TRACE-Afford: Context-Sensitive Affordance Retrieval from an LLM-Constructed Typed Knowledge Graph

Shunsuke Liu[1,*], Katsuma Marumo[1], Kazuma Arii[1] and Satoshi Kurihara[1]

[1]Keio University, Kanagawa, Japan

Abstract

The actions afforded by an object depend on its situation: changing a location, time, instrument, or co-occurring object can change which of several plausible actions is preferred. We present TRACE-Afford, a neural-to-symbolic pipeline that converts screened language-model descriptions of everyday human actions into a reusable, typed affordance knowledge graph. The graph separates entities, intrinsic attributes, action contexts, bare actions, and object-bound affordance instances. Repeated observations are retained as frequency evidence and normalized locally to support inspectable shortest-path retrieval from multiple inputs. We also introduce a 255-pair Context Intervention Benchmark. Each pair holds the base object and two candidate actions fixed and requires their expected ordering to reverse between two situations that differ in instrument or resource, location, temporal or task-stage context, or co-occurring object. Built from 10.0 million screened sentence occurrences elicited with 2,000 seed terms, the resulting graph contains 3.35 million nodes and 12.08 million distinct labeled edges. TRACE-Afford achieves 82.4% pair accuracy; replacing local relative-frequency costs with absolute-frequency decay reduces accuracy to 71.8%, and removing frequency information reduces it to 53.3%. These results show that the proposed node schema and edge structure, together with relative frequency, enable a broad symbolic graph to recover contextual preferences while retaining the paths that support each ranking.

Keywords

knowledge graphs, neurosymbolic AI, affordances, large language models, contextual reasoning

CEUR-WS.org/Vol-4313/kgnesy-paper18.pdf

1. Introduction

An object rarely determines a single action independently of its surroundings. An apple beside a knife supports cutting, while the same apple in a store supports buying. These examples concern the same base object but change the preferred action through an instrument or location. This context dependence is central to the ecological account of affordances [1] and to their use in design and human–object interaction [2]. It is also required when a robotic agent uses perceived object and context symbols to retrieve candidate actions for downstream task planning.

Commonsense knowledge graphs provide reusable symbols and explicit relations, but resources such as ConceptNet and ATOMIC predominantly expose fixed relations or event-centered assertions [3, 4, 5]. Such assertions relate one concept or event to another but do not define how multiple observed inputs should jointly change an affordance ranking. A relation such as *UsedFor* can associate an object with several plausible actions without representing why one should be preferred in a particular situation. Conversely, a large language model (LLM) can answer a contextual query directly, but each answer requires another model call and does not itself create a persistent, editable symbolic resource or a reusable evidence path.

We address this gap with TRACE-Afford, a neural-to-symbolic pipeline that externalizes broad-coverage observations of ordinary human actions in a typed graph. The graph separates objects, intrinsic attributes, action contexts, bare actions, and object-bound actions using six node types and ten labeled relations. Its final rankings are determined by graph structure, observed frequencies, and shortest paths rather than by another generation step. Repeated observations are aggregated as edge frequencies and normalized within local outgoing-edge pools. On a manually constructed and reviewed

Third Workshop on Knowledge Graphs and Neurosymbolic AI (KG-NeSy 2026), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy

*Corresponding author.

liushunsuke@keio.jp (S. Liu)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-05

benchmark spanning four contextual factors, the full method achieves 82.4% pair accuracy, compared with 53.3% when frequency information is removed.

The main technical question is not whether a target appears somewhere in a large graph. With enough generated text, many plausible actions may be reachable. The stronger question is whether controlled contextual evidence changes their *relative preference* in the human-expected direction. We therefore ask:

RQ1 Does the system reverse the preference between two plausible actions when only the location, time, instrument/resource, or co-occurring entity changes?

RQ2 How much does observed relative frequency improve contextual preference retrieval over the same typed graph using topology and edge-type penalties alone?

The contributions are:

- a typed, frequency-aware affordance graph that retains both intransitive Action nodes and object-bound Affordance Instance nodes;
- an evaluation protocol, instantiated as a manually curated 255-pair Context Intervention Benchmark, that tests whether contextual changes reverse the ordering of two fixed candidate actions;
- controlled comparisons against direct LLM selection and graph variants that isolate the effects of the node schema and edge structure, frequency weighting, and path penalties, with retrieval paths that make individual rankings auditable.

2. Related Work

2.1. Affordance Representation and Prediction

Affordances have been modeled through sensorimotor learning, object–action complexes, visual recognition, language, and knowledge bases [6, 7, 8, 9]. Language-based methods broaden coverage beyond a fixed visual inventory. Fulda et al. extracted affordances with word embeddings [10]; more recent work probes language models directly [11] or combines language with robot and visual grounding [12, 13]. These systems establish that linguistic representations contain object–action knowledge. Our focus differs: we test whether a persistent symbolic resource can condition affordance rankings on explicit contextual inputs.

Arii et al. propose a three-stage approach that prompts an LLM for human-action sentences, converts them into a symbol network through dependency analysis, and retrieves actions by graph search [14]. Their main experiment starts network construction from `apple` and shows intuitive outputs under selected attributes, locations, and accompanying objects, such as buying an apple at a store and slicing one with a knife. Separate scale experiments consider `apple`, `paper`, `chair`, and `computer`. These results establish the feasibility and inspectability of language-derived affordance retrieval, but do not determine how well one broad graph supports diverse object domains or controlled changes in contextual preference. Their four-type representation also combines intrinsic attributes with action contexts and bare verbs with object-bound actions, while leaving graph links unlabeled and using different connection rules. This motivates evaluating whether a revised node schema and edge structure improve retrieval in a broad graph. We address this open question with 2,000 seed terms and a quantitative intervention benchmark.

2.2. LLMs and Symbolic Knowledge Construction

Knowledge graphs offer explicit entities, relations, and graph operations [15]. Neural commonsense graph construction instead uses language models to generate or distill assertions, as in COMET and symbolic knowledge distillation [16, 17]. Recent work also treats LLMs as ontology learners, domain-KG constructors, or prompt-to-triple converters [18, 19, 20]. These approaches highlight both the coverage

of neural generators and the need to constrain their outputs to a symbolic schema. Quality control for machine-generated knowledge commonly combines scalable automatic screening with human assessment of a sampled subset. For example, ATOMIC10x trains a critic from human acceptability judgments, while SAC-KG checks the agreement of its LLM evaluator with human domain judgments [17, 19]. We follow this division between corpus-wide screening and sample-based validation rather than treating an LLM verdict as ground truth.

TRACE-Afford differs from direct triple generation. The generator produces short natural-language descriptions under one fixed prompt, Stanford CoreNLP derives lemmas and dependencies [21], and deterministic extraction rules construct the graph. Repeated observations are aggregated as edge frequency rather than discarded. The final reasoning procedure is therefore symbolic and inspectable even though the observation source is neural. This design is consistent with broader neural-symbolic KG reasoning, where learned components and explicit graph operations play distinct roles [22, 23].

2.3. Contextual Evaluation

Existing affordance evaluations commonly predict action labels for isolated objects, images, or short activity contexts. Neuro-symbolic action anticipation shows the value of combining action knowledge and context [24]. Our benchmark complements these tasks by holding two action candidates fixed and intervening on one explicit contextual factor. This makes responsiveness measurable without treating the system's own distance scale as a universal metric.

3. Method

3.1. Neural-to-Symbolic Pipeline

Figure 1 summarizes the pipeline. A neural generator supplies varied observations of human actions. A prompt-compliance stage removes malformed or implausible outputs. CoreNLP then provides tokenization, lemmas, part-of-speech tags, and enhanced dependencies. Deterministic rules transform each analysis into typed nodes and labeled edges. At query time, no generative model is needed: weighted graph search ranks Action and Affordance Instance nodes and returns the paths used to compute each score.

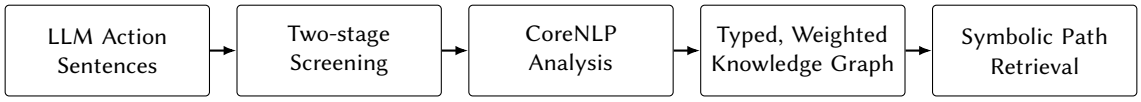


Figure 1: TRACE-Afford construction and retrieval. Neural models provide and screen observations; the final ranking is computed by explicit graph operations.

3.2. Sentence Elicitation and Screening

For each seed noun or noun phrase o, the generator receives the unchanged prompt shown below; `{object}` is replaced by o, and `{count}` controls the number of sentences requested in that completion.

```
Write {count} short sentences describing typical human actions involving
“{object}.”
Each sentence must:
- start with “I”
- include “{object}” as a noun only
- describe a realistic human action
Output only the sentences.
```

Generated sentences undergo two screening stages before parsing. Deterministic checks enforce the prompt's observable constraints: the sentence must start with “I” and contain the seed under an

ordinary singular, plural, or possessive inflection. A semantic validator then checks whether each sentence is grammatical, uses the seed literally as a noun, expresses a coherent relation between the seed and the action, describes an action or intentional experience involving a human agent, and depicts an ordinary, physically possible situation. The screen tests compliance with the elicitation task rather than claiming exhaustive factual verification. Repeated accepted sentences are retained because their multiplicity supplies the frequency signal used by retrieval. Section 4.2 gives the models, corpus scale, and validation protocol.

3.3. Typed Affordance Graph

The directed multirelational graph $G = (V, E)$ has six node types (Figure 2); Table 1 defines its ten labeled relations. Entity nodes represent objects and qualified object expressions. Attribute nodes represent intrinsic properties such as `red`, whereas Context nodes preserve external action settings such as `with_knife`, `at_store`, or `before_lunch`. Action nodes contain bare lemmatized verbs, allowing intransitive affordances such as `sleep` to remain retrievable without an artificial object-bound form. Affordance Instance nodes instead bind transitive actions to their patients, for example `cut_apple`. Canonical Concept nodes represent lemmas.

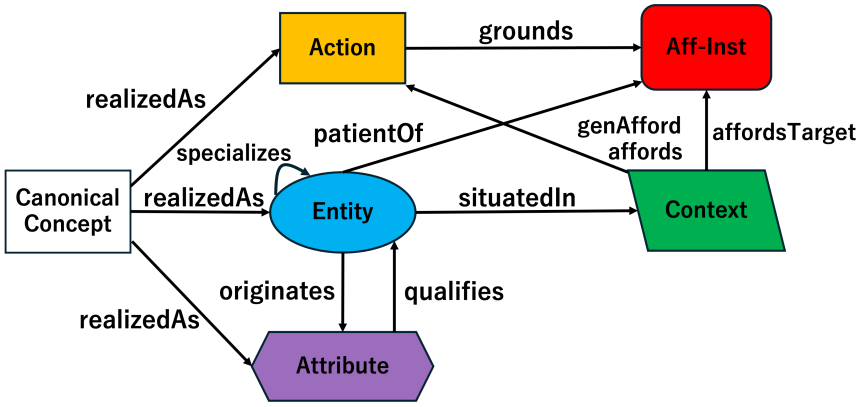


Figure 2: Typed graph schema. Node shape and color distinguish canonical concepts, entities, attributes, contexts, actions, and object-bound affordance instances; arrows show the permitted labeled relations.

Table 1
Labeled edge definitions.

Relation	Direction	Meaning
realizedAs	Canonical → typed node	realizes a lemma as an Entity, Attribute, or Action
specializes	Entity → Entity	links a base entity to a qualified entity
originates	Entity → Attribute	derives an attribute from its nominal source
qualifies	Attribute → Entity	applies an intrinsic property to an entity
grounds	Action → Affordance Instance	binds a bare action to an object-specific action
patientOf	Entity → Affordance Instance	identifies the patient of a transitive action
situatedIn	Entity → Context	forms a prepositional or temporal action context
affordsTarget	Context → Affordance Instance	directly supports an object-bound action
affords	Context → Action	supports an intransitive action
genAfford	Context → Action	transfers transitive-action evidence across objects

The retained sentences are analyzed with Stanford CoreNLP. POS tags and a restricted set of enhanced dependency relations identify predicates, patients, modifiers, and contexts; a deterministic converter then maps these analyses to the typed nodes and edges defined above. Figure 3 illustrates the conversion of "I cut the red apple with a knife." CoreNLP identifies *cut* as the predicate, *apple* as its object, *red* as an adjectival modifier, and *knife* as an oblique nominal introduced by *with*. The converter

creates Attribute `red`, Entities `apple`, `red_apple`, and `knife`, Context `with_knife`, Action `cut`, and Affordance Instance `cut_apple`. The modifier qualifies the specialized Entity `red_apple`; `apple` and `red_apple` are patients of `cut_apple`; and `knife` is linked to `with_knife` by *situatedIn*. The context supports `cut` through *genAfford* and `cut_apple` through *affordsTarget*. For an intransitive sentence such as "I sleep in a bed," the converter creates `in_bed`:Context and `sleep`:Action, then connects the context to the action with an *affords* edge.

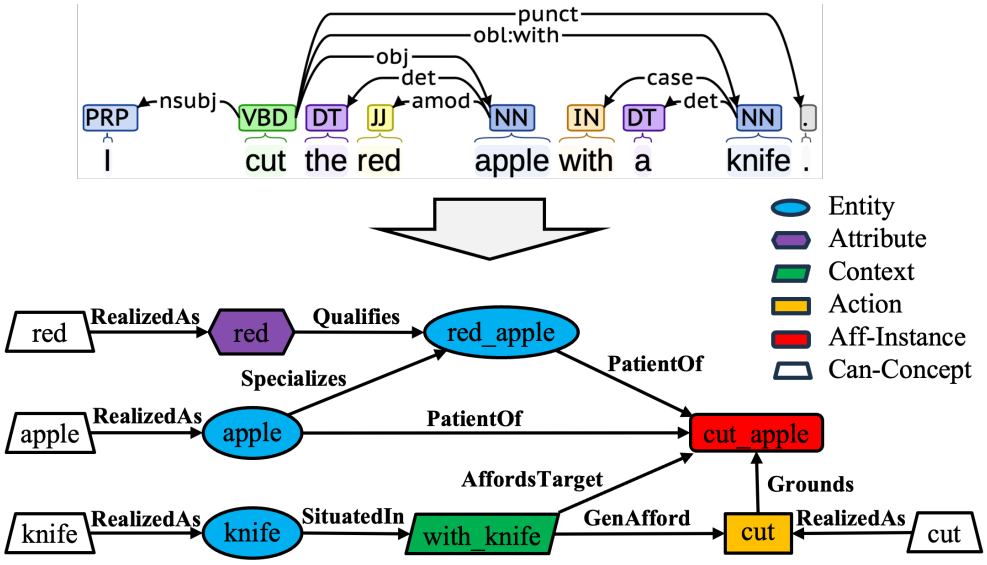


Figure 3: Example conversion from a CoreNLP dependency analysis to typed graph nodes and edges.

3.4. Relative-Frequency Retrieval

Repeated observations provide evidence that one relation is more typical than another, but raw counts are not directly comparable across objects with very different amounts of generated text. We therefore evaluate each edge against its local alternatives. Each non-Context source defines one pool containing all of its outgoing edges. For a Context source, *genAfford* edges form one pool and all remaining outgoing edges form another. For an edge e, let $f(e)$ be its observed frequency and $f_{\max}(e)$ the largest frequency in its pool. We convert relative support into a traversal cost:

$$w(e) = \epsilon_e + 1 - \frac{f(e)}{f_{\max}(e)}. \tag{1}$$

The frequency term lies in $[0, 1)$, so structural penalties use the same unit scale. The edge penalty is $\epsilon_e = 0.5$ for standard edges and $\epsilon_e = 1.0$ for *genAfford*, which transfers action evidence across objects. These values were fixed before benchmark evaluation without consulting its labels or system predictions. All costs are nonnegative, and Dijkstra's algorithm finds the least-cost path $d(q, a)$ from an observed input q to a candidate action a [25].

Suppose a query contains m observed inputs $q_1, \ldots, q_m$, such as `apple` and `knife`. Let $\widetilde{d}(q_j, a)$ equal the shortest-path distance when input q_j reaches candidate a, and a fixed penalty ρ otherwise. For a candidate reached from at least one input, the score is the arithmetic mean:

$$D(q_1, \ldots, q_m; a) = \frac{1}{m} \sum_{j=1}^{m} \widetilde{d}(q_j, a). \tag{2}$$

We set $\rho = 5$, following the multi-input penalty used by Arii et al. A candidate reached from both `apple` and `knife` is therefore favored over one supported by only one input, all else being equal. A candidate unreachable from every input receives $+\infty$; lower scores are preferred.

4. Experimental Setup

4.1. Task and Benchmark

An item i contains two situations S_i^A and S_i^B and two fixed candidate actions t_i^A and t_i^B. The situations differ in one contextual cue; Entity-centered pairs retain the same focal Entity. Situation A should prefer t_i^A and situation B should prefer t_i^B. For example:

$$S^A = \{\texttt{orange}, \texttt{knife}\}: \quad \texttt{cut_orange} \succ \texttt{juice_orange},$$
$$S^B = \{\texttt{orange}, \texttt{juicer}\}: \quad \texttt{juice_orange} \succ \texttt{cut_orange}.$$

Here the contextual cue, rather than the base object alone, determines the preferred action. Transitive targets use object-bound names such as `cut_orange`, while naturally intransitive targets remain Action nodes such as `sleep` or `sit`.

We instantiate the contextual preference-reversal task with 255 pairs across four intervention categories (Table 2). The input representation follows the graph schema: an intervention is expressed by a second Entity or by a Context node.

Table 2
Context Intervention Benchmark categories and query forms.

Category	Situation input	Intended contrast	Pairs
Instrument/resource	Entity (+ Entity)	tool or usable resource changes the task	66
Location	Entity (+ Context), or Context	setting changes the typical action	72
Temporal/task stage	Entity (+ Context)	time or stage changes the typical action	54
Co-object	Entity (+ Entity)	content, recipient, support, or companion object changes the task	63
Total			255

Two annotators jointly constructed and reviewed all pairs, requiring ordinary candidate actions, one primary contextual difference, and a preference reversal. They finalized the benchmark before graph construction without consulting retrieval results or graph statistics.

4.2. Corpus and Graph Construction

The seed inventory contains 2,000 everyday nouns and noun phrases spanning food, household objects, tools, furniture, electronics, clothing, transport, recreation, and public places. The generator is GPT-4o-mini at temperature 0.7. One generation set requests 20, 21, . . ., 50 sentences in separate calls for each seed, or 1,085 requested sentences per seed. Cumulative 1-, 3-, and 5-set conditions contain 1,085, 3,255, and 5,425 requested sentences per seed. These requests yielded 10,848,671 sentence occurrences across the five sets.

After deterministic screening, GPT-5.1 applies the semantic rubric at temperature zero. Repeated accepted occurrences are retained as frequency evidence during graph construction. Deterministic checks remove 0.46% of generated occurrences, and semantic screening removes a further 787,076, leaving 10,011,954 occurrences (Table 3). Because fluent LLM-generated text can still contain factual or semantic errors [26], we audit the screening decisions against human judgments. One author independently labels 50 validator-accepted and 50 validator-rejected sentences without seeing their validator labels. The human judgments agree with the validator on 49 accepted cases and 35 rejected cases, yielding 84/100 agreement (84%).

To assess the CoreNLP-to-graph conversion, we randomly sampled 200 retained sentences and manually inspected their resulting graph fragments. Five sentences (2.5%) contained erroneous emitted edges; 27 of 2,102 emitted edges were unsupported or mistyped (1.28%). The estimated precision among emitted edges was therefore 98.72%; this audit does not measure omitted relations.

The resulting principal graph contains 3,349,480 nodes and 12,083,305 distinct directed, labeled edges (Table 4). Repeated observations provide 99,230,268 edge instances before aggregation.

Table 3
Generated-corpus screening in sentence occurrences.

Quantity	Count
Generated sentence occurrences	10,848,671
Rejected by deterministic checks	49,641
Submitted to semantic screening	10,799,030
Rejected after semantic screening	787,076
TRACE-Full corpus	**10,011,954**

Table 4
Node composition of the graph constructed from 2,000 seeds and five generation sets.

Component	Count
Entity nodes	1,552,846
Context nodes	910,453
Affordance Instance nodes	551,873
Attribute nodes	307,185
Canonical Concept nodes	20,419
Action nodes	6,704
Total nodes	3,349,480

4.3. Evaluation Metric

Raw distances are internal to each weighting system and are never compared across systems. We set the distance to an unreachable candidate to $+\infty$. For each pair, the system scores both fixed candidates under both situations:

$$C_i^A = \mathbb{1}[D(S_i^A, t_i^A) < D(S_i^A, t_i^B)], \tag{3}$$

$$C_i^B = \mathbb{1}[D(S_i^B, t_i^B) < D(S_i^B, t_i^A)]. \tag{4}$$

Our primary pair accuracy is

$$\mathrm{PairAcc} = \frac{1}{N} \sum_{i=1}^{N} C_i^A C_i^B. \tag{5}$$

A reachable expected candidate beats an unreachable contrast; an unreachable expected candidate, a tie, or a missing input node is incorrect. Because some pairs share a focal Entity, we compute percentile 95% confidence intervals from 10,000 cluster-bootstrap samples over 149 benchmark clusters, grouping pairs by focal Entity and treating Context-only items separately. Paired system differences use the same sampled clusters. The primary comparison for RQ2 is TRACE-Full versus TRACE-NoFreq.

4.4. Experiment 1: Representation and Edge Costs

Experiment 1 tests contextual reversal while separating representation from edge costs (Table 5). The direct-choice reference sends both situations and the same candidates to GPT-4o-mini in one temperature-zero request; it independently shuffles candidate order, and pair accuracy requires both selections. All graph variants are constructed from the screened corpus and CoreNLP analyses described in Section 4.2; only the graph representation and edge-cost function vary. The Arii schema + TRACE weighting condition combines the proposed cost with Arii's four-type node organization and unlabeled connection rules, placing all outgoing edges in one pool. In both Arii-schema conditions, benchmark Context inputs map to Attributes, while Action and Affordance Instance targets map to Actions. All graph conditions use the same retrieval and multi-input scoring rules. Comparing TRACE-Full with Arii schema + TRACE weighting isolates the representation, including node definitions and connectivity; comparing it with TRACE schema + Arii weighting isolates edge weighting.

Table 5
Principal conditions in Experiment 1.

System	Condition
GPT-4o-mini (direct)	Selects between the two fixed actions from each natural-language situation.
Arii schema + Arii weighting	Arii's node-and-edge structure and absolute-frequency cost, $0.99^{f(e)}$.
Arii schema + TRACE weighting	Arii's node-and-edge structure with the TRACE relative-frequency cost.
TRACE schema + Arii weighting	Proposed node schema and edge structure with the edge cost replaced by Arii's absolute-frequency decay, $0.99^{f(e)}$.
TRACE-Full	Six types and ten relations with the local relative-frequency cost in Eq. 1 and the missing-input penalty.

4.5. Experiment 2: Corpus Scale

Experiment 2 varies one corpus factor at a time for TRACE-Full. One fixed random permutation supplies nested subsets of 500, 1,000, and 2,000 seeds at five generation sets. We separately compare one, three, and five cumulative generation sets at 2,000 seeds.

4.6. Experiment 3: Retrieval Mechanisms

Experiment 3 ablates three components of TRACE-Full. TRACE-NoFreq removes the observed-frequency term from Eq. 1, leaving only the edge-type penalty. TRACE-ϵ0 sets all edge-type penalties to zero, while TRACE-NoInputPenalty sets the missing-input penalty to zero.

5. Results and Discussion

5.1. Result 1: Representation and Edge Costs

Table 6
Pair accuracy (%) by intervention category for direct choice and the graph conditions. The direct-choice reference and the best graph result in each column are bold.

System	Instr./res.	Location	Temporal	Co-obj.	All
GPT-4o-mini (direct)	**93.9**	**100.0**	**92.6**	**85.7**	**93.3**
Arii schema + Arii weighting	81.8	43.1	44.4	66.7	59.2
Arii schema + TRACE weighting	80.3	51.4	53.7	76.2	65.5
TRACE schema + Arii weighting	81.8	72.2	59.3	71.4	71.8
TRACE-Full	**83.3**	**87.5**	**74.1**	**82.5**	**82.4**

TRACE-Full answers 210 pairs correctly (82.4%, cluster 95% CI 77.3–86.9) and leads the graph conditions in every category. It exceeds TRACE schema + Arii weighting by 10.6 points (95% CI 5.7–15.5) and Arii schema + TRACE weighting by 16.9 points (11.0–23.0). Only 2 pairs (0.8%) have neither situation correct. Results across all four categories support contextual preference retrieval (RQ1).

The graph comparisons show complementary contributions from representation and weighting. The TRACE node schema and edge structure, which change both node organization and edge connectivity, improve over the Arii structure under the same relative-frequency cost, while local relative normalization improves over absolute decay in the same TRACE representation. With $w_{\text{abs}}(e) = 0.99^{f(e)}$, costs for frequent edges converge exponentially toward zero. Frequency differences among well-supported edges consequently become negligible, while rarer links can dominate a path's total cost. Local relative

normalization instead preserves frequency contrasts among edges in the same local outgoing pool. In the broad, large-scale graph and contextual intervention setting studied here, Arii's absolute-frequency cost therefore transfers less effectively than the proposed relative-frequency cost. The same ordering appears within Arii's representation: replacing absolute decay with local relative normalization raises pair accuracy from 59.2% to 65.5%. For 212 of 255 pairs (83.1%), the Arii schema + Arii weighting condition contains all required input nodes and can reach both candidates in both situations; it answers 151 pairs correctly. Of the remaining pairs, 42 lack an input node and one has an unreachable candidate.

Prompted with both situations and the two fixed candidates, direct GPT-4o-mini answers 238 pairs correctly (93.3%, cluster 95% CI 90.0–96.2), compared with 210 (82.4%) for TRACE-Full, an 11.0-point difference (95% CI 5.6–16.6). Although the direct model attains higher pair accuracy, TRACE-Afford stores its observations in an editable graph, enabling repeated model-free queries, incorporation of additional observations, and inspection of the paths supporting each result. At the evaluated graph scale, TRACE-Full also had lower observed per-pair latency. On an Apple M3 system with 24 GB RAM, retrieval averaged 0.618 seconds per pair after graph loading, whereas one GPT-4o-mini API request returning both decisions averaged 1.288 seconds. Peak resident memory during graph loading and retrieval was 2.97 GiB.

5.2. Result 2: Corpus Scale

Table 7 reports both corpus-scale analyses.

Table 7
Corpus-scale ablations for TRACE-Full with cluster-bootstrap 95% CIs. Seed scale uses five sets; generation scale uses 2,000 seeds.

Varied factor	Level	Pair Acc. [95% CI] (%)
Generation sets	1	69.8 [63.5, 75.7]
Generation sets	3	80.4 [75.2, 85.1]
Generation sets	5	**82.4 [77.3, 86.9]**
Seeds	500	50.6 [43.3, 57.8]
Seeds	1,000	62.7 [55.8, 69.3]
Seeds	2,000	**82.4 [77.3, 86.9]**

Accuracy increases substantially from one to three generation sets, with a further gain of only 2.0 points from three to five. Increasing seed coverage produces the larger observed gain: the 2,000-seed graph is 31.8 points above the 500-seed graph. Because this comparison uses one nested seed order, repeated random samples are needed to establish whether that pattern generalizes.

5.3. Result 3: Retrieval Mechanisms

Table 8 isolates the frequency term, edge-type penalty, and missing-input penalty in the full representation.

Table 8
Weighting and scoring ablations at 2,000 seeds and five generation sets. Δ is TRACE-Full minus the ablation, with a cluster-bootstrap 95% CI.

System	Pair Acc. (%)	Δ [95% CI]
TRACE-Full	**82.4**	0.0
TRACE-NoFreq	53.3	29.0 [23.4, 34.9]
TRACE-ϵ0	76.1	6.3 [3.1, 9.7]
TRACE-NoInputPenalty	21.2	61.2 [54.7, 67.3]

Removing frequency gives 53.3%, 29.0 points below TRACE-Full (95% CI 23.4–34.9). Setting the edge penalties ϵ_e to zero gives 76.1%, a 6.3-point difference (3.1–9.7), while setting the missing-input penalty to zero reduces accuracy to 21.2%, a 61.2-point difference (54.7–67.3).

On this benchmark, the Full–NoFreq difference shows that observed frequency contributes beyond graph topology and edge-type penalties (RQ2). The 6.3-point decrease under TRACE-ϵ0 shows that relation-specific traversal costs also affect retrieval. The large NoInputPenalty drop indicates that the missing-input penalty prevents candidates supported by only one query input from being favored over alternatives supported by both.

5.4. Qualitative Analysis

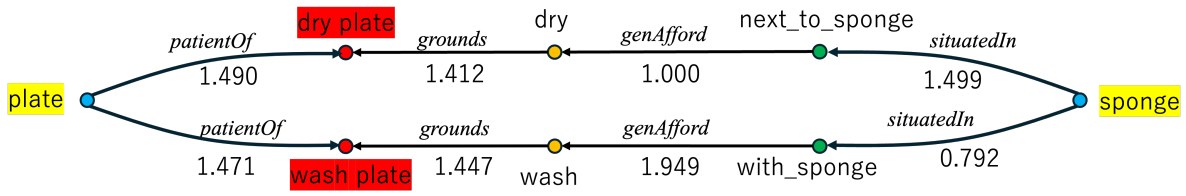
Table 9 presents unrestricted top-three retrievals for an entity alone and with instrument and location evidence. These rankings are produced by the general retrieval procedure rather than being limited to the benchmark's two fixed candidates.

Table 9
Successful context-sensitive retrieval examples. Lower scores indicate stronger preference.

`apple`		`apple + knife`		`apple + at_grocery_store`	
Candidate	Score	Candidate	Score	Candidate	Score
`pick_apple`	1.231	`peel_apple`	1.568	`buy_apple`	1.357
`slice_apple`	1.341	`slice_apple`	1.666	`pick_up_apple`	1.443
`share_apple`	1.390	`cut_apple`	1.697	`compare_apple`	1.484

The entity-only query yields a broad set of ordinary apple actions. Adding a knife concentrates the ranking on preparation actions, while the grocery-store context produces shopping actions. The displayed candidates are broadly plausible for their respective inputs, showing that contextual evidence can change the broader retrieved neighborhood rather than only the ordering of two fixed targets.

Figure 4 uses one failed pair to illustrate how the returned paths expose the source of a ranking error.



Input: `plate+sponge` **Scores:** `dry_plate` 2.701 (selected), `wash_plate` 2.829 (expected)

Figure 4: Path analysis of the failed `plate+sponge` query. Edge labels appear above the arrows and traversal costs below them. Each score averages the paths from `plate` and `sponge`.

For this pair, `wash_plate` should rank ahead of `dry_plate`, but the retrieved scores reverse this order by 0.128. The main spurious path passes through `next_to_sponge`, whose only outgoing *genAfford* observation points to `dry`; local normalization therefore gives this singleton edge maximal relative support. The graph also contains no direct *affordsTarget* edge from `with_sponge` to `wash_plate`. Nevertheless, 16 observations connect `with_sponge` to the generalized action `wash`, and 132 observations ground `wash` to `wash_plate`. The expected candidate is thus reached compositionally from experiences of washing other objects with a sponge, although the weak dry path narrowly outranks it.

This failure also exposes a limitation of purely local normalization: an edge observed once can be maximal within a sparse outgoing pool. A support-aware extension could preserve relative preference while reducing confidence in poorly observed edges. Spreading activation or a GNN reranker over the retrieved subgraph could further combine evidence from multiple paths without discarding the explicit paths used for inspection; these alternatives require evaluation on held-out interventions.

6. Conclusion and Future Work

We introduced TRACE-Afford, a neural-to-symbolic pipeline that externalizes screened descriptions of everyday human actions as an editable, frequency-aware graph for repeatable and inspectable context-sensitive affordance retrieval. The Context Intervention Benchmark evaluates a demanding two-sided preference reversal rather than mere target reachability. Across 255 pairs, the full system attains 82.4% accuracy and achieves 74.1–87.5% across all four intervention categories. Replacing local relative-frequency costs with absolute-frequency decay lowers accuracy to 71.8%, while removing frequency information lowers it to 53.3%. The representation and weighting comparisons therefore show that the proposed node schema and edge structure, together with locally normalized observation frequency, contribute to contextual retrieval. The returned paths additionally expose the evidence and failure modes behind each ranking.

Future work will expand the benchmark beyond 255 two-situation comparisons, involve additional independent annotators, and cover scenes with multiple simultaneous contextual cues. We will also investigate RDF/OWL serialization and alignment with existing commonsense resources while preserving the contextual distinctions represented by TRACE-Afford. We will further connect camera-based scene recognition to graph inputs and evaluate whether the retrieved affordances improve plan feasibility and task success in robot planning and execution.

Acknowledgments

This work was supported by the Honda Bridge Project.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5.5 for language editing and grammar and spelling checks. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] J. J. Gibson, The Ecological Approach to Visual Perception: Classic Edition, Psychology Press, New York, 2014. Originally published 1979.

[2] D. A. Norman, The Design of Everyday Things, Basic Books, New York, 1988.

[3] R. Speer, J. Chin, C. Havasi, ConceptNet 5.5: An open multilingual graph of general knowledge, Proceedings of the AAAI Conference on Artificial Intelligence 31 (2017) 4444–4451. doi:10.1609/aaai.v31i1.11164.

[4] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, Y. Choi, ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3027–3035.

[5] J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, Y. Choi, (Comet-)Atomic 2020: On Symbolic and Neural Commonsense Knowledge Graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 6384–6392.

[6] L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor, Learning object affordances: From sensory–motor coordination to imitation, IEEE Transactions on Robotics 24 (2008) 15–26.

[7] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, R. Dillmann, Object–action complexes: Grounded abstractions of sensorimotor processes, Robotics and Autonomous Systems 59 (2011) 740–757.

[8] Y. Zhu, A. Fathi, L. Fei-Fei, Reasoning about object affordances in a knowledge base representation, in: Computer Vision – ECCV 2014, volume 8690, Springer, 2014, pp. 408–424. doi:10.1007/978-3-319-10605-2_27.

[9] C.-Y. Chuang, J. Li, A. Torralba, S. Fidler, Learning to act properly: Predicting and explaining affordances from images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 975–983.
[10] N. Fulda, D. Ricks, B. Murdoch, D. Wingate, What can you do with a rock? Affordance extraction via word embeddings, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 1039–1045.
[11] S. Adak, D. Agrawal, A. Mukherjee, S. Aditya, TEXT2AFFORD: Probing Object Affordance Prediction abilities of Language Models solely from Text, arXiv preprint arXiv:2402.12881 [cs.CL] (2024). URL: https://arxiv.org/abs/2402.12881.
[12] N. Wake, D. Saito, K. Sasabuchi, H. Koike, K. Ikeuchi, Text-driven object affordance for guiding grasp-type recognition in multimodal robot teaching, Machine Vision and Applications 34 (2023) 58.
[13] M. Ahn, A. Brohan, et al., Do as i can, not as i say: Grounding language in robotic affordances, in: Proceedings of the Conference on Robot Learning, volume 205 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 287–318.
[14] K. Arii, S. Liu, S. Kurihara, Affordance-driven symbol network construction via large language models, Artificial Life and Robotics 31 (2026) 547–561. doi:10.1007/s10015-025-01106-1.
[15] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, ACM Computing Surveys 54 (2021) Article 71.
[16] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, Y. Choi, COMET: Commonsense Transformers for Automatic Knowledge Graph Construction, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), 2019, pp. 4762–4779.
[17] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. Le Bras, X. Lu, S. Welleck, Y. Choi, Symbolic Knowledge Distillation: From General Language Models to Commonsense Models, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2022, pp. 4602–4625.
[18] H. Babaei Giglou, J. D'Souza, S. Auer, LLMs4OL: Large Language Models for Ontology Learning, in: Proceedings of the International Semantic Web Conference (ISWC), 2023, pp. 408–427.
[19] H. Chen, X. Shen, Q. Lv, J. Wang, X. Ni, J. Ye, SAC-KG: Exploiting Large Language Models as Skilled Automatic Constructors for Domain Knowledge Graph, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2024, pp. 4345–4360.
[20] T. Çöplü, A. Bendiken, A. Skomorokhov, E. Bateiko, S. Cobb, J. J. Bouw, Prompt-time symbolic knowledge capture with large language models, arXiv preprint arXiv:2402.00414 (2024). URL: https://arxiv.org/abs/2402.00414.
[21] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, D. McClosky, The Stanford CoreNLP Natural Language Processing Toolkit, in: K. Bontcheva, J. Zhu (Eds.), Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 55–60. URL: https://aclanthology.org/P14-5010/. doi:10.3115/v1/P14-5010.
[22] K. Cheng, N. K. Ahmed, R. A. Rossi, T. Willke, Y. Sun, Neural-symbolic methods for knowledge graph reasoning: A survey, ACM Transactions on Knowledge Discovery from Data 18 (2024) 225:1–225:44. doi:10.1145/3686806.
[23] A. Bosselut, R. Le Bras, Y. Choi, Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 4923–4931. doi:10.1609/aaai.v35i6.16625.
[24] S. Bhagat, S. Li, J. Campbell, Y. Xie, K. Sycara, S. Stepputtis, Let me help you! neuro-symbolic short-context action anticipation, IEEE Robotics and Automation Letters 9 (2024) 9749–9756. doi:10.1109/LRA.2024.3421848.
[25] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1

(1959) 269–271.
[26] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, P. Fung, Survey of hallucination in natural language generation, ACM Computing Surveys 55 (2023) 1–38.