The Learnability of Model-Theoretic Interpretation Functions in Artificial Neural Networks

Adrian Brasoveanu
UC Santa Cruz
abrsvn@gmail.com

Jakub Dotlačil
Utrecht University
j.dotlacil@uu.nl

Abstract

The systematicity of natural language interpretation—our ability to understand novel expressions by compositionally combining familiar elements—has been central to debates about symbolic versus neural approaches to cognition since Fodor and Pylyshyn (1988). We investigate whether artificial neural networks can learn model-theoretic interpretation functions that generalize systematically to out-of-training-sample sentences, framing interpretation as an *encoding task* from discrete linguistic input to continuous truth-conditional representations. We extend Frank et al. (2009) with entity-level semantic representations, modern architectures (LSTM, GRU, Attention with AbsPE/RoPE), principled competing event generation, extended systematicity tests (∼350 vs. ∼80 sentences), and a two-dimensional difficulty analysis disaggregating results by modifier complexity. Across 140 trained models (7 architectures), we find that capacity-matched architectures perform comparably on easy tests, but gated recurrent networks (LSTM and GRU) significantly outperform all transformer architectures on the hardest compositional generalization test (Basic Event), while ungated SRN does not—indicating that the gating mechanism is a critical factor. Entity vectors significantly improve scores on Basic Event across most architectures, with gated architectures benefiting most, validating formal semantics' treatment of entities as important theoretical primitives. The extended test set reveals that systematicity difficulty has two dimensions: the type of systematicity test (as in Frank et al. 2009), and the number of modifiers being composed.

1 Introduction

A fundamental property of human language understanding is *systematicity* (Fodor and Pylyshyn, 1988): our capacity to interpret novel expressions by compositionally combining familiar elements. Understanding "the cat chased the mouse" implies understanding "the mouse chased the cat"—we systematically extend learned interpretation abilities to out-of-training-sample (OOTS) combinations. This property has been central to debates about cognitive architecture: Fodor and Pylyshyn (1988) argued that systematicity requires compositional symbolic representations, and that neural-network models can exhibit it only by covertly implementing a symbol system. If true, this relegates neural networks to mere implementation vehicles, offering little explanatory value for cognitive theory.

Frank et al. (2009) challenged this view by demonstrating that simple recurrent networks (SRNs) can learn to map sentences in a microlanguage to distributed truth-conditional representations that support (some) systematic generalization *without* implementing symbolic structures. They argue that systematicity can arise to a significant extent from *structure in the world* encoded in analogical situation vectors, rather than from symbolic compositionality in mental representations.

1.1 Interpretation as Encoding

We adopt and extend Frank et al.'s paradigm, with an important conceptual clarification: their SRN-based cognitive model is conceptualized as learning *model-theoretic interpretation functions*, which in turn is understood as an *encoding task*. Unlike language modeling (predicting next tokens), classification (assigning labels), or generation (producing output sequences), interpretation encoding maps discrete linguistic input to continuous semantic representations where vector similarity reflects co-occurrence in possible worlds. The target space is not embeddings learned from text distributions, but vectors encoding truth-conditional meaning grounded in a structured model of situations.

This framing connects to sentence encoding research (Reimers and Gurevych, 2019), but with a crucial difference: standard sentence en-

Findings of the Association for Computational Linguistics: ACL 2026, pages 10920–10949

coders learn similarity from text co-occurrence and only indirectly from the latently inferred truth-conditional structure; our encoders learn similarity from explicitly encoded world structure. Representations reflect which events can occur together in valid situations, not which sentences co-occur in corpora. We are formalizing and implementing semantic evaluation in a model-theoretic sense.

1.2 Gaps in Prior Work

While Frank et al. (2009) established the foundational paradigm, and Venhuizen et al. (2019, 2022) extended it (e.g., with expectation-based comprehension), critical gaps remain. First, prior implementations used MATLAB, C and Prolog-based systems (e.g., the original implementation of Frank et al., `dispace` Venhuizen et al. 2019, or `dfs-tools` Venhuizen et al. 2022, for which a light Python wrapper is available `dfs-python-bindings`), limiting accessibility and architectural exploration. Second, the sampling methodology in both Frank et al. and Venhuizen et al. is independent proposition sampling, which underestimates correlations induced by soft constraints—Brasoveanu (2026) provides a detailed discussion. Third, the architectural scope is limited: only SRNs were evaluated, leaving open whether findings generalize to modern architectures. Fourth, semantic representations are limited to truth-value information (type t), but entities (type e) are a co-equal primitive in formal semantics, and a core focus in dynamic semantics (Heim, 1982; Kamp, 1981). Therefore, making entity information explicit might improve systematicity. Finally, systematicity evaluation uses a limited test set in Frank et al. 2009 ($\sim$80 to 120 sentences), potentially missing generalization patterns.

1.3 Our Contributions

We address these gaps with five extensions:

1. **Entity vectors**: Extend representations to include entity-level information (type e), testing whether neural models benefit from this formal semantic primitive. Training targets: 150-dim (truth-conditional only) or 300-dim (truth-conditional + entities).
2. **Modern architectures**: Evaluate LSTMs (Hochreiter and Schmidhuber, 1997), GRUs (Cho et al., 2014), and Transformers (Vaswani et al., 2017) (with absolute and rotary positional encodings) alongside SRNs, using capacity matching ($\sim$66k parameters for truth-conditional only models) for fair comparison.
3. **Principled competing events**: Replace hard-coded rules with universal two-step Z3 validation for principled competing-event generation (candidate must be valid; candidate + described must be unsatisfiable; see below).
4. **Extended systematicity tests**: Expand from $\sim$80 to $\sim$350 test sentences across four groups (Word, Sentence, Complex Event, Basic Event), with complementary train/test splits.
5. **Two-dimensional difficulty analysis**: Disaggregate test results by modifier complexity (canonical, +location, +manner, +both), revealing that systematicity difficulty reflects two factors rather than a single difficulty gradient: the type of systematicity test, and the number of modifiers being composed.

We make publicly available the Python / PyTorch (Paszke et al., 2019) and R based implementation of neural-model evaluation that is the central focus of this paper.[1] Another extension of previous work is sampling methodology: Brasoveanu (2026) proposes *Pool MCMC sampling*, which respects the complete joint distribution defined by hard and soft microworld constraints, unlike independent sampling, yielding more accurate data for truth-conditional representations. This extension critically underpins the neural-model training and evaluation in the present work, but is a separate issue discussed only as needed here.

1.4 Key Findings

Training 140 models (7 architectures$\times\pm$ entity$\times$2 train/test splits$\times$5 seeds), we find:

1. Systematicity difficulty has 2 dimensions. When controlling for modifier complexity, the expected difficulty hierarchy Word $>$ Sentence $>$ Complex Event $>$ Basic Event is preserved across all architecture $\times$ entity conditions. Adding location and manner modifiers introduces a second, orthogonal difficulty gradient that systematically reduces scores within every test group—by up to 47% for the most complex modifier combinations. Thus, compositional generalization difficulty reflects both systematicity type and the number of components that must be simultaneously composed.

2. Gating in recurrent networks is critical. When matched at $\sim$66K parameters, architectures perform comparably on easy tests (Word: all $\sim$1.7), but *gated recurrent networks (GRU* $p = .001$–$.034$,

[1] https://github.com/abrsvn/neural_model_theoretic_interpretation_ACL_2026

LSTM $p = .002$–$.053$) outperform all transformer architectures on the hardest compositional generalization test when entity vectors are available (Basic Event +ent: LSTM/GRU 0.77–0.78, SRN 0.74 vs. Attention 0.67–0.69). SRN—which shares the sequential inductive bias but lacks gating—does not significantly outperform attention in Basic Event. However, SRN outperforms all models (attention and GRU / LSTM) in the second hardest systematicity test (Complex Event).

3. Entity info helps most where it matters. All architectures improve with entity information, with the largest gains on Basic Event (the test requiring generalization to entirely novel entity combinations) and gated architectures benefiting most (GRU: +0.125, $p < .001$; LSTM: +0.106, $p < .001$). Entity vectors also significantly improve Word scores. The Basic Event benefit is almost entirely a generalization (test set) effect, validating formal semantics' treatment of entities as theoretical primitives alongside truth values.

The paper is organized as follows. Section 2 presents the framework. Our full Python reimplementation preserves the same microworld and microlanguage structure as Frank et al. (2009) so that our SRN results are comparable to their original results. We modify the constraint system, situation sampling and microlanguage generation as necessary to obtain correct, reliable neural-network training data, and develop a new event-structure mediated language-to-Z3-logical-formula-translation (also used for principled competing-event generation). Section 3 describes neural architectures, capacity matching, systematicity tests, competing events, and evaluation metrics. Section 4 presents results and discusses implications for compositionality research and formal semantics. Section 5 summarizes and concludes. Section 6 discusses some of the limitations of the present work.

2 Framework

We adopt a truth-conditional, model-theoretic approach to meaning (Montague, 1973; Dowty et al., 1981; Heim and Kratzer, 1998). The meaning of a sentence is its truth conditions—the circumstances under which it is true. In our setup, a *situation* is a complete state of a microworld: an m-dimensional binary vector $\omega \in \{0,1\}^m$ (where $m = 44$, see below) specifying the truth value of each of the m atomic propositions. A sentence's meaning corresponds to the set of situations in which it holds. Neural networks learn to encode sentences as vectors in a learned "situation space" where similarity reflects overlap in truth conditions: sentences true in the same situations receive similar vectors.

2.1 Microworld Structure

The microworld features **17 entities** in six categories: *persons* (charlie, heidi, sophia), *games* (chess, hide&seek, soccer), *toys* (puzzle, ball, doll), *locations* (bathroom, bedroom, playground, street), *play-manners* (well, badly), and *win-manners* (with-ease, with-difficulty). Each person has a "specialty" game (charlie/chess, heidi/hide&seek, sophia/soccer) that influences win probabilities. These entities combine into **44 atomic propositions**: *play(person, game)* yields 9 propositions, *play(person, toy)* yields 9, *win(person)* and *lose(person)* yield 6, *location(person, place)* yields 12, *play(person, manner)* yields 6, and *win(manner)* yields 2. A situation $\omega \in \{0,1\}^{44}$ assigns a truth value to each proposition—1 if the proposition holds in that situation, 0 otherwise.

2.2 Constraint System

Not all 2^{44} possible truth-value assignments represent consistent states of affairs. The microworld enforces **167 hard constraints** specifying which combinations of propositions are jointly possible.

- **Game constraints** (19): A person plays at most one game; chess requires exactly 2 players; soccer / hide&seek require 2–3.
- **Toy constraints** (33): A person plays with at most 1 toy; puzzle is single-player; soccer requires ball; no toys during chess / hide&seek.
- **Location constraints** (72): Each person is in exactly one location; chess players must be co-located; activities have location restrictions (chess: bedroom/playground; soccer: street only; puzzle: bedroom only etc.).
- **Winning constraints** (31): A person cannot both win and lose; at most one person wins; any winner or loser must be playing a game.
- **Manner constraints** (12): A person has at most one play manner; having a play manner requires playing a game.

All constraints are formalized as Z3 formulas of the form $\neg(c_1 \wedge c_2 \wedge \ldots)$—conjunctions of conditions that cannot simultaneously hold. This enables consistency checking and competing event generation via satisfiability queries. Of $2^{44} \approx 17.6$ trillion possible binary vectors, only **28,676** satisfy all hard constraints—a reduction factor of ∼613 mil-

lion. This rich constraint structure is essential for learning systematicity: constraints encode which propositions can be jointly true, and this structure is reflected in the learned representations.

We also impose **soft** (not only hard) **constraints** that induce correlations between propositions, e.g., playing well makes winning more likely. Soft constraints can be violated, but they shape the probability distribution over situations, which in turn shapes the learned meaning representations.

2.3 Truth-Conditional Representations

Situation sampling is different from Frank et al. (2009), while still enabling comparison with the original results. We learn distributed representations of truth conditions from sampled situations via a competitive neural model with $n = 150$ neurons—a Self-Organizing Map (Kohonen, 1995, 2013) variant (Python reimplementation of the competitive layer in Frank et al. 2009; see Brasoveanu 2026 for details). The trained competitive model outputs a weight matrix $\boldsymbol{W} \in [0,1]^{150\times 44}$ that assigns meanings / distributed truth-conditional representations to the atomic props $\varphi_k (k = 1 \ldots 44)$ in our microworld: $[\![\varphi_k]\!] = \boldsymbol{W}_{\cdot k}$ (the kth $\boldsymbol{W}$ column) is a vector in $[0,1]^{150}$.

The meaning of complex sentences is computed via differentiable vectorized fuzzy logic operations on these atomic-prop vectors: negation $[\![\neg\varphi]\!] = \mathbf{1} - [\![\varphi]\!]$, conjunction $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \odot [\![\psi]\!]$ (element-wise / Hadamard product), disjunction $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] + [\![\psi]\!] - [\![\varphi]\!] \odot [\![\psi]\!]$. All these meaning vectors support probability estimation: $P(\varphi) \approx \frac{1}{n}\sum_i^n [\![\varphi]\!]$, $P(\varphi \wedge \psi) \approx \frac{1}{n}\sum_i^n ([\![\varphi]\!] \odot [\![\psi]\!])$, and $P(\varphi \mid \psi) \approx \frac{P(\varphi\wedge\psi)}{P(\psi)}$. These probability approximations are very accurate: correlations with underlying situation-sample frequencies are $r = 1.0$ for base probabilities and conjunctions, and $r = 0.998$ for conditionals (Brasoveanu, 2026).

Entity vectors. Formal semantics treats entities (type e) and truth values (type t) as co-equal primitives. To test whether neural models benefit from explicit entity information, we extend the competitive model input from 44 to 61 by appending 17 binary entity-presence indicators: entity e is 1 in a situation if e appears as an argument in any true atomic prop. Training the competitive model on these 61-dim binary vectors produces 17 entity-presence distributed representations in $[0,1]^{150}$ in addition to the 44 atomic prop meanings: a weight matrix $\boldsymbol{W}^+ \in [0,1]^{150\times 61}$. Training neural interpretation functions will target either 150-dim vectors (propositional meanings only, derived from the first 44 columns of $\boldsymbol{W}^+$) or 300-dim vectors (we additionally concatenate a 150-dim entity-presence vector derived from the last 17 columns of $\boldsymbol{W}^+$).

2.4 Microlanguage & Z3 Logical Forms

The microlanguage consists of **13,556 sentences** (like Frank et al.) and is generated by a feature-based context-free grammar with **40 lexical items** and rules for active voice (charlie plays chess), passive voice (chess is played by charlie), beat / lose constructions (sophia beats boy, boy loses to sophia), and for optional modifiers for manner (plays well, wins with ease) and location (in bedroom, outside). **Four synonym pairs** enable systematicity testing via lexical substitution—charlie/boy, soccer/football, puzzle/jigsaw, bathroom/shower. **Existential quantifiers** are interpreted as disjunctions over their denotations; someone $\rightarrow$ charlie $\vee$ heidi $\vee$ sophia; girl $\rightarrow$ heidi $\vee$ sophia; game $\rightarrow$ chess $\vee$ hide&seek $\vee$ soccer; inside $\rightarrow$ bathroom $\vee$ bedroom etc.

Unlike Frank et al., we **translate sentences into Z3 logical formulas via an intermediate event structure** (thematic roles: agent, theme, patient, location, manner), e.g., *sophia beats charlie at chess* translates to *win(sophia)* $\wedge$ *lose(charlie)* $\wedge$ *play(sophia, chess)* $\wedge$ *play(charlie, chess)*. We generate target vectors for Z3 translations by applying fuzzy logic operations to the competitive-model representations of the atomic propositions in the Z3 formula. These **target vectors (150-dim w/o entity and 300-dim w/ entity)** serve as supervision for neural-network training.

A sentence is *consistent* iff it is true in at least one of the 28,676 situations satisfying all the hard constraints. After filtering out inconsistent sentences, 6,279–6,789 training sentences and 2,330–2,840 test sentences remain in each of the **2 train/test splits**. Sentences featuring toys constitute only 6% of training data, and are repeated $4\times$ per training epoch to ensure adequate exposure for the hardest systematicity test (Basic Event).

3 Models: Training and Evaluation

3.1 Neural Architectures and Training

We evaluate 7 architectures in 2 experiments, all performing the same encoding task: mapping token sequences (sentences from the microlanguage) to situation vectors via learned parameters. All share

the same structure: embedding layer → sequence encoder → linear projection → sigmoid, producing output in $[0,1]^n$ ($n = 150$) if truth-conditional only (**w/o entities**), or in $[0,1]^{2n}$ ($2n = 300$) if truth-conditional + entities (**w/ entities**). Training minimizes MSE against competitive target vectors.

Capacity matching. To isolate architectural effects from parameter count, we match models at approximately 66k parameters (w/o entities). The SRN (Elman, 1990) is Frank et al.'s original architecture. The LSTM uses a standard modern forget-gate variant based on Hochreiter and Schmidhuber (1997) and Gers et al. (2000). The GRU uses a 2-gate recurrent cell (Cho et al., 2014), providing a minimal gated architecture without the separate cell state of LSTM (GRU models were originally estimated as part of a follow-up; for ease and uniformity of exposition, we report them as part of Experiment 1 throughout the paper). Attention models use 2-layer transformers with 4 heads each, $d_{model} = 48$ (80 in follow-up Experiment 2), and final-position output (no BERT-like CLS token). AbsPE uses sinusoidal positional encoding (Vaswani et al., 2017). RoPE uses rotary positional encoding (Su et al., 2021).

Architecture	Hidden	Layers	Heads	Params
SRN	178	1	—	66,010
GRU	90	1	—	66,210
LSTM	80	1	—	66,950
Attention AbsPE	48	2	4	65,670
Attention RoPE	48	2	4	65,670

With entity vectors, parameter counts increase to 73k–93k, with attention models gaining fewest additional parameters (7.3k vs. 26.9k for SRN) because they have the smallest hidden dim (48). We also explore hidden-dim matched attention (80).

Training: AdamW optimizer (Loshchilov and Hutter, 2019) with weight decay 10^{-5}, batch size 64, gradient clipping at norm 3.0, and adaptive learning-rate schedule (warmup → plateau / reduction → cyclical). All models train for 750 epochs (saturation after ∼100 epochs; see Appendices A,B). We train 5 seeds per architecture × 2 train/test splits × ±entity. Experiment 1 investigates SRN, GRU, LSTM, AbsPE 48, and RoPE 48 (100 models). The follow-up Experiment 2 adds higher-capacity (hidden 80) AbsPE and RoPE attention models (40 models). Total: 140 models.

3.2 Evaluation: Systematicity Tests

Like Frank et al., we evaluate compositional generalization via 4 test groups of increasing difficulty, using 2 train/test splits where patterns held out in one split are trained in the other.

Word Group (12 test sentences): tests synonym generalization. Models see *charlie plays football* and *boy plays soccer* in training; at test time, they interpret *charlie plays soccer* and *boy plays football*. This is the easiest: truth conditions for test sentences appear in training for sentences with the same syntactic structure but different lexical forms.

Sentence Group (160 test sentences): tests novel argument combinations in *beat/lose* constructions. Models see various *X beats/loses to Y* pairs during training; at test time, they see held-out person pairs. Crucially, if the test has *X beats Y*, the training set included *Y loses to X* and vice-versa: the model has seen the test truth-conditions under different syntactic structures and lexical forms in training. This tests compositional understanding of novel agent-patient combinations, but where all truth-conditions have been seen during training.

Complex Event Group (∼120 test sentences): tests novel conjunctions of activities and locations. Models see *charlie plays chess in bedroom* but not *charlie plays chess in playground* (or vice versa). Target truth conditions—the *conjunction* of activity and location—are never seen in training, but the individual conjuncts are. This tests sentence-level composition of novel truth conditions.

Basic Event Group (∼60 test sentences): tests entirely novel person-toy combinations. If the test includes *heidi plays with ball*, the model has never seen target vectors for the atomic prop *play(heidi, ball)* in any form (but it did see other sentences for other atomic props, including other *play* sentences involving *heidi* and *ball*, just never together). This is the hardest: we test sub-sentential composition of novel truth conditions for atomic props.

3.3 Evaluation: Metric and Competing Events

Compositional-generalization capabilities are evaluated using comprehension scores, which are calculated based on the first 150 dimensions of the target semantic vector (truth conditions only; entity presence is ignored for comprehension evaluation).

Comprehension score (Frank et al., 2009): given network output $\mathbf{z}$ for a sentence describing event $\mathbf{a}$ (this is its exact meaning vector, used in training), we compute the conditional probability $P(\mathbf{a} \mid \mathbf{z})$, which is high if $\mathbf{z}$ is a good meaning prediction. The comprehension score normalizes this conditional probability relative to the prior $P(\mathbf{a})$:

$$\text{Comp}(\mathbf{a} \mid \mathbf{z}) = \begin{cases} \frac{P(\mathbf{a}|\mathbf{z}) - P(\mathbf{a})}{1 - P(\mathbf{a})} & \text{if } P(\mathbf{a} \mid \mathbf{z}) \geq P(\mathbf{a}) \\ \frac{P(\mathbf{a}|\mathbf{z}) - P(\mathbf{a})}{P(\mathbf{a})} & \text{otherwise} \end{cases}$$

This yields +1 for perfect understanding ($P(\mathbf{a} \mid \mathbf{z}) = 1$), 0 for unrelated predictions $\mathbf{z}$, and -1 for complete misunderstanding ($P(\mathbf{a} \mid \mathbf{z}) = 0$).

Positive comprehension scores alone do not demonstrate understanding. A model that learns only "someone wins, someone loses" scores positively on "charlie beats heidi" while also activating the *wrong* interpretation (heidi wins, charlie loses). True systematicity requires preferring described events over **competing events**—alternatives that are individually valid but inconsistent with what the sentence actually describes. Rather than hard-coding competing events per test group like Frank et al., we generate them systematically via a two-step Z3 validation process. A candidate event c competes with described event d iff:

1. Validity: SAT($c \wedge$ constraints)—c is possible in at least one of the 28,676 valid situations
2. Competition: UNSAT($c \wedge d \wedge$ constraints)—c and d cannot both hold

To generate competing formula candidates, we vary exactly one event-structure slot at a time (theme, agent/patient, location, or manner, depending on the pattern), then validate via this two-step procedure. Thus, competing formulas are local structural alternatives to the described interpretation: they are individually valid, contradict the sentence's meaning, and differ from it in a controlled slotwise way.

Advantage score. For each test sentence, we compute comprehension scores for both described and competing events. The *advantage*, i.e., described score minus mean competing score, measures systematic understanding and is our primary metric: high advantage means the model correctly identifies the target interpretation over alternatives.

4 Results and Discussion

Table 1 provides test-only advantage scores across the 5 architectures in Experiment 1 (SRN, GRU, LSTM, AbsPE, RoPE), the 4 test groups, and ±entity. Scores are averaged over 10 models (2 train/test splits × 5 seeds). Figure 1 displays Experiment 1 test-only means and standard errors (SEs) of described and competing scores separately, showing that advantages arise from both higher described scores and more negative competing scores. SEs are ≤0.05 (consistent patterns across seeds).

Arch	Ent	Word	Sent	Cmplx	Basic
SRN	−ent	1.65	1.16	1.36	0.67
	+ent	1.73	1.12	1.37	0.74
GRU	−ent	1.72	1.13	1.30	0.65
	+ent	1.72	1.08	1.20	0.78
LSTM	−ent	1.68	1.14	1.28	0.67
	+ent	1.71	1.12	1.24	0.77
Attn AbsPE	−ent	1.71	0.95	1.05	0.59
	+ent	1.72	1.02	1.10	0.68
Attn RoPE	−ent	1.67	1.14	1.07	0.63
	+ent	1.70	1.12	1.05	0.69

Table 1: Experiment 1: test-only advantage (higher is better) by architecture, test group and ±entity, aggregated across all test sentences including modifier variants. GRU & LSTM +ent achieve the highest Basic Event scores (**0.78** and 0.77; statistically indistinguishable: $p = 1.00$). See Table 2 for how modifier complexity affects these aggregate scores.

Test	Can	+Loc	+Man	+L+M	Agg
Word	1.94	—	1.58	—	1.70
Sentence	1.62	1.26	1.12	0.86	1.10
Complex	1.37	—	1.10	—	1.19
Basic	0.83	0.65	—	—	0.68

Table 2: Experiment 1: test-only advantage by test group and modifier complexity, averaged across all 5 architectures and ±entity settings. **Can**: canonical (unmodified) sentences. **+Loc/+Man/+L+M**: sentences with location, manner, or both modifiers. —: modifier types absent from test group. **Agg**: aggregate over all test sentences. Canonical column preserves the expected hierarchy Word > Sentence > Complex > Basic. **Agg** reverses Sentence and Complex b/c Sentence is 94% non-canonical vs. 67% for Complex—see **Observation 1**.

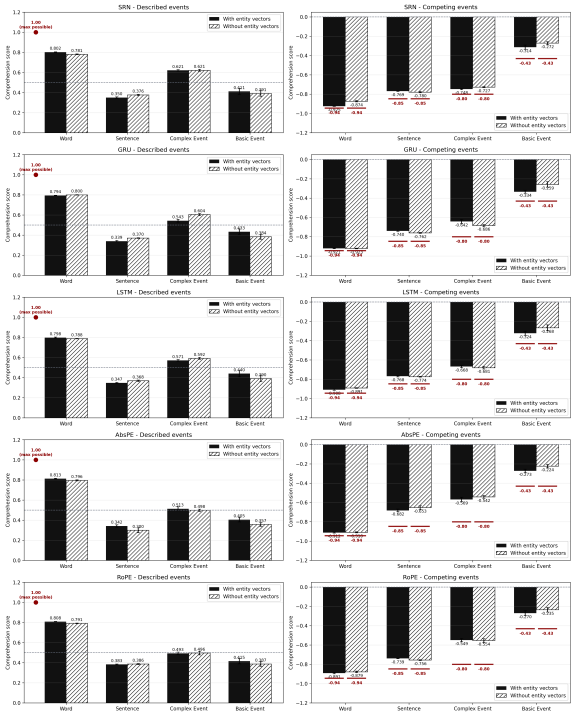

Figure 1: Experiment 1: test-only comp. scores for described (left) and competing (right) events aggregated across all modifier variants. Solid vs. hatched bars: mean scores ±entity. Whiskers: SEs. Dark red: max/min for ideal perfect model.

Observation 1: Difficulty has 2 dimensions, test type and modifier complexity. Table 2 reveals that systematicity difficulty is not a single gradient but the result of 2 factors. The first is the test group: Word (synonym substitution) > Sentence (novel argument combinations with known truth conditions) > Complex Event (novel conjunctions of activities and locations) > Basic Event (entirely novel atomic props). When we restrict evaluation to *canonical* (unmodified) test sentences, this hierarchy is preserved across the 5 Table 1/Figure 1 architectures: Word 1.94 > Sentence 1.62 > Complex 1.37 > Basic 0.83 (Table 2: grand means across architectures & ±entity). This replicates Frank et al.'s ordering.

The second dimension is *modifier complexity*: adding optional location and manner modifiers to test sentences systematically reduces advantage scores within every test group. In the Sentence test, canonical sentences (e.g., *sophia beats charlie*) score 1.62, adding a location modifier (*sophia beats charlie in the bedroom*) drops this to 1.26, adding a manner modifier (*sophia beats charlie with ease*) drops it to 1.12, and combining both (*sophia beats charlie with ease in the bedroom*) drops it further to 0.86—a 47% reduction from canonical. The Complex Event test shows a similar pattern: canonical 1.37 vs. manner-modified 1.10 (19% reduction). Basic Event: canonical 0.83 vs. location-modified 0.65 (22% reduction). Even Word, the easiest test, shows the effect: canonical 1.94 vs. manner-modified 1.58 (18% reduction).

This two-dimensional structure is our *first main finding*. It explains why the aggregate ordering in Table 1 (also Figure 1) reverses Sentence (1.10) and Complex (1.19): the Sentence group is 94% non-canonical sentences, dominated by the hardest modifier category (location+manner, 44% of Sentence, scoring just 0.86). In contrast, Complex is 67% non-canonical with only manner modifiers (scoring 1.10). The aggregate scores thus reflect different *mixtures* of modifier complexity, not a true reversal of the underlying test-type difficulty. Each modifier adds conjuncts to the described Z3 formula, and the comprehension score evaluates the model's output against this *full*, harder conjunction. The per-architecture compositional breakdown in Appendix D confirms that this pattern holds across all architectures and entity settings.

Observation 2: Robust recurrent vs. attention effects across difficulty levels. Without entity vectors, the 5 Experiment 1 architectures achieve comparable Word scores (1.65–1.72; Table 1). In the full 7-architecture analysis (including the Attention models with hidden-dim of 80 from follow-up Experiment 2; see below), the architecture effect on Word is small but significant ($F(6, 148.5) = 2.55$, $p = .02$; Table 19), reflecting GRU's high Word scores. Architecture becomes the dominant factor on harder tests: Sentence $F(6, 139.0) = 20.72$, $p < .001$; Complex $F(6, 144.5) = 48.76$, $p < .001$; Basic $F(6, 137.0) = 7.18$, $p < .001$. These differences emerge *even at the canonical level*: on Sentence, AbsPE scores 1.27 vs. SRN 1.63 and LSTM 1.69 (Table 11)—a large deficit that cannot be attributed to modifier composition; on canonical Complex, AbsPE scores 1.16 vs. SRN 1.60 (Table 12)—a 28% gap. RoPE performs better than AbsPE on canonical Sentence (1.72, Table 11), matching or exceeding recurrent models, but still falls behind on canonical Complex (1.19 vs. SRN 1.60; Table 12). Modifier complexity amplifies these architecture differences but does not create them: the attention deficit is an intrinsic architectural effect, not an artifact of modifier complexity. Capacity matching controls for parameter count but cannot control for architectural inductive biases, which manifest precisely on tests requiring genuine compositional generalization.

Observation 3: Gated recurrent networks outperform transformers on Basic Event. The hardest test reveals a clear architecture gradient: gated recurrent (LSTM 0.77, GRU 0.78) > SRN (0.74) > attention (RoPE 0.69, AbsPE 0.68) with entity vectors; without entities, LSTM / SRN (0.67) > GRU (0.65) > RoPE (0.63) > AbsPE (0.59). Table 20 shows the broad recurrent > attention pattern on Basic Event, and Table 22 gives the entity-conditioned pairwise contrasts. In the +ent condition, GRU significantly outperforms all 4 attention variants ($p = .001$–$.034$), while LSTM does so for 3 of 4, with a near-threshold contrast against RoPE ($p = .053$). Critically, LSTM and GRU are statistically indistinguishable ($p = 1.00$), while SRN—which shares the sequential inductive bias but lacks gating—does not significantly outperform any attention model in the +ent condition ($p = .15$–$.73$). In the −ent condition, SRN does outperform some attention variants (SRN–AbsPE $p = .04$; SRN–RoPE H80 $p = .007$), but this advantage largely vanishes once entity vectors are added. This is our *second main finding: gated recurrent networks (LSTM and GRU)—but not ungated SRN—consistently outperform modern transformer architectures on true compositional generalization*.

The three-way SRN/GRU/LSTM contrast directly demonstrates that gating, not sequential processing alone (or the LSTM cell state), is the critical factor: SRN (sequential, ungated) $\approx$ attention, while GRU (sequential, gated) = LSTM (sequential, gated + cell state) $\gg$ attention. Investigating whether gating is a key ingredient for compositional generalization was the primary reason for including GRU in Experiment 1.

Observation 4: Entity vectors help most on Basic Event. This is our *third main finding: the improvement from adding entity information is largest precisely where it should matter most—deriving truth conditions involving novel entity combinations* (novel atomic props). Entity vectors provide +0.06 to +0.13 improvement on Basic Event across the five Experiment 1 architectures (Table 1), with a strong overall entity effect on Basic Event ($F(1, 189.0) = 59.18$, $p < .001$; Table 23). Table 24 shows that gated architectures benefit most: GRU +0.125 ($p < .001$) and LSTM +0.106 ($p < .001$). Entity vectors also significantly improve Word scores ($F(1, 130.2) = 20.45$, $p < .001$; Table 23), suggesting that entity-presence information aids even synonym substitution—plausibly by sharpening the model's representation of which entities are involved in each event. On Sentence and Complex, the entity main effect is null ($p = .95$ and $p = .44$, respectively; Table 23). This validates the formal semantics intuition that entities are fundamental primitives, with the largest benefit on Basic Event (within Basic, gated recurrent networks exhibit the largest per-architecture benefit), and a secondary benefit on Word.

Observation 5: Substantial generalization gap, especially on hard tests. Comparing train and test performance on Basic Event reveals the challenge of true compositional generalization. All models achieve strong training performance ($\sim$1.01–1.10; Table 18), but test performance drops substantially ($\sim$0.57–0.78; Table 17). Entity vectors help primarily at test time, indicating a genuine benefit for compositional generalization rather than improved training fit.

The generalization gap is not limited to Basic Event. Complex Event also exhibits a strong generalization gap, with architecture continuing to matter under generalization (Appendix F). Detailed sentence-level and generalization-gap results are available in the repository's `statistical_analysis` directory.

Why do gated recurrent models generalize better? In Experiment 1, GRU/LSTM were the best-performing architectures on Basic Event (Observation 3). A follow-up experiment investigates why. The first reason that comes to mind is *representational capacity per position*: capacity matching yields hidden dimensions of 178 (SRN), 90 (GRU), and 80 (LSTM) vs. 48 (attention). Higher per-position capacity may help recurrent models maintain richer intermediate representations during sequential processing. To investigate this possibility, Experiment 2 trained attention models with hidden dim (d_{model}) of 80, the same as the LSTM; the rest of the setup was unchanged (2 layers, 4 heads, same training procedure). This resulted in models with $\sim$170,000 parameters w/o entity and $\sim$183,000 w/ entity, substantially more than the recurrent models above. As Figure 2 and Table 17 show, this increase in capacity does not materially improve over the H48 (hidden-dim=48) attention models above, and in several cases H80 is slightly worse. Thus, capacity *simpliciter* is not the reason for the higher systematicity of gated recurrent models.

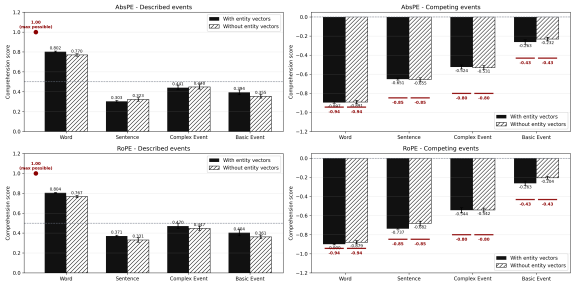

Figure 2: Experiment 2: test-only comp. scores for described (left) and competing (right) events for high-capacity Attention (Transformer) models with a hidden dim of 80, yielding models with $\sim$170k parameters w/o entity and $\sim$183k parameters w/ entity. Bars, whiskers, dark red: same as in Figure 1.

Another possible reason for the better compositional generalization of GRU and LSTM is *sequential inductive bias*: recurrent models process tokens one-by-one, naturally accumulating meaning incrementally. This creates a temporal bottleneck that may encourage learning abstract, reusable representations. The reason for LSTM's top performance relative to SRN is unclear, given its full additional machinery (3 gates + cell state). However, the equally maximal performance of the GRU models, which have reset and update gates but no separate cell state (Cho et al., 2014), indicates that the full LSTM machinery is not needed. This points to the *gating mechanism* as a critical factor for compositional generalization (Observation 3), not the sequential bottleneck alone or the specific LSTM

architecture involving cell state.

Standard self-attention provides an inductive bias, but perhaps not exactly the right one for compositional interpretation. Recent work suggests that self-attention *entangles* object-level information with relational information (Webb et al. 2024; Altabaa and Lafferty 2025 and references). Every attention head computes pairwise interactions between all positions, mixing "what" information (lexical features) with "where/how" information (positional relations, structural roles). For compositional semantics, this conflation may be problematic: lexical semantics (the meaning of individual words) and compositional semantics (how meanings combine) may benefit from separate processing streams.

The sequential bottleneck of recurrent networks may implicitly achieve a form of disentanglement: at each timestep, the model must compress all prior context into a fixed-size hidden state. In addition, learned gates enable selective retention and updating of information across timesteps, perhaps maintaining compositionally relevant features while discarding irrelevant ones. Notably, GRU's minimal 2-gate mechanism (reset + update) achieves the same compositional generalization as LSTM's more complex 3-gate + cell state architecture, suggesting that the key property is the *ability to gate information flow at all*, not the specific gating architecture. The simple SRN overwriting dynamics does not seem to provide enough control over information flow for robust sub-sentential compositional generalization, despite sharing the sequential inductive bias. Interestingly, the gating advantage is specific to Basic Event (sub-sentential composition). *On Complex Event (sentence-level composition), SRN significantly outperforms both gated architectures* (Table 21).

5 Conclusion

Implications for compositionality research. Our results challenge the assumption that architectural sophistication guarantees compositional ability. The hardest systematicity tests—interpreting sentences whose truth conditions were never seen during training—significantly favor recurrent architectures over attention, but all architectures are ultimately doing fairly poorly. This aligns with findings from SCAN (Lake and Baroni, 2018), COGS (Kim and Linzen, 2020), and CFQ (Keysers et al., 2020) showing that neural models struggle with compositional generalization, though our setting differs by targeting truth conditions grounded in model-theoretic structures rather than symbolic sequence transduction.

Implications for formal semantics. The consistent benefit of entity-presence vectors, especially for Basic Event, provides empirical support for the formal semantic treatment of entities as theoretical primitives alongside truth values. Just as dynamic semantics (Kamp, 1981; Heim, 1982) requires entity discourse referents for adequate representation of quantification and anaphora, neural compositional models benefit from disentangling and making explicit entity-level information when generalizing to novel atomic props (the entity benefit on Basic is almost entirely a generalization effect, see Observation 5). The secondary entity benefit on Word suggests that entity-presence information aids sub-sentential compositional processing more broadly. The two-dimensional difficulty structure (Observation 1) further supports a compositional semantics perspective: modifier complexity—the number of semantic components that must be simultaneously interpreted—is a principled, linguistically grounded predictor of generalization difficulty, distinct from systematicity test type.

Implications for the systematicity debate. All architectures exhibit positive systematicity even on the hardest test (Basic Event: 0.57–0.78), demonstrating that neural networks can learn interpretation functions, supporting compositional generalization without implementing classical symbols. However, systematicity is *graded* along the two dimensions mentioned above, from near-ceiling on synonym substitution in canonical (unmodified) sentences to very modest on novel atomic propositions, especially when increasing the number of modifiers being composed. The degree of systematicity also depends on architecture and meaning representation: gated recurrent networks show the clearest advantage on Basic Event, with ungated SRN intermediate on that test but superior on Complex Event, while entity vectors help most on Basic Event and secondarily on Word.

The experiments reveal a complex neural systematicity landscape emerging from the interaction of appropriate inductive biases, sufficiently rich semantic representations, and the compositional-complexity demands of the target interpretation. We close with a discussion of limitations and suggestions for potential new investigations of this systematicity landscape.

6 Limitations

Our work has several limitations.

Scale. Our microworld (44 propositions, 13,556 sentences) is orders of magnitude smaller than real-world semantic domains. Extending to larger microworlds and languages would test whether findings generalize, and might provide a more suitable dataset for transformer architectures that seem to require data "burstiness" to develop certain generalization capabilities (Chan et al., 2022).

Richer phenomena—universal quantification, negation scope, intensionality, temporal and modal operators etc.—remain unexplored.

Mechanistic understanding. We document what architectures achieve but not *how* they implement compositional operations internally. Mechanistic interpretability methods (circuit analysis, causal tracing) could reveal whether models learn genuine compositional structure or exploit dataset-specific regularities (despite mitigating the chance of such confounds with the two train/test splits).

Single training paradigm. All models use the same two-stage pipeline: pre-computing dense situation vectors via competitive layer training, then supervised training using these dense vectors as targets. Alternative training regimes (end-to-end, contrastive, multi-task) might yield different architectural rankings. Training end-to-end with binary truth conditions as direct supervision would let dense representations emerge through the interpretation task itself. While binary target vectors of size 28,676 would lead to hidden-to-output layers that dwarf the rest of the model, thinning the PoolMCMC samples by a higher factor (e.g., $\sim$500; see also model subsampling in Venhuizen et al. 2022) could make this feasible.

Granularity of the gating finding. The GRU-LSTM comparison indicates that gating, not the specific LSTM architecture, drives the compositional generalization advantage (Observation 3). GRU's minimal 2-gate mechanism matches LSTM's 3-gate + cell state architecture exactly. This raises the question: what about even simpler gated units (e.g., minimal gated units with a single gate)? Additionally, while gating helps on Basic Event (novel entity combinations), SRN outperforms both gated architectures on Complex Event (novel conjunctions of known components). Understanding why gating helps entity-level generalization but slightly hurts compositional generalization of known propositional components remains an open question.

Relational architectures. Our attention models use standard self-attention, which entangles object-level information with relational information in every layer. Recent work suggests that *disentangling* these types of information may be crucial for compositional generalization. Relational abstractors (Altabaa et al., 2024) implement "relational cross-attention" that separates relational reasoning from object-level feature processing. The Dual Attention Transformer (Altabaa and Lafferty, 2025) introduces parallel attention mechanisms for 'sensory' (object-level) and relational information routing. These architectures operationalize the "relational bottleneck" hypothesis (Webb et al., 2024), which proposes that forcing networks to represent relations independently of object features improves abstraction and generalization from limited data. Testing whether such architectural disentanglement of lexical semantics (object/entity properties) from compositional semantics (relational structure) improves systematicity in our paradigm is a natural and promising next step.

Acknowledgments

We thank the anonymous ACL (ARR January 2026) reviewers for their helpful feedback and suggestions, and Stefan Frank, Harm Brouwer, Noortje Venhuizen, and the audiences of UCSC S-Circle (December 2025), and CPL 2025 (December 2025) for all their comments about this work. Our conversations with Stefan Frank, Harm Brouwer and Noortje Venhuizen in particular have been very helpful and enlightening. We are also grateful to Stefan Frank for sharing his original Matlab implementation of the Frank et al. (2009) paper, which was the starting point for the Python implementation underlying the present results.

LLM assistants were used in circumscribed, closely supervised ways during code development and the editing process of the manuscript. On the spectrum between local, narrow autocomplete vs. global, large-scale LLM-based generation of code or text, this work stays decidedly close to the local / narrow end of the spectrum. The authors retain full responsibility for all content, and all errors remain our own.

References

Awni Altabaa and John Lafferty. 2025. Disentangling and integrating relational and sensory information in transformer architectures. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 1271–1297. PMLR.

Awni Altabaa, Taylor Webb, Jonathan D Cohen, and John Lafferty. 2024. Abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in transformers. In *ICLR*.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Adrian Brasoveanu. 2026. Distributed situation models: Probabilistic representations of truth-conditions. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 48.

Stephanie C.Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, James McClelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 18878–18891.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Reidel, Dordrecht.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Stefan L Frank, Willem FG Haselager, and Iris Van Rooij. 2009. Connectionist semantic systematicity. *Cognition*, 110(3):358–379.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Irene Heim. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. thesis, University of Massachusetts, Amherst.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell, Oxford.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hans Kamp. 1981. A theory of truth and semantic representation. In *Formal Methods in the Study of Language*, pages 277–322, Amsterdam. Mathematical Centre.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105. Association for Computational Linguistics.

Teuvo Kohonen. 1995. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg.

Teuvo Kohonen. 2013. Essentials of the self-organizing map. *Neural Networks*, 37:52–65.

Alexandra Kuznetsova, Per B Brockhoff, and Rune HB Christensen. 2017. lmerTest package: Tests in linear mixed effects models. *Journal of Statistical Software*, 82(13):1–26.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Øyvind Langsrud. 2003. Anova for unbalanced data: Use type ii instead of type iii sums of squares. *Statistics and Computing*, 13:163–167.

Russell V Lenth. 2024. *emmeans: Estimated Marginal Means, aka Least-Squares Means*. R package version 1.10.0.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 221–242. Reidel, Dordrecht.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Noortje J. Venhuizen, Matthew W. Crocker, and Harm Brouwer. 2019. Expectation-based comprehension: Modeling the interaction of world knowledge and linguistic experience. *Discourse Processes*, 56(3):229–255.

Noortje J. Venhuizen, Petra Hendriks, Matthew W. Crocker, and Harm Brouwer. 2022. Distributional formal semantics. *Information and Computation*, 287:1–21.

Taylor W Webb, Steven M Frankland, Awni Altabaa, and 1 others. 2024. The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*.

A Training Details

This appendix summarizes the architecture, training, evaluation, and data-generation details used in the experiments.

A.1 Architecture Specifications

Table 3 provides exact specifications for the five Experiment 1 architecture families. Experiment 2 uses the same two Attention architectures with hidden dimension 80. All models share the same input/output structure: embedding layer → sequence encoder → linear projection → sigmoid activation.

Parameter	SRN	GRU	LSTM	Attn AbsPE	Attn RoPE
Architecture					
Hidden dim	178	90	80	48	48
Num layers	1	1	1	2	2
Attn heads	—	—	—	4	4
Head dim	—	—	—	12	12
FFN expansion	—	—	—	4×	4×
FFN hidden dim	—	—	—	192	192
FFN act.	—	—	—	GELU	GELU
Pos. enc.	implicit	implicit	implicit	sinusoidal	rotary
Recurrent act.	sigmoid	tanh/sigmoid	tanh/sigmoid	—	—
Output act.	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
Parameters (without entity vectors, output=150)					
Total params	66,010	66,210	66,950	65,670	65,670
Parameters (with entity vectors, output=300)					
Total params	92,860	79,860	79,100	73,020	73,020

Table 3: Detailed architecture specifications for the five Experiment 1 architecture families.

Simple Recurrent Network (SRN). Standard implementation following Elman (1990) and Frank et al. (2009). The embedding layer serves as the input-to-hidden transformation (W_x), so the recurrent cell applies only the hidden-to-hidden transformation:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b) \tag{1}$$

where σ is the sigmoid activation. The hidden state at each timestep overwrites the previous one entirely—there is no gating mechanism to selectively retain or discard information.

LSTM. LSTM with forget gate, following Hochreiter and Schmidhuber (1997) and the forget-gate extension of Gers et al. (2000); forget gate bias is initialized to 1.0 to encourage gradient flow at initialization:

$$\begin{aligned}
i_t &= \sigma(W_{xi} x_t + W_{hi} h_{t-1} + b_i) && \text{(input gate)} \\
f_t &= \sigma(W_{xf} x_t + W_{hf} h_{t-1} + b_f) && \text{(forget gate)} \\
o_t &= \sigma(W_{xo} x_t + W_{ho} h_{t-1} + b_o) && \text{(output gate)} \\
\tilde{c}_t &= \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c) && \text{(candidate)} \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{(cell state)} \\
h_t &= o_t \odot \tanh(c_t) && \text{(hidden state)}
\end{aligned} \tag{2}$$

Three gates (input, forget, output) plus a separate cell state provide fine-grained control over

information flow. The cell state update $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ enables gradient flow through time without weight matrix multiplication (constant error carousel).

GRU. GRU implementation (Cho et al., 2014):

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) && \text{(reset gate)} \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) && \text{(update gate)} \\
\tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) && \text{(candidate)} \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t && \text{(hidden state)} \quad (3)
\end{aligned}
$$

GRU provides a minimal gated architecture with two gates (reset, update) and no separate cell state or output gate, making it a critical test of whether LSTM's compositional generalization advantage requires its full gating machinery or whether the simpler 2-gate mechanism suffices.

Attention models. Pre-norm 2-layer transformer encoder. Each layer applies multi-head self-attention followed by a feed-forward network, both with residual connections and pre-normalization:

$$
\begin{aligned}
\hat{H}^l &= H^{l-1} + \text{MultiHead}(\text{LayerNorm}(H^{l-1})) \\
H^l &= \hat{H}^l + \text{FFN}(\text{LayerNorm}(\hat{H}^l)) \qquad (4)
\end{aligned}
$$

where $\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$ and each head computes $\text{head}_i = \text{softmax}(Q_i K_i^\top / \sqrt{d_k})V_i$ with $Q_i = XW_i^Q$, $K_i = XW_i^K$, $V_i = XW_i^V$ (no bias in Q/K/V projections). The FFN is $\text{FFN}(x) = \text{Dropout}(\text{GELU}(xW_1 + b_1))W_2 + b_2$ with $4\times$ expansion. AbsPE adds sinusoidal positional encoding (Vaswani et al., 2017) to the input embeddings; RoPE applies rotary positional encoding (Su et al., 2021) to the query and key vectors within each attention head. For the pre-norm models used here, a final LayerNorm is applied after the last encoder layer, so the model computes $\sigma(\text{LayerNorm}(h_t^L)W_{\text{out}} + b_{\text{out}})$ at each position t; the training and evaluation pipeline then selects only the final position $t = n$ (no CLS token). Dropout is set to 0.01 for these attention models and is applied to attention weights, both residual branches, and between the two FFN linear layers.

A.2 Optimization

Optimizer. AdamW (Loshchilov and Hutter, 2019) with:

- Weight decay: 1×10^{-5}
- $\beta_1 = 0.9$, $\beta_2 = 0.999$
- $\epsilon = 1 \times 10^{-8}$

Loss function. Mean squared error (MSE) on the full target vector:

$$
\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{z}_i - \mathbf{a}_i)^2 \qquad (5)
$$

where $n = 150$ (without entities) or $n = 300$ (with entities), $\mathbf{z}$ is the network output after sigmoid, and $\mathbf{a}$ is the competitive-layer-based target vector.

Gradient clipping. Max. gradient norm: 3.0

A.3 Learning Rate Schedule

We use a three-phase adaptive learning rate schedule.

Phase 1: Warmup. Linear ramp from minimum LR to maximum LR over warmup epochs, implemented via PyTorch's `OneCycleLR`. Early exit if loss exceeds $1.5\times$ best loss (prevents divergence).

Phase 2: Plateau. Monitor training loss; reduce LR by factor 0.7 when no improvement for patience epochs. Maximum 3 reductions before transitioning to cyclical.

Phase 3: Cyclical. `CyclicLR` with `triangular2` mode (diminishing amplitude). Returns to plateau phase if improvement detected.

Architecture	Max LR	Warmup Epochs	Plateau Patience
SRN	1×10^{-2}	1	2
GRU	7×10^{-3}	1	2
LSTM	7×10^{-3}	1	2
Attention (AbsPE)	5×10^{-3}	2	4
Attention (RoPE)	5×10^{-3}	2	4

Table 4: Architecture-specific learning rate hyperparameters. Minimum LR is 1×10^{-4} for all architectures.

A.4 Weight Initialization

All architectures:

- Embedding weights: Uniform(-0.15, $+0.15$)
- Output layer weights: Uniform(-0.15, $+0.15$)
- Output layer bias: zeros

Recurrent-specific:

- Internal linear weights: Uniform(-0.15, $+0.15$), following Frank et al. (2009)
- Internal linear biases: zeros

LSTM-specific:

- Forget gate bias: initialized to 1.0 (all other gate biases: 0.0)

Attention-specific:

- Q/K/V/O projection weights: Xavier Uniform with gain $1/\sqrt{2}$
- FFN linear weights: Xavier Uniform

- Attention and FFN biases: zeros
- LayerNorm: weight=1.0, bias=0.0
- Positional encoding (AbsPE): sinusoidal, not learned
- Positional encoding (RoPE): precomputed up to max length and stored as cosine/sine buffers

A.5 Training Configuration

Parameter	Value
Batch size	64
Maximum epochs	750
Early stopping	None (all models train to completion)
Checkpoint criterion	Best validation comprehension score
Random seeds	5 per configuration
Dataset splits	2 (complementary train/test)
Experiment 1 models	100 (5 architectures $\times$ 2 entity $\times$ 2 splits $\times$ 5 seeds)
Experiment 2 models	40 (2 architectures $\times$ 2 entity $\times$ 2 splits $\times$ 5 seeds)
Total models	140

Table 5: Training config shared across all runs.

A.6 Evaluation Configuration

- MSE loss: computed on **full** target vector (150 or 300 dimensions)
- Comprehension score: computed on **first 150 dimensions only** (truth-conditional part)

Checkpointing: Best model saved based on validation comprehension score.

A.7 Dataset Configuration

Dataset	Total Rows	Consistent	Toy Sentences	Non-Toy
Train split 1	9,443	6,279	395 (6.3%)	5,884
Train split 2	9,443	6,789	399 (5.9%)	6,390
Test split 1	4,113	2,840	70 (2.5%)	2,770
Test split 2	4,113	2,330	66 (2.8%)	2,264

Table 6: Dataset statistics. Only consistent sentences (satisfying hard constraints) are used.

This table summarizes the full corpora used in the experiments; the paper repository includes only the systematicity subset used for final evaluation.

Toy sentence repetition. Toy sentences are repeated $4\times$ per epoch to balance the rare toy events (6% of training data). Effective samples per epoch:

- Split 1: $5884 + (4 \times 395) = 7{,}464$
- Split 2: $6390 + (4 \times 399) = 7{,}986$

A.8 Vocabulary and Synonym Pairs

The vocabulary of our feature-based context-free grammar consists of 40 words, indices 0–39. Padding index: 40 (outside vocabulary).

Four (perfect) synonym pairs enable systematicity testing via lexical substitution:

Category	Words	Count
Persons	charlie, heidi, sophia, someone, boy, girl	6
Games	chess, hide-and-seek, soccer, football, game	5
Toys	puzzle, jigsaw, ball, doll, toy	5
Locations	bathroom, shower, bedroom, street, playground	5
Verbs (active)	plays, wins, loses, beats	4
Verbs (passive)	is, played, won, lost	4
Manner adverbs	well, badly, ease, difficulty	4
Prepositions	with, to, at, in, by	5
Place adverbs	inside, outside	2
Total		**40**

Table 7: Vocabulary organized by category (for convenience; these categories play no role in our framework). Padding index is 40.

Synonym	'Canonical' Term
boy	charlie
football	soccer
jigsaw	puzzle
shower	bathroom

A.9 Feature-Based Context-Free Grammar

The grammar uses two features for agreement:

- **NTYPE**: Noun type $\in$ {person, game, toy}
- **VTYPE**: Verb type $\in$ {play, win, lose}

Start Symbol and Sentence Structure

```
S[NTYPE=?n, VTYPE=?v] -> NP[NTYPE=?n] VP[NTYPE=?n, VTYPE=?v]
                         APP[NTYPE=?n, VTYPE=?v]
```

Noun Phrases

```
# Person NPs
NP[NTYPE=person] -> 'charlie' | 'heidi' | 'sophia'
NP[NTYPE=person] -> 'someone' | 'boy' | 'girl'

# Game NPs
NP[NTYPE=game] -> 'chess' | 'hide-and-seek' | 'soccer'
NP[NTYPE=game] -> 'football' | 'game'

# Toy NPs
NP[NTYPE=toy] -> 'puzzle' | 'jigsaw' | 'ball' | 'doll' | 'toy'
```

VPs (Active Voice, Person Subject)

```
# Playing
VP[NTYPE=person, VTYPE=play] -> 'plays'

# Winning
VP[NTYPE=person, VTYPE=win] -> 'wins'
VP[NTYPE=person, VTYPE=win] -> 'beats' NP[NTYPE=person]

# Losing
VP[NTYPE=person, VTYPE=lose] -> 'loses'
VP[NTYPE=person, VTYPE=lose] -> 'loses' 'to' NP[NTYPE=person]
```

VPs (Passive Voice, Game Subject)

```
VP[NTYPE=game, VTYPE=play] -> 'is' 'played'
VP[NTYPE=game, VTYPE=win]  -> 'is' 'won'
VP[NTYPE=game, VTYPE=lose] -> 'is' 'lost'
```

VPs (Passive Voice, Toy Subject)

```
VP[NTYPE=toy, VTYPE=play] -> 'is' 'played' 'with'
```

Prepositional Phrases

```
# By-phrase (passive agent)
PPperson -> 'by' NP[NTYPE=person]

# At-phrase (game specification)
PPgame -> 'at' NP[NTYPE=game]

# With-phrase (toy specification)
PPtoy -> 'with' NP[NTYPE=toy]

# In-phrase (location)
```

```
PPplace -> 'in' Location
Location -> 'bathroom' | 'shower' | 'bedroom'
          | 'street' | 'playground'

# Place (location or inside/outside)
Place -> 'inside' | 'outside' | PPplace

# Manner adverbs
Manner -> 'well' | 'badly'

# PP manner (for winning)
PPmanner -> 'with' 'ease' | 'with' 'difficulty'
```

Optional Modifiers

The APP non-terminal generates optional modifiers. Different subject/verb combinations allow different modifier patterns:

```
# Person playing a game: game + manner + place (all combinations)
APP[NTYPE=person, VTYPE=play] -> NP[NTYPE=game] Manner Place
APP[NTYPE=person, VTYPE=play] -> NP[NTYPE=game] Manner
APP[NTYPE=person, VTYPE=play] -> NP[NTYPE=game] Place
APP[NTYPE=person, VTYPE=play] -> NP[NTYPE=game]
APP[NTYPE=person, VTYPE=play] -> Manner Place
APP[NTYPE=person, VTYPE=play] -> Manner
APP[NTYPE=person, VTYPE=play] -> Place
APP[NTYPE=person, VTYPE=play] -> PPtoy Place
APP[NTYPE=person, VTYPE=play] -> Place PPtoy
APP[NTYPE=person, VTYPE=play] -> PPtoy
APP[NTYPE=person, VTYPE=play] ->

# Person winning: PPmanner + PPgame + Place (various orders)
APP[NTYPE=person, VTYPE=win] -> PPmanner PPgame Place
APP[NTYPE=person, VTYPE=win] -> PPmanner PPgame
APP[NTYPE=person, VTYPE=win] -> PPgame PPmanner
APP[NTYPE=person, VTYPE=win] -> PPmanner Place
APP[NTYPE=person, VTYPE=win] -> PPgame Place
APP[NTYPE=person, VTYPE=win] -> Place PPgame
APP[NTYPE=person, VTYPE=win] -> PPmanner
APP[NTYPE=person, VTYPE=win] -> PPgame
APP[NTYPE=person, VTYPE=win] -> Place
APP[NTYPE=person, VTYPE=win] ->

# Person losing: PPgame + Place only (NO PPmanner allowed)
APP[NTYPE=person, VTYPE=lose] -> PPgame Place
APP[NTYPE=person, VTYPE=lose] -> Place PPgame
APP[NTYPE=person, VTYPE=lose] -> PPgame
APP[NTYPE=person, VTYPE=lose] -> Place
APP[NTYPE=person, VTYPE=lose] ->

# Game passive (is played): PPperson + Manner + Place
APP[NTYPE=game, VTYPE=play] -> Manner PPperson Place
APP[NTYPE=game, VTYPE=play] -> PPperson Manner Place
APP[NTYPE=game, VTYPE=play] -> Manner PPperson
APP[NTYPE=game, VTYPE=play] -> PPperson Manner
APP[NTYPE=game, VTYPE=play] -> Manner Place
APP[NTYPE=game, VTYPE=play] -> PPperson Place
APP[NTYPE=game, VTYPE=play] -> Place PPperson
APP[NTYPE=game, VTYPE=play] -> Manner
APP[NTYPE=game, VTYPE=play] -> PPperson
APP[NTYPE=game, VTYPE=play] -> Place
APP[NTYPE=game, VTYPE=play] ->

# Game passive (is won): PPmanner + PPperson + Place
APP[NTYPE=game, VTYPE=win] -> PPmanner PPperson Place
APP[NTYPE=game, VTYPE=win] -> PPperson PPmanner Place
APP[NTYPE=game, VTYPE=win] -> PPmanner PPperson
APP[NTYPE=game, VTYPE=win] -> PPperson PPmanner
APP[NTYPE=game, VTYPE=win] -> PPmanner Place
APP[NTYPE=game, VTYPE=win] -> PPperson Place
APP[NTYPE=game, VTYPE=win] -> Place PPperson
APP[NTYPE=game, VTYPE=win] -> PPmanner
APP[NTYPE=game, VTYPE=win] -> PPperson
APP[NTYPE=game, VTYPE=win] -> Place
APP[NTYPE=game, VTYPE=win] ->

# Game passive (is lost): PPperson + Place only
APP[NTYPE=game, VTYPE=lose] -> PPperson Place
APP[NTYPE=game, VTYPE=lose] -> Place PPperson
APP[NTYPE=game, VTYPE=lose] -> PPperson
APP[NTYPE=game, VTYPE=lose] -> Place
APP[NTYPE=game, VTYPE=lose] ->

# Toy passive (is played with): PPperson + Place only (NO manner)
APP[NTYPE=toy, VTYPE=play] -> PPperson Place
APP[NTYPE=toy, VTYPE=play] -> Place PPperson
APP[NTYPE=toy, VTYPE=play] -> PPperson
APP[NTYPE=toy, VTYPE=play] -> Place
APP[NTYPE=toy, VTYPE=play] ->
```

Construction	Manner (well/badly)	PPmanner (with ease/diff.)	Location
person plays game	✓	×	✓
person plays with toy	×	×	✓
person wins	×	✓	✓
person beats X	×	✓	✓
person loses	×	×	✓
person loses to X	×	×	✓

Table 8: Grammaticality of modifier combinations by construction type.

A.10 Existential Quantifiers

Existential quantifiers expand to disjunctions:

Term	Interpretation (Disjunction)
someone	charlie ∨ heidi ∨ sophia
girl	heidi ∨ sophia
game	chess ∨ hide_n_seek ∨ soccer
toy	puzzle ∨ ball ∨ doll
inside	bathroom ∨ bedroom
outside	playground ∨ street

Table 9: Expansion of quantificational terms to disjunctions over their denotations.

A.11 Systematicity Test Sentences

Test Group	Total	OG	Added	Consistent (per split)
Word	12	4	8	12 (all)
Sentence	160	10	150	160 (all)
Complex	168	56	112	96–144
Basic	150	10	140	58–66
Total	490	80	410	326–382

Table 10: Systematicity test sentence counts. Consistent counts vary by split due to game-location constraint filtering.

A.12 Pool MCMC and Competitive Neural Model for Truth-Conditional Representations

Brasoveanu (2026) describes the full Pool MCMC sampling methodology, competitive neural model and training procedure assumed by the experiments in this paper.

B Training Trajectories for Experiment 1 Models

This appendix presents training trajectories for the ten Experiment 1 model configurations (5 architectures × 2 entity settings). All models were trained for 750 epochs. Most reach 95% of their eventual best validation score within roughly 100 epochs, though small later improvements can push the selected best-score checkpoint substantially later.

Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

The test-group curves in these training figures include all sentences in each group, not just the ∼350 sentences selected for the model evaluations. These ∼350 sentences were approximately length-matched across Sentence, Complex, and Basic (the Word group is shorter on average), and excluded existential quantifiers like *girl*, *someone*, *toy* and *game*.

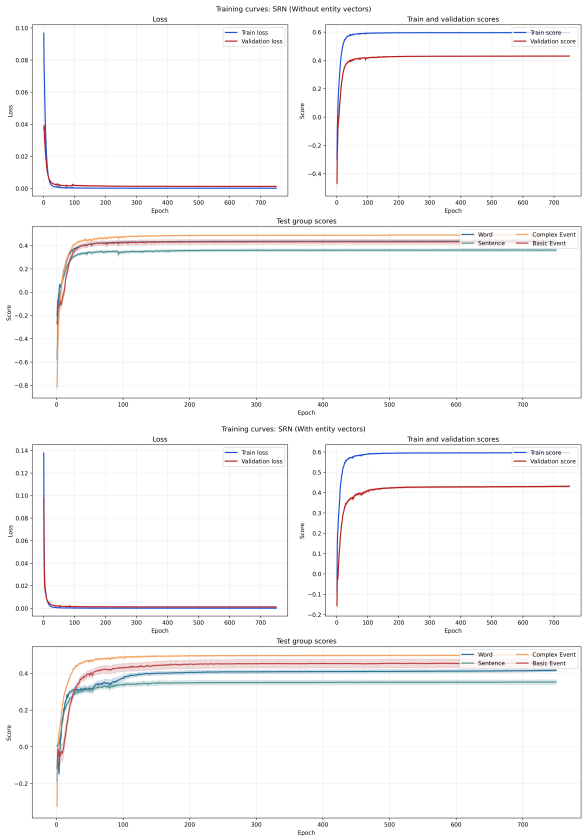

Figure 3: SRN Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

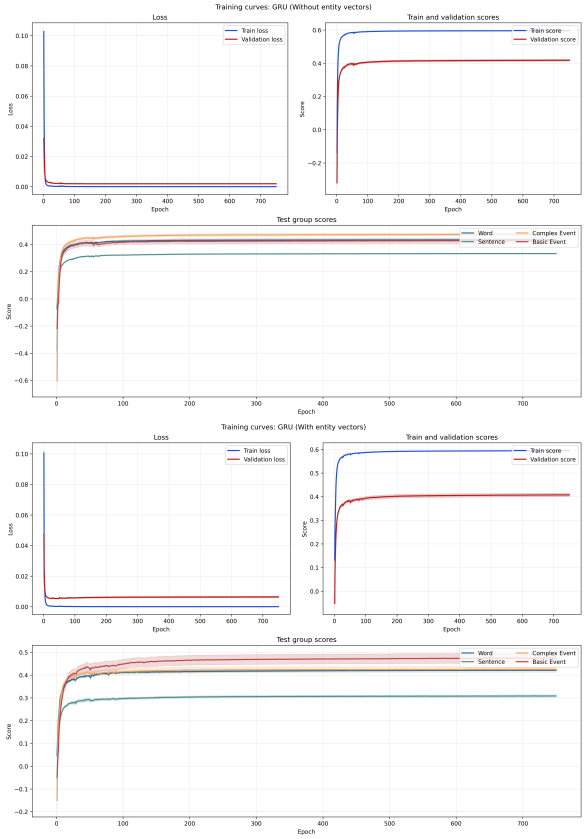

Figure 4: GRU Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

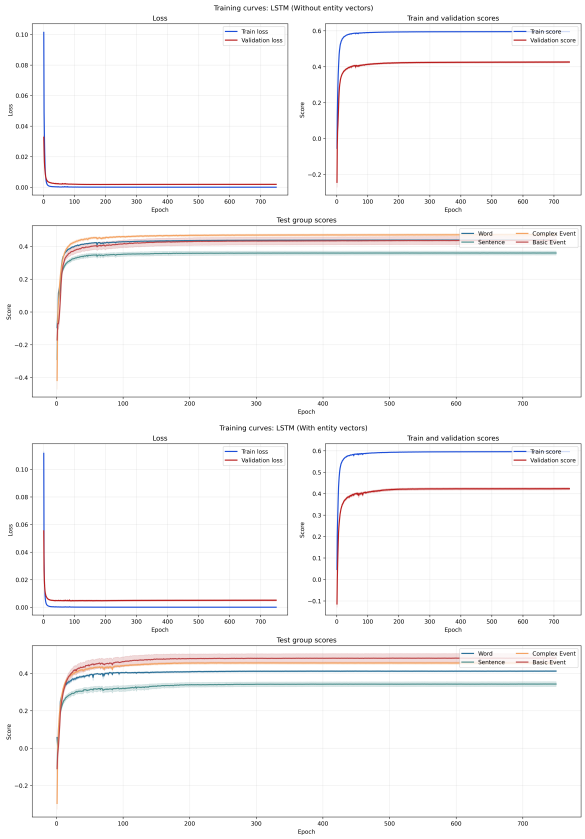

Figure 5: LSTM Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

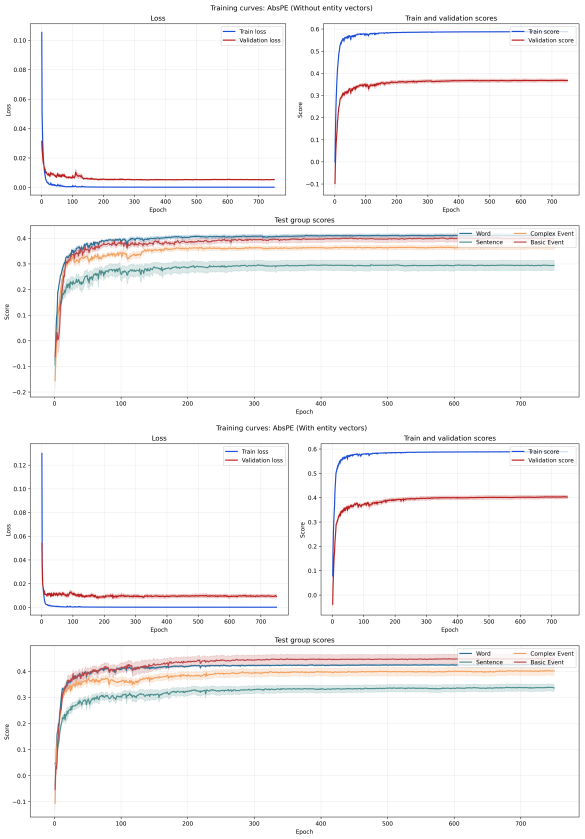

Figure 6: Attention (AbsPE, H48) Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

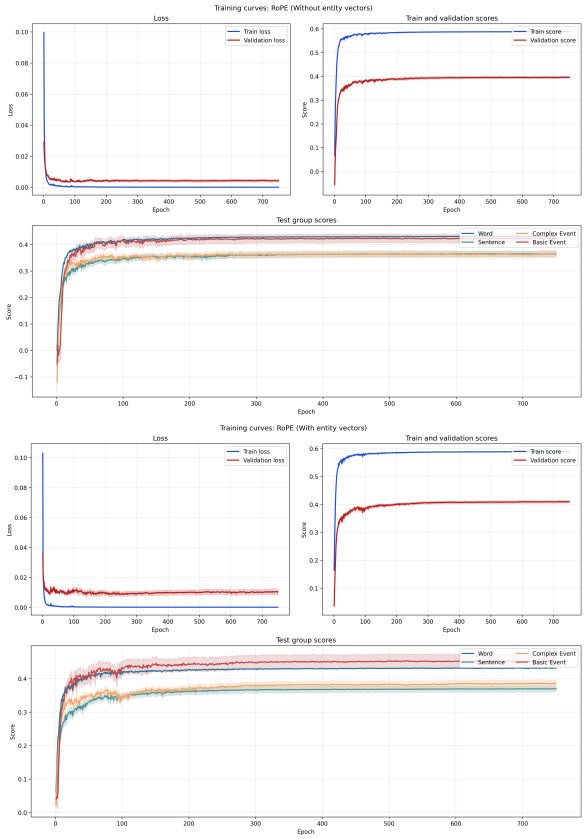

Figure 7: Attention (RoPE, H48) Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds $\times$ 2 train/test splits.

C Training Trajectories for High-Capacity Attention Models

This appendix presents training trajectories for the two Experiment 2 high-capacity Attention models (hidden dimension 80), matching the hidden dimension of the LSTM models.

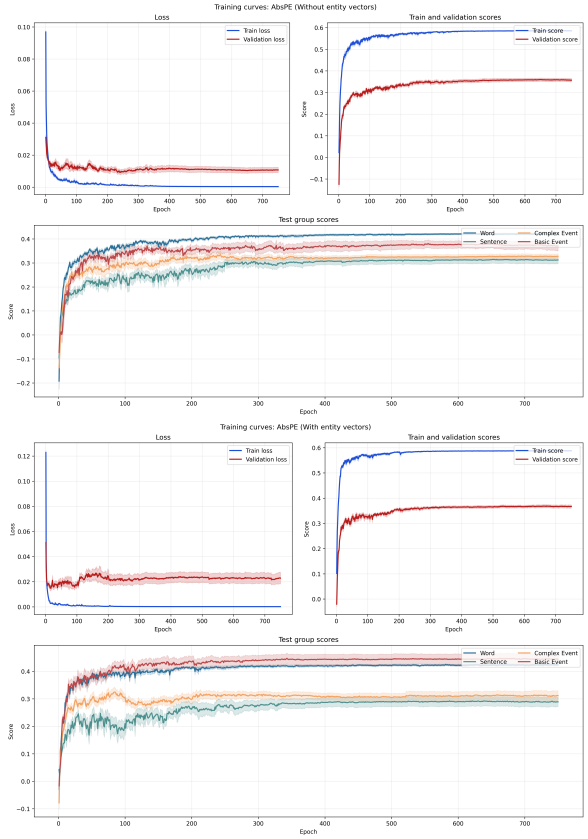

Figure 8: Attention (AbsPE, H80) Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

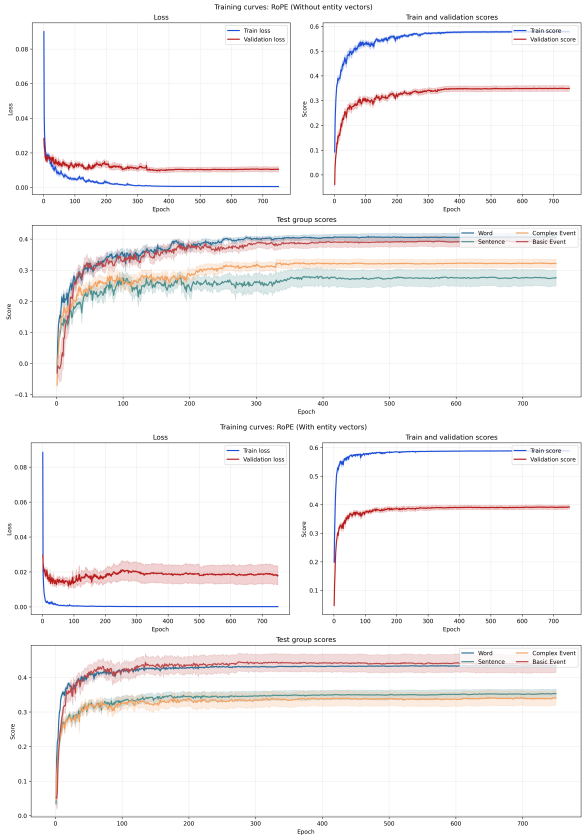

Figure 9: Attention (RoPE, H80) Training Trajectories. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Shaded regions indicate standard errors across 5 random seeds × 2 train/test splits.

D Compositional Systematicity Breakdown

Figures 10 and 11 together provide detailed per-architecture breakdowns for the five Experiment 1 architectures, while Figure 12 provides the corresponding breakdown for the two Experiment 2 high-capacity Attention models. These figures complement the aggregate results in Table 2 and Observation 1 in Section 4. Within each subplot, bars are disaggregated by modifier type (Canonical, +Location, +Manner, +Location+Manner where applicable), making the modifier-driven difficulty gradient visible at the level of individual architectures. The systematic reduction in described scores with increasing modifier complexity is consistent across all architectures and entity settings, confirming that the two-dimensional difficulty structure is a property of the task, not of any particular architecture.

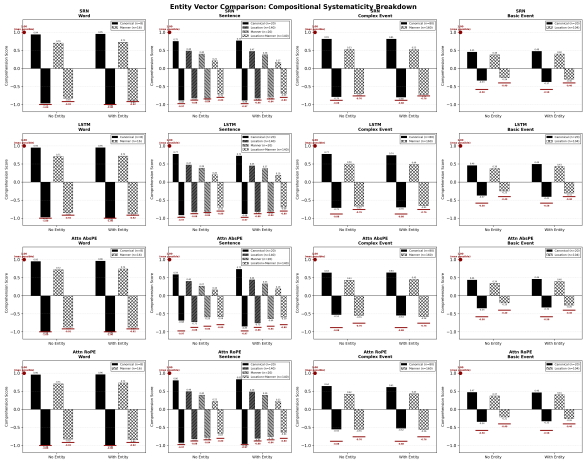

Figure 10: Compositional systematicity breakdown for four of the five Experiment 1 architectures (SRN, LSTM, Attn AbsPE, Attn RoPE). Each subplot shows described (left group) and competing (right group) comprehension scores for one architecture × test group combination, disaggregated by modifier complexity level. Hatched bars: without entity vectors. Solid bars: with entity vectors. Dark red lines: theoretical max/min for a perfect model.

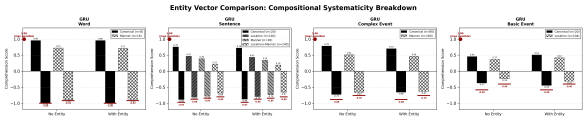

Figure 11: Compositional systematicity breakdown for the fifth Experiment 1 architecture, GRU. Same format as Figure 10. GRU shows the same modifier-driven difficulty gradient as all other architectures, with a pattern closely matching LSTM across all test groups and modifier levels.

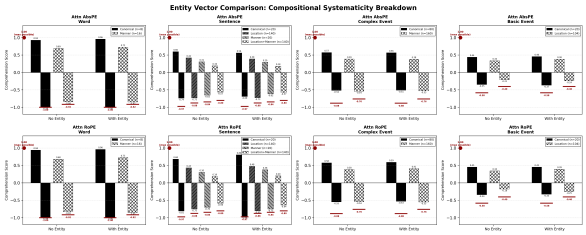

Figure 12: Compositional systematicity breakdown for the high-capacity Attention models (H80). Same format as Figure 10. The modifier-driven difficulty gradient is present even with higher-capacity attention models, confirming that the two-dimensional difficulty structure is not an artifact of limited model capacity.

E Per-Modifier Disaggregation

Table 11 disaggregates Word and Sentence test-only advantage scores by modifier type. On Sentence, the attention deficit is already visible on *canonical* sentences: without entity vectors, AbsPE scores 1.27 vs. SRN 1.63 and LSTM 1.69, while RoPE reaches 1.72. Table 12 then shows the corresponding Complex and Basic disaggregation.

Table 12 disaggregates Complex and Basic test-only advantage scores by modifier type. SRN's Complex advantage over gated architectures (Observation 3) is uniform across both modifier levels (Canonical and +Manner), confirming it is a genuine architectural effect rather than an artifact of one modifier type. For Basic, gated architectures show the largest entity-vector benefit on Canonical sentences: GRU improves from 0.83 to 0.95 (+0.12), LSTM from 0.83 to 0.90 (+0.07). On Location-modified sentences, entity-vector gains remain positive for all architectures, ranging from +0.06 to +0.13.

		Word		Sentence			
Arch		**Can**	**+Man**	**Can**	**+Loc**	**+Man**	**+L+M**
SRN	−ent	1.90	1.53	1.63	1.31	1.22	0.93
	+ent	1.94	1.62	1.65	1.30	1.17	0.86
GRU	−ent	1.95	1.61	1.64	1.28	1.17	0.91
	+ent	1.95	1.60	1.60	1.23	1.08	0.86
LSTM	−ent	1.92	1.56	1.69	1.30	1.20	0.90
	+ent	1.94	1.59	1.60	1.27	1.19	0.88
AbsPE	−ent	1.94	1.59	1.27	1.13	0.86	0.75
	+ent	1.95	1.61	1.58	1.19	0.97	0.78
RoPE	−ent	1.95	1.53	1.72	1.31	1.18	0.89
	+ent	1.95	1.57	1.78	1.29	1.15	0.85

Table 11: Test-only advantage scores disaggregated by modifier type and ±entity condition (10 models per cell) for Word and Sentence. **Can**: canonical (unmodified). **+Loc/+Man/+L+M**: sentences with location, manner, or both modifiers. The canonical Sentence columns back Observation 2: the AbsPE deficit is already present before any modifier composition is added.

		Complex			Basic		
Arch		**Can**	**+Man**	**Agg**	**Can**	**+Loc**	**Agg**
SRN	−ent	1.60	1.22	1.35	0.79	0.64	0.66
	+ent	1.61	1.25	1.37	0.85	0.70	0.73
LSTM	−ent	1.48	1.17	1.27	0.83	0.63	0.66
	+ent	1.44	1.14	1.24	0.90	0.74	0.76
GRU	−ent	1.51	1.18	1.29	0.83	0.61	0.64
	+ent	1.36	1.10	1.18	0.95	0.73	0.77
AbsPE	−ent	1.16	0.98	1.04	0.78	0.54	0.58
	+ent	1.18	1.03	1.08	0.78	0.66	0.68
RoPE	−ent	1.19	0.98	1.05	0.81	0.59	0.62
	+ent	1.14	0.99	1.04	0.79	0.67	0.68
AbsPE_H80	−ent	1.09	0.92	0.98	0.79	0.55	0.59
	+ent	1.08	0.91	0.97	0.82	0.63	0.66
RoPE_H80	−ent	1.12	0.92	0.99	0.79	0.52	0.57
	+ent	1.12	0.96	1.01	0.78	0.65	0.67

Table 12: Test-only advantage scores disaggregated by modifier type and ±entity condition (10 models per cell). **Can**: canonical (unmodified). **+Man/+Loc**: with manner/location modifiers. **Agg**: aggregate. SRN's Complex advantage over LSTM/GRU is present at both modifier levels and both entity conditions, confirming a genuine architectural effect. On Basic, entity vectors provide the largest benefit for GRU on Canonical sentences (+ent: 0.95 vs. −ent: 0.83).

F Generalization Gap (Train vs. Test) for Experiment 1 Models

This appendix provides detailed plots of the generalization gap between training and test performance for the ten Experiment 1 model configurations (5 architectures $\times$ 2 entity settings). Note that unlike the plots in Section 4, the solid bars in these plots show training scores, and the hatched bars show test scores. The $\pm$entity contrast is **not** encoded by bar type here; instead, separate plots are provided for each entity setting: top panels without entity vectors and bottom panels with entity vectors. The generalization gap can be observed by comparing the solid (training) and hatched (test) bars within each subplot. The largest generalization gap is generally in the most challenging Basic group, as reported in Section 4, but different architectures exhibit varying degrees of overfitting across the test groups.

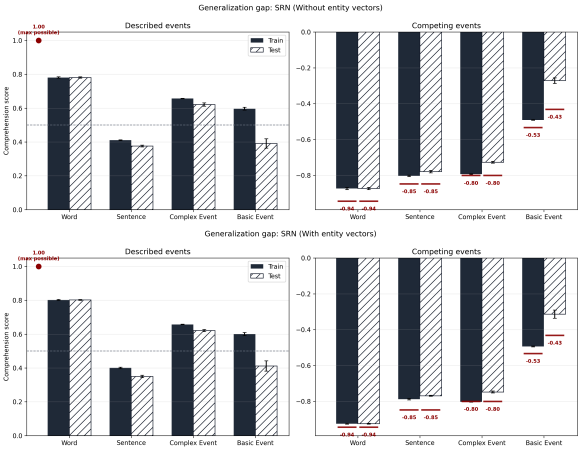

Figure 13: SRN Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds $\times$ 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

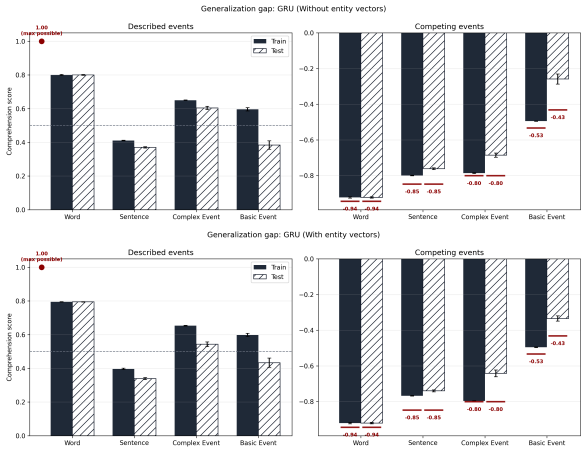

Figure 14: GRU Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds $\times$ 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

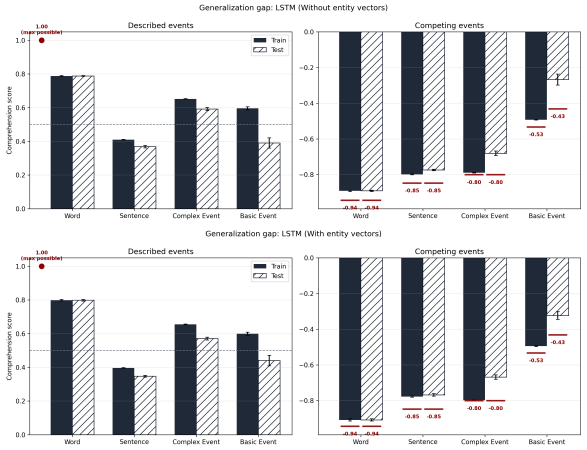

Figure 15: LSTM Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds $\times$ 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

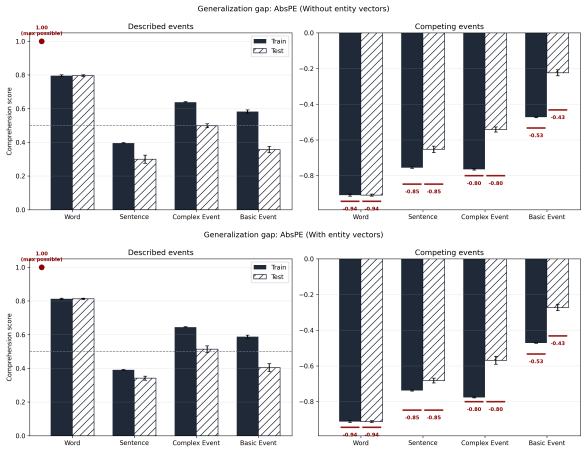

Figure 16: Attention (AbsPE, H48) Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds × 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

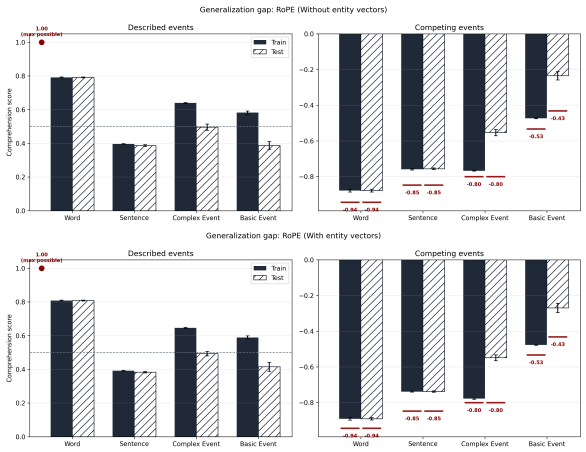

Figure 17: Attention (RoPE, H48) Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds × 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

G Generalization Gap (Train vs. Test) for High-Capacity Attention Models

This appendix provides detailed plots of the generalization gap between training and test performance for the two Experiment 2 high-capacity Attention models (hidden dimension 80), matching the hidden dimension of the LSTM models. Note that unlike the plots in Section 4, the solid bars in these plots show training scores, and the hatched bars show test scores. The ±entity contrast is **not** encoded by bar type here; instead, separate plots are provided for each entity setting: top panels without entity vectors and bottom panels with entity vectors. The generalization gap can be observed by comparing the solid (training) and hatched (test) bars within each subplot.

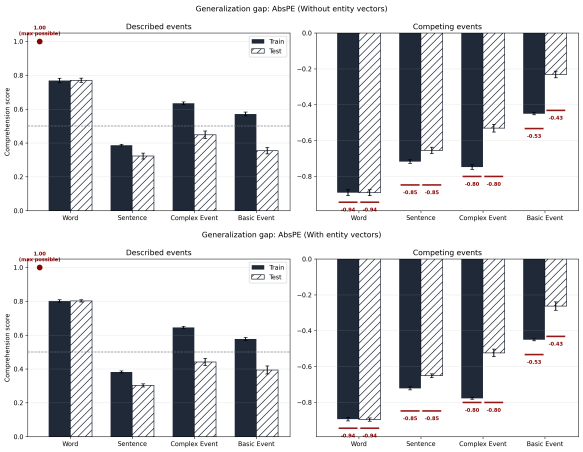

Figure 18: Attention (AbsPE, H80) Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds × 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

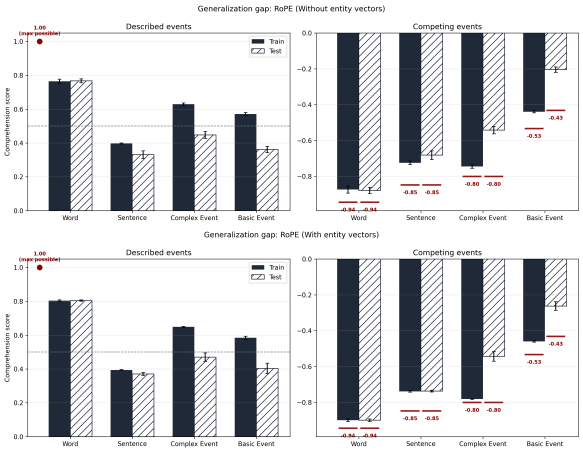

Figure 19: Attention (RoPE, H80) Generalization Gap. Top panels: Without Entity Vectors. Bottom panels: With Entity Vectors. Described (Left) vs. Competing (Right) Scores, Training (Solid bars) vs. Test (Hatched bars). Whiskers: Standard errors across 5 random seeds × 2 train/test splits. Dark red lines: theoretical max/min for a perfect model.

H Study Design and Data Description

This appendix documents the structure of the experimental data: observation counts across test groups and splits and the distribution of competing events per sentence.

Group	Split	Train sent.	Test sent.	Models
Word	S1	12	12	70
	S2	12	12	70
Sentence	S1	160	160	70
	S2	160	160	70
Complex	S1	96	144	70
	S2	144	96	70
Basic	S1	126	66	70
	S2	134	58	70

Table 13: Observation structure per group and split. Total observations per cell = sentences × models. Grand total: 108,640 observations. The asymmetry in Complex and Basic sentence counts arises from the complementary exclusion structure.

Category	N	Mean	SD	Median	Min	Max
Split 1						
Word	24	2.7	0.5	3	2	3
Sentence	320	7.0	1.4	7	3	8
Complex	240	2.9	1.0	3	1	5
Basic	192	2.3	0.9	2	1	4
All	776	4.4	2.4	3	1	8
Split 2						
Word	24	2.7	0.5	3	2	3
Sentence	320	7.0	1.4	7	3	8
Complex	240	2.9	1.0	3	1	5
Basic	192	2.3	0.9	2	1	4
All	776	4.4	2.4	3	1	8

Table 14: Number of competing events per sentence by test category and data split, pooled across train and test within each split. Word/Sentence/Complex/Basic: systematicity test groups with increasing compositional difficulty. Competitors are produced by substituting one semantic slot at a time in the described event(s).

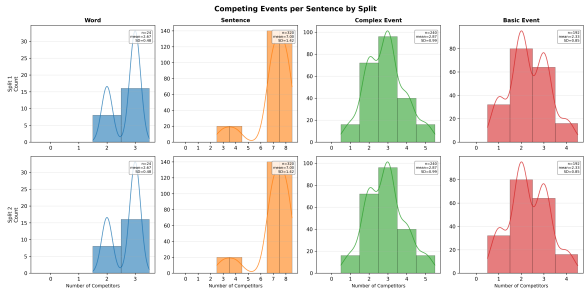

Figure 20: Distribution of competing events per sentence across test groups (histogram).

Category	Phase	N	Mean	SD	Median	Min	Max
Split 1							
Word	Train	12	2.7	0.5	3	2	3
	Test	12	2.7	0.5	3	2	3
Sentence	Train	160	7.0	1.4	7	3	8
	Test	160	7.0	1.4	7	3	8
Complex	Train	96	3.4	1.0	3	2	5
	Test	144	2.5	0.8	2	1	4
Basic	Train	126	2.4	0.9	2	1	4
	Test	66	2.2	0.7	2	1	4
Split 2							
Word	Train	12	2.7	0.5	3	2	3
	Test	12	2.7	0.5	3	2	3
Sentence	Train	160	7.0	1.4	7	3	8
	Test	160	7.0	1.4	7	3	8
Complex	Train	144	2.5	0.8	2	1	4
	Test	96	3.4	1.0	3	2	5
Basic	Train	134	2.4	0.8	2	1	4
	Test	58	2.1	0.9	2	1	4

Table 15: Number of competing events per sentence disaggregated by train/test phase and data split. This shows how the competitor distribution differs between sentences used for training and those reserved for systematicity testing.

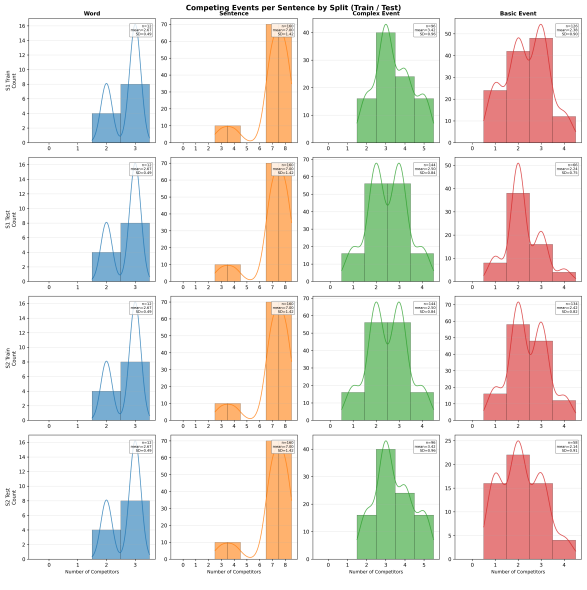

Figure 21: Distribution of competing events per sentence across test groups, broken down by train/test split (histogram).

I Descriptive Statistics

This appendix provides descriptive statistics for the comprehension advantage scores. Table 16 shows model-level means (averaging over sentences within each model first), complementing the sentence-level means in Table 1. The tables below give pooled train and test breakdowns, and the histograms summarize the sentence-level distributions. Detailed per-split and distributional descriptive summaries are available in the repository's `statistical_analysis/` directory.

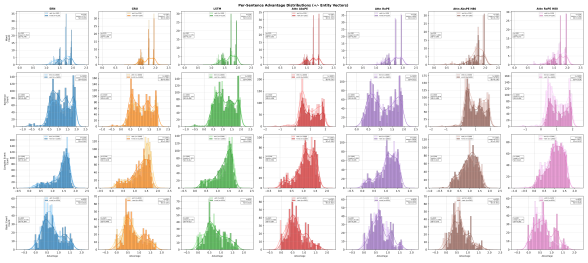

Figure 22: Comprehension advantage score distributions for test sentences (histogram).

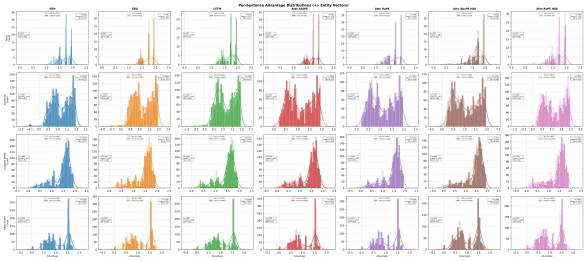

Figure 23: Comprehension advantage score distributions for training sentences (histogram).

		Word	Sent	Cmplx	Basic
SRN	−ent	1.65	1.16	1.35	0.66
	+ent	1.73	1.12	1.37	0.73
GRU	−ent	1.72	1.13	1.29	0.64
	+ent	1.72	1.08	1.18	0.77
LSTM	−ent	1.68	1.14	1.27	0.66
	+ent	1.71	1.12	1.24	0.76
Attn AbsPE	−ent	1.71	0.95	1.04	0.58
	+ent	1.72	1.02	1.08	0.68
Attn RoPE	−ent	1.67	1.14	1.05	0.62
	+ent	1.70	1.12	1.04	0.68

Table 16: Test-only advantage scores (described − competing) by architecture, test group and ±entity. Averaged over 10 models per architecture/entity condition. Higher is better.

		Sentence-level			Model-level			
Group	**Architecture**	−ent	+ent	Δ	−ent	+ent	SE_+	Δ
Word	SRN	1.654	1.727	+0.073	1.654	1.727	0.004	+0.073
	GRU	1.723	1.715	-0.007	1.723	1.715	0.005	-0.007
	LSTM	1.679	1.707	+0.027	1.679	1.707	0.010	+0.027
	Attn AbsPE	1.705	1.725	+0.019	1.705	1.725	0.008	+0.019
	Attn RoPE	1.670	1.699	+0.030	1.670	1.699	0.009	+0.030
	Attn AbsPE H80	1.661	1.699	+0.038	1.661	1.699	0.015	+0.038
	Attn RoPE H80	1.646	1.704	+0.058	1.646	1.704	0.009	+0.058
Sentence	SRN	1.156	1.119	-0.037	1.156	1.119	0.010	-0.037
	GRU	1.132	1.080	-0.052	1.132	1.080	0.009	-0.052
	LSTM	1.143	1.115	-0.027	1.143	1.115	0.008	-0.027
	Attn AbsPE	0.953	1.024	+0.071	0.953	1.024	0.025	+0.071
	Attn RoPE	1.143	1.121	-0.022	1.143	1.121	0.007	-0.022
	Attn AbsPE H80	0.978	0.954	-0.024	0.978	0.954	0.019	-0.024
	Attn RoPE H80	1.013	1.108	+0.095	1.013	1.108	0.012	+0.095
Complex	SRN	1.357	1.373	+0.017	1.349	1.369	0.012	+0.020
	GRU	1.300	1.196	-0.104	1.290	1.185	0.032	-0.105
	LSTM	1.281	1.244	-0.037	1.273	1.239	0.020	-0.034
	Attn AbsPE	1.054	1.099	+0.046	1.041	1.082	0.042	+0.042
	Attn RoPE	1.069	1.054	-0.014	1.050	1.042	0.030	-0.008
	Attn AbsPE H80	1.003	0.971	-0.031	0.979	0.965	0.042	-0.014
	Attn RoPE H80	1.009	1.027	+0.018	0.989	1.014	0.051	+0.025
Basic	SRN	0.670	0.735	+0.065	0.662	0.725	0.052	+0.063
	GRU	0.652	0.775	+0.124	0.643	0.768	0.041	+0.125
	LSTM	0.670	0.773	+0.104	0.658	0.764	0.052	+0.106
	Attn AbsPE	0.586	0.685	+0.099	0.581	0.678	0.039	+0.096
	Attn RoPE	0.630	0.694	+0.064	0.622	0.685	0.052	+0.063
	Attn AbsPE H80	0.590	0.665	+0.075	0.586	0.657	0.045	+0.071
	Attn RoPE H80	0.570	0.676	+0.106	0.566	0.666	0.054	+0.101

Table 17: Pooled descriptive statistics (test sentences, both splits combined). Sentence-level: mean weighted by observations (splits with more test sentences count more). Model-level: mean weighted equally per model (each of 10 models counts once). SE_+: standard error for the +ent condition across 10 models (5 seeds × 2 splits). Δ: +ent − −ent.

		Sentence-level			Model-level			
Group	**Architecture**	−ent	+ent	Δ	−ent	+ent	SE_+	Δ
Word	SRN	1.654	1.727	+0.073	1.654	1.727	0.004	+0.073
	GRU	1.723	1.715	-0.007	1.723	1.715	0.005	-0.007
	LSTM	1.679	1.707	+0.027	1.679	1.707	0.010	+0.027
	Attn AbsPE	1.705	1.725	+0.019	1.705	1.725	0.008	+0.019
	Attn RoPE	1.670	1.699	+0.030	1.670	1.699	0.009	+0.030
	Attn AbsPE H80	1.660	1.696	+0.036	1.660	1.696	0.017	+0.036
	Attn RoPE H80	1.638	1.704	+0.066	1.638	1.704	0.009	+0.066
Sentence	SRN	1.214	1.189	-0.026	1.214	1.189	0.007	-0.026
	GRU	1.211	1.164	-0.046	1.211	1.164	0.004	-0.046
	LSTM	1.208	1.174	-0.034	1.208	1.174	0.005	-0.034
	Attn AbsPE	1.152	1.129	-0.023	1.152	1.129	0.005	-0.023
	Attn RoPE	1.156	1.131	-0.025	1.156	1.131	0.004	-0.025
	Attn AbsPE H80	1.105	1.104	-0.000	1.105	1.104	0.013	-0.000
	Attn RoPE H80	1.122	1.133	+0.011	1.122	1.133	0.006	+0.011
Complex	SRN	1.450	1.461	+0.011	1.450	1.460	0.002	+0.010
	GRU	1.438	1.451	+0.013	1.436	1.449	0.003	+0.013
	LSTM	1.443	1.454	+0.011	1.442	1.453	0.003	+0.011
	Attn AbsPE	1.406	1.423	+0.017	1.404	1.421	0.005	+0.017
	Attn RoPE	1.408	1.428	+0.020	1.407	1.424	0.008	+0.017
	Attn AbsPE H80	1.389	1.428	+0.039	1.383	1.423	0.011	+0.040
	Attn RoPE H80	1.380	1.432	+0.052	1.374	1.430	0.004	+0.056
Basic	SRN	1.088	1.095	+0.007	1.087	1.094	0.011	+0.007
	GRU	1.092	1.095	+0.003	1.091	1.094	0.011	+0.003
	LSTM	1.090	1.095	+0.005	1.089	1.094	0.011	+0.005
	Attn AbsPE	1.057	1.061	+0.004	1.056	1.060	0.009	+0.004
	Attn RoPE	1.057	1.067	+0.011	1.056	1.066	0.011	+0.011
	Attn AbsPE H80	1.023	1.029	+0.006	1.022	1.028	0.013	+0.006
	Attn RoPE H80	1.012	1.045	+0.033	1.011	1.044	0.012	+0.033

Table 18: Pooled descriptive statistics (train sentences, both splits combined). Sentence-level: mean weighted by observations (splits with more train sentences count more). Model-level: mean weighted equally per model (each of 10 models counts once). SE_+: standard error for the +ent condition across 10 models (5 seeds × 2 splits). Δ: +ent − −ent.

J Learning Rate Schedule and Convergence

Figure 24 shows the adaptive learning rate schedules used during training. Figure 25 shows the distribution of best-score epochs across models. Many models reach near-peak validation performance well before the 750-epoch budget, but the selected best checkpoint can occur much later because validation improvements often continue incrementally over long plateaus.

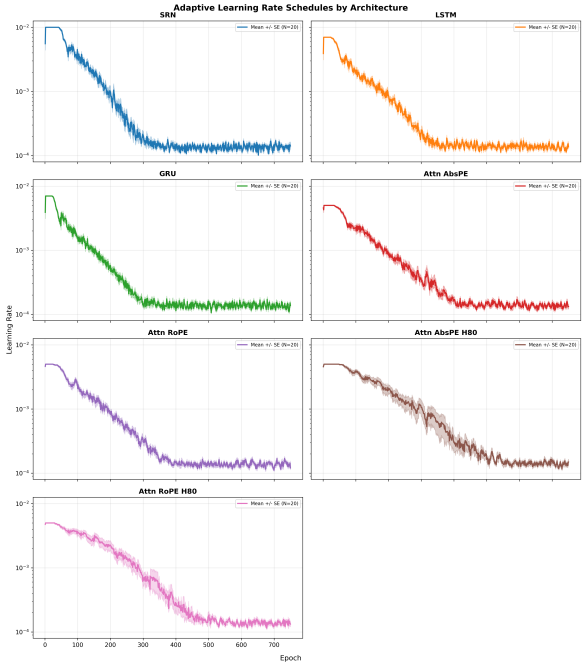

Figure 24: Adaptive learning rate schedules across training epochs for all model configurations.

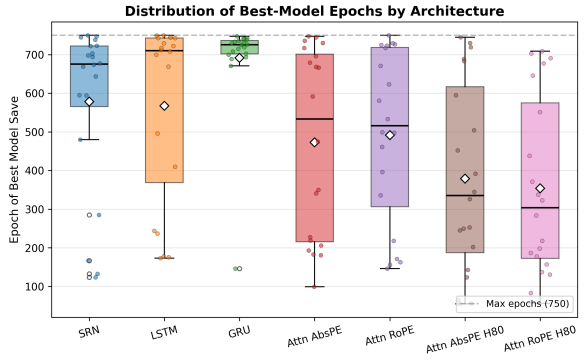

Figure 25: Distribution of best epochs (epoch with highest validation comprehension score) across model configurations.

K Brief Overview of Statistical Methods

All statistical analyses use linear mixed-effects models (LMMs) fit with the `lme4` package in R (Bates et al., 2015), with Satterthwaite degrees of freedom from `lmerTest` (Kuznetsova et al., 2017). Post-hoc contrasts were estimated with `emmeans` (Lenth, 2024), using Tukey adjustment for architecture-comparison families and multivariate-t (`mvt`) adjustment for entity-comparison families. This section defines the key statistics reported in the subsequent appendices.

For readers more familiar with classical t-tests and ANOVA than with mixed-effects models, the easiest way to read the appendices is to translate each table into the closest familiar question:

- **ANOVA tables**: analogous to omnibus ANOVA questions of the form "does anything differ at all?" for architecture, entity vectors, split, or their interactions.
- **Pairwise contrast tables**: analogous to multiple pairwise t-tests on model-adjusted means, with Tukey adjustment used to control the family-wise error rate across the full set of architecture comparisons.
- **Entity-effect tables**: analogous to a two-condition comparison asking whether $+$entity differs from $-$entity, but estimated while accounting for repeated observations from the same sentences and models, with multivariate-t (`mvt`) adjustment used for the simultaneous inference.
- **Interaction terms**: analogous to a difference-in-differences question. For example, Architecture $\times$ Train/test asks whether the train-to-test drop is larger for some architectures than for others.
- **Random effects**: these are the part that departs most from a simple t-test. They account for the fact that we repeatedly observe the same sentences and trained models, so the model does not treat all rows as independent observations.

In short, the mixed-effects analyses answer the same substantive questions a reader might ask with ANOVA and t-tests, but they do so while respecting the repeated-measures structure of the data and avoiding pseudoreplication.

K.1 Appendix Structure

The appendix tables below are structured as follows:

- each subsection begins with a brief plain-English statement of what the following tables are meant to show before presenting the statistical evidence.

- **Only reliable effects are foregrounded**: the tables retain only statistically reliable patterns. Blank cells or '–' mean that no reliable effect is being highlighted there.
- **Dominance summaries are qualitative guides**: in tables with 'wins-losses' cells, each entry counts how many statistically significant pairwise comparisons a configuration wins versus loses within that group. These tables summarize the direction of the pairwise evidence; the full details can be found in the repository's `statistical_analysis/` directory.

K.2 Model Structure

For each of the four test groups (Word, Sentence, Complex, Basic), we fit separate LMMs with the comprehension advantage score as the dependent variable. Fixed effects include architecture (7 levels: SRN, LSTM, GRU, Attn_AbsPE, Attn_RoPE, Attn_AbsPE_H80, Attn_RoPE_H80), entity vector condition (±entity), and train/test split. The random-effects structure includes random intercepts for sentences and, where justified by likelihood ratio tests (LRT), random slopes for architecture and/or entity condition by sentence. The specific fixed-effect terms retained for each test group were selected by backward elimination from the full model (all main effects and interactions), dropping non-significant higher-order interactions. The generalization gap models additionally include a train/test subset factor (whether the sentence was seen during training) and its interactions with architecture and entity vectors.

K.3 ANOVA Tables

The ANOVA tables report Type III tests of fixed effects with Satterthwaite denominator degrees of freedom. A Type III test asks whether a given fixed-effect term still explains variance after the other fixed-effect terms in the model have already been taken into account; with interactions present, this is therefore a conditional test of each retained term given the rest of the fitted model (Langsrud, 2003; Kuznetsova et al., 2017).

- F: The F-statistic, the ratio of between-group variance to within-group variance for each effect. Larger values indicate stronger evidence that the groups differ.
- **df**: Degrees of freedom (numerator, denominator). With Satterthwaite approximation for mixed-effects models, denominator degrees of freedom can be fractional, reflecting the effective sample size after accounting for random-effects structure.
- p: The probability of observing the obtained F-value (or larger) if the null hypothesis of no difference were true.

K.4 Pairwise Comparisons

Architecture pairwise comparison tables report Tukey-adjusted contrasts from estimated marginal means:

- **Est.**: Estimated mean difference between two architectures (positive means the first architecture scores higher).
- **[95% CI]**: 95% confidence interval for the mean difference.
- **SE**: Standard error of the estimated difference.
- t: The t-statistic (Est. / SE).
- p_{Tukey}: p-value adjusted for all pairwise comparisons using Tukey adjustment. This is more conservative than unadjusted p-values and controls the family-wise error rate across all 21 architecture pairs.

K.5 Entity Effect Tables

Entity effect tables report the estimated difference between +entity and −entity conditions (entity − no entity). Positive values indicate that entity vectors improve the comprehension advantage. These are raw model-based score differences on the comprehension-advantage scale. The reported p-values and confidence intervals use multivariate-t (`mvt`) adjustment for simultaneous inference within each entity-comparison family. Where the selected model includes an architecture × entity interaction, separate estimates are provided per architecture; otherwise, a single pooled estimate ("All") is reported.

K.6 Generalization Gap Analysis

The generalization-gap analyses use combined train+test LMMs that include a binary train/test subset factor (whether a sentence was seen during training) alongside architecture and entity vectors. The key interaction terms ask whether some architectures show larger train-to-test drops than others (Architecture × Train/test) and whether entity vectors help novel sentences more than seen sentences (Entity × Train/test). Detailed gap-analysis outputs are available in the repository's `statistical_analysis/` directory.

L Mixed-Effects ANOVA

Table 19 retains only the statistically significant fixed effects from the per-group mixed-effects analyses for both test and training sentences. The broad pattern is that architecture matters across nearly all groups, while entity-vector and split effects are more selective. This appendix contains summaries of the results; for more details, see the result CSVs in the repository's `statistical_analysis/` directory.

Phase	Group	Effect	F	p
Test	Word	Arch	2.55	0.022
Test	Word	Entity	20.45	$<.001$
Test	Sentence	Arch	20.72	$<.001$
Test	Sentence	Arch × Ent	3.63	0.002
Test	Complex	Arch	48.76	$<.001$
Test	Complex	Split	6.91	0.009
Test	Basic	Arch	7.18	$<.001$
Test	Basic	Entity	59.17	$<.001$
Test	Basic	Split	15.74	$<.001$
Test	Basic	Arch × Split	3.43	0.003
Test	Basic	Entity × Split	4.94	0.027
Test	Basic	Arch × Ent × Split	2.65	0.019
Train	Word	Arch	2.63	0.019
Train	Word	Entity	19.70	$<.001$
Train	Sentence	Arch	34.54	$<.001$
Train	Sentence	Entity	23.40	$<.001$
Train	Sentence	Arch × Ent	3.58	0.003
Train	Complex	Arch	13.42	$<.001$
Train	Complex	Entity	30.98	$<.001$
Train	Complex	Arch × Ent	2.85	0.013
Train	Complex	Arch × Split	2.61	0.020
Train	Basic	Arch	66.43	$<.001$
Train	Basic	Entity	9.56	0.002
Train	Basic	Split	4.67	0.032
Train	Basic	Arch × Ent	3.08	0.008

Table 19: Compact significant ANOVA summary for the mixed-effects results.

M Architecture Comparisons

This appendix presents the Tukey-adjusted architecture-contrast results in three tables. In Table 20, each cell reports how many significant pairwise comparisons that architecture wins and loses within a given group. The dominant pattern is the superiority of recurrent over attention models in the sentence, complex, and basic groups. Table 21 then lists the smaller set of within-family exceptions that the dominance table would otherwise hide. Table 22 gives the targeted Basic contrasts by entity condition cited in Observation 3.

Phase	Architecture	Word	Sentence	Complex	Basic	Total
Test	SRN	0-0	3-0	6-0	3-0	12-0
Test	GRU	0-0	2-0	4-1	3-0	9-1
Test	LSTM	0-0	3-0	4-1	3-0	10-1
Test	Attn_AbsPE	0-0	0-5	1-3	0-3	1-11
Test	Attn_RoPE	0-0	3-0	0-3	0-0	3-3
Test	Attn_AbsPE_H80	0-0	0-5	0-4	0-3	0-12
Test	Attn_RoPE_H80	0-0	2-3	0-3	0-3	2-9
Train	SRN	0-0	4-0	4-0	4-0	12-0
Train	GRU	1-0	4-0	4-0	4-0	13-0
Train	LSTM	0-0	4-0	4-0	4-0	12-0
Train	Attn_AbsPE	0-0	1-3	0-3	2-3	3-9
Train	Attn_RoPE	0-0	1-3	0-3	2-3	3-9
Train	Attn_AbsPE_H80	0-0	0-6	0-3	0-5	0-14
Train	Attn_RoPE_H80	0-1	1-3	0-3	0-5	1-12

Table 20: Compact significant architecture-dominance summary for the mixed-effects results. Each cell shows significant wins-losses within that group.

Phase	Group	Contrast	Estimate	p
Test	Sentence	Attn_AbsPE - Attn_RoPE	-0.144	$<.001$
Test	Sentence	Attn_AbsPE - Attn_RoPE_H80	-0.072	0.022
Test	Sentence	Attn_RoPE - Attn_AbsPE_H80	+0.166	$<.001$
Test	Sentence	Attn_RoPE - Attn_RoPE_H80	+0.072	0.022
Test	Sentence	Attn_AbsPE_H80 - Attn_RoPE_H80	-0.094	$<.001$
Test	Complex	SRN - GRU	+0.121	$<.001$
Test	Complex	SRN - LSTM	+0.103	0.003
Test	Complex	Attn_AbsPE - Attn_AbsPE_H80	+0.089	0.017
Train	Sentence	Attn_AbsPE - Attn_AbsPE_H80	+0.036	$<.001$
Train	Sentence	Attn_RoPE - Attn_AbsPE_H80	+0.039	$<.001$
Train	Sentence	Attn_AbsPE_H80 - Attn_RoPE_H80	-0.023	0.045
Train	Basic	Attn_AbsPE - Attn_AbsPE_H80	+0.033	$<.001$
Train	Basic	Attn_AbsPE - Attn_RoPE_H80	+0.031	$<.001$
Train	Basic	Attn_RoPE - Attn_AbsPE_H80	+0.036	$<.001$
Train	Basic	Attn_RoPE - Attn_RoPE_H80	+0.034	$<.001$

Table 21: Compact within-family architecture contrasts for the mixed-effects results. These rows capture the recurrent-vs-recurrent and attention-vs-attention exceptions hidden by the dominance summary.

Contrast	$-$ent est.	$-$ent p	$+$ent est.	$+$ent p
LSTM - GRU	+0.015	0.997	-0.004	1.000
SRN - AbsPE	+0.081	0.041	+0.048	0.557
SRN - RoPE	+0.041	0.728	+0.040	0.731
SRN - AbsPE H80	+0.076	0.068	+0.068	0.146
SRN - RoPE H80	+0.097	0.007	+0.059	0.292
LSTM - AbsPE	+0.077	0.065	+0.086	0.025
LSTM - RoPE	+0.036	0.821	+0.079	0.053
LSTM - AbsPE H80	+0.072	0.103	+0.106	0.002
LSTM - RoPE H80	+0.092	0.012	+0.097	0.006
GRU - AbsPE	+0.062	0.243	+0.090	0.015
GRU - RoPE	+0.021	0.986	+0.083	0.034
GRU - AbsPE H80	+0.057	0.340	+0.111	0.001
GRU - RoPE H80	+0.077	0.064	+0.102	0.004

Table 22: Targeted Basic architecture contrasts by entity condition for the mixed-effects results. Positive estimates favor the architecture on the left of the contrast. These are the key pairwise comparisons cited in Observation 3.

N Entity Vector Effects

This appendix summarizes the entity-vector results in two complementary ways. Table 23 gives the test-only omnibus entity main effects for all four groups, including the non-significant Sentence and Complex results cited in Observation 4. Table 24 then reports, for each architecture and group, which condition is favored when the entity-vector effect is statistically reliable under the `mvt`-adjusted simultaneous inference. Cells of the form 'Ent 0.034' mean that entity vectors improve performance by 0.034, while 'NoEnt 0.026' means the no-entity condition is better by 0.026; '–' indicates no significant effect. The main pattern is that entity vectors help broadly on word-level items and several test basic-event cases, while the direction is more selective in some training groups.

Group	F	**df**	p
Word	20.45	1, 130.2	$<.001$
Sentence	0.00	1, 128.0	0.949
Complex	0.60	1, 147.2	0.441
Basic	59.17	1, 188.9	$<.001$

Table 23: Test-only entity main effects from the mixed-effects ANOVA models. This table includes both significant and non-significant entity effects so the null results cited in Observation 4 are directly visible.

Phase	**Architecture**	**Word**	**Sentence**	**Complex**	**Basic**
Test	SRN	Ent 0.034	–	–	–
Test	GRU	Ent 0.034	–	–	Ent 0.125
Test	LSTM	Ent 0.034	–	–	Ent 0.106
Test	Attn_AbsPE	Ent 0.034	–	–	Ent 0.096
Test	Attn_RoPE	Ent 0.034	–	–	–
Test	Attn_AbsPE_H80	Ent 0.034	–	–	–
Test	Attn_RoPE_H80	Ent 0.034	Ent 0.095	–	Ent 0.101
Train	SRN	Ent 0.035	–	–	–
Train	GRU	Ent 0.035	NoEnt 0.046	–	–
Train	LSTM	Ent 0.035	NoEnt 0.034	–	–
Train	Attn_AbsPE	Ent 0.035	–	–	–
Train	Attn_RoPE	Ent 0.035	–	–	–
Train	Attn_AbsPE_H80	Ent 0.035	–	Ent 0.040	–
Train	Attn_RoPE_H80	Ent 0.035	–	Ent 0.056	Ent 0.031

Table 24: Compact entity-effect matrix for the mixed-effects results. Cells show the favored condition and absolute estimate magnitude; '–' indicates no significant effect.