headline experiment (§4.1–4.3), not this run.

5 Limitations and conclusion

The headline experiment measures a single training point (five taught sentences), not an accuracy-versus-examples curve; the meanings are simple (subject, predicate, polarity); and word-class membership for a small closed class is supplied as data, with everything about *using* it derived. The scaled run establishes breadth of coverage but uses teaching-time model calls for vocabulary and so does not carry the strict zero-call claim. Within that scope the finding is clear: reading generalizes as a derived program, not as language modeling — five examples yield a procedure that reads unseen sentences with unseen words, and multiple fitting procedures agree on the held-out set, with the reading loop provably free of any model call.

References

[1] S. Gulwani. Automating string processing in spreadsheets using input-output examples. *POPL*, 2011.

[2] K. Ellis et al. DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. *PLDI*, 2021.

[3] L. S. Zettlemoyer, M. Collins. Learning to map sentences to logical form. *UAI*, 2005.

[4] B. Lake, M. Baroni. Generalization without systematicity: compositional skills of sequence-to-sequence networks. *ICML*, 2018.

[5] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5, 1970. (Anti-unification.)