Separating Grounding from Verification: Property-Graph-Backed Generation and Independent Claim Checking for Trustworthy LLM Recommendations

Rafał Maciejewski[1,*], Robert A. Kłopotek[1]

[1]Faculty of Mathematics and Natural Sciences, Cardinal Stefan Wyszyński University in Warsaw, ul. Wóycickiego 1/3, 01-938 Warsaw, Poland

Abstract

Large language models (LLMs) can justify a recommendation with fluent prose, but the statistics quoted in such justifications are often wrong. Graph-augmented pipelines reduce this problem by placing retrieved data in the prompt, yet they rarely check the generated text against the data afterwards, so misquoted or invented numbers pass through unnoticed. This paper describes and evaluates a pipeline for hockey lineup recommendation in which a property graph built from official National Hockey League (NHL) data is used twice: as the retrieval source that grounds generation, and as an independent reference against which atomic claims extracted from the justification are checked by deterministic graph queries, without an LLM judge. Each claim receives one of three verdicts: supported, contradicted (a hard hallucination), or unsupported (the graph cannot decide). We evaluate four conditions of increasing graph usage (no knowledge, flat-text context, structured subgraph context, and structured context followed by a verification-driven rewrite) on 332 queries spanning all 32 NHL teams, using a local open-weights model and Polish prompts. Grounding raises faithfulness from 21.5% to 85.5–88.4% and reduces hard hallucinations from 67% to 12–16%; structured context outperforms an information-matched flat-text baseline ($p = 0.012$). The one-pass rewrite is harmful: it does not raise faithfulness, increases the number of contradicted claims by 36%, and lowers recommendation quality ($p < 10^{-5}$). The pipeline, benchmark, and analysis code are publicly available.

Keywords

property graphs, large language models, hallucination detection, retrieval-augmented generation, fact-checking, claim verification, sports analytics

1. Introduction

Large language models (LLMs) are increasingly used as decision-support assistants that recommend an action and justify it in natural language. In data-intensive domains the justification consists largely of numbers: totals, averages, and comparisons that the reader is expected to take at face value. These numbers are frequently wrong. The tendency of LLMs to produce fluent but factually incorrect text is well documented [1, 2], and it is most damaging where the prose is most convincing, because a confident justification built on invented statistics is harder to reject than an unjustified recommendation.

The usual remedy is to ground generation in external data. Retrieval-augmented generation (RAG) [3], its graph-based variants [4, 5], and knowledge-graph prompting [6, 7, 8] place retrieved facts in the prompt and measurably reduce hallucination in question answering. In these pipelines, however, the role of the graph ends once the prompt is built. Whether the model quotes the context correctly, extrapolates beyond it, or mixes it with parametric memory is not checked, because no component compares the generated text with the data. A complementary line of work checks generated text after the fact: FActScore [9] decomposes long-form text into atomic facts and verifies each against Wikipedia, and knowledge-graph retrofitting [10] verifies and revises answers against a triple store. These methods target encyclopedic, entity-relational claims, rely on an LLM to judge each claim, and were evaluated

Graph-enhanced LLMs for trustwOrthy Web data management (GLOW), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy

*Corresponding author.

rmaciejewski2001@icloud.com (R. Maciejewski); r.klopotek@uksw.edu.pl (R. A. Kłopotek)

0009-0008-5510-5594 (R. Maciejewski); 0000-0001-9783-4914 (R. A. Kłopotek)

© 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

CEUR-WS.org/Vol-4313/glow-paper11.pdf

on open-domain question answering. Numeric and comparative claims over a property graph with rich edge attributes, which are typical of decision-support justifications, are not covered, and existing evaluations rarely separate a claim that the reference *refutes* from one that the reference *cannot decide*.

This paper applies the generate-then-verify pattern to that setting and measures what each part of the pipeline contributes. A property graph built from official National Hockey League (NHL) data is used twice. In the generation layer, a retrieved subgraph is serialized into the prompt. In the verification layer, atomic claims extracted from the finished justification are checked by deterministic graph queries that do not depend on what the model was shown; each claim is labelled SUPPORTED, CONTRADICTED (the graph holds the data and the claim is wrong), or UNSUPPORTED (the graph cannot decide). Because the verifier is independent of the generator's context, it can serve both as a measurement instrument and, optionally, as a feedback signal for rewriting the justification. We evaluate four conditions of increasing graph usage under one fixed judge: no external knowledge, an information-matched flat-text context, a structured subgraph context, and the structured context followed by a verification-driven rewrite.

NHL lineup recommendation was chosen as the testbed for three reasons. Ground truth is public and fine-grained: every statistic a justification may cite is available from the official NHL web API. The data is naturally a property graph of players, teams, and games connected by attribute-rich relationships. And the task is generative rather than a lookup: the model must choose and argue, which leaves room for reasoning and for error.

The contributions of this paper are:

- a property-graph instantiation of generate-then-verify in which the verifier is a set of deterministic graph queries over numeric edge attributes, is independent of the retrieved context, and issues three-way verdicts that separate hard hallucinations from unverifiable statements (Sections 4 and 5);
- a four-condition ablation with a metric suite that pairs faithfulness and hard-hallucination rate with two guards, claims per response (against evasion) and top-scorer coverage (against degraded recommendations), and a paired statistical protocol over a deterministic benchmark of 332 queries covering all 32 NHL teams (Section 6);
- empirical findings for one concrete configuration (a local open-weights model, Polish prompts): grounding raises faithfulness from 21.5% to 88.4% ($p < 10^{-83}$), structured context beats an information-matched flat-text dump ($p = 0.012$), and a one-pass verification-driven rewrite is harmful, adding contradicted claims and lowering recommendation quality ($p < 10^{-5}$) (Section 7);
- a reproducible, local-first implementation covering data acquisition, graph construction, generation, verification, benchmark, and analysis, with a graph-store abstraction that runs on Neo4j and on an embedded engine (Section 5.6).

Section 2 states the problem and the scope of the study, Section 3 reviews related work, Sections 4 and 5 describe the data and the system, Sections 6 and 7 present the experimental design and the results, Section 8 discusses limitations and directions, and Section 9 concludes.

2. Problem Overview

2.1. Setting

A decision-support assistant receives a query, returns a recommendation, and justifies it. We study lineup recommendation: a query is a triple $q = (team, opponent?, exclude?)$ asking for a forward lineup for $team$, optionally against a named $opponent$ and optionally excluding one or two unavailable players. The output is a lineup together with a natural-language justification that cites statistics. The justification, not the lineup, is the object of study: which lineup is best is a matter of judgment, but every statistic quoted in its defence is either consistent with the official record or not.

An *atomic claim* is a statement in the justification that a single query against the reference data can decide, for example "player X has at least 30 points this season". A justification is *faithful* to the extent that its atomic claims hold in the reference.

2.2. Failure modes

Four things can go wrong, and a pipeline has to be measured against all four. (i) A claim can be *contradicted*: the reference holds the relevant data and the claim is false. This is a hard hallucination and the quantity a trustworthy system must drive to zero. (ii) A claim can be *unsupported*: it names an entity or a statistic the reference does not hold, so it cannot be decided either way. Such claims may be true, but the reader cannot check them, and a system that produces many of them is escaping its reference rather than using it. (iii) The justification can be *evasive*: a model that makes no checkable statements is trivially faithful. (iv) The recommendation can *degrade*: a model can raise its faithfulness by avoiding strong or risky statements about the players who matter most, which changes the recommendation instead of improving it. Modes (i) and (ii) are properties of individual claims; modes (iii) and (iv) are properties of the whole response and need their own metrics (Section 6.3).

2.3. Reference and its boundary

Verdicts are relative to a declared reference: SUPPORTED means supported by the loaded data, not true in the world. Anything the reference does not hold, such as a season that was not ingested or a statistic that was not derived, is UNSUPPORTED regardless of its real-world status. This is the reference-relative reading of attribution proposed by Rashkin et al. [11]. Stating the boundary explicitly is what makes the contradicted rate interpretable, because it separates "the data disagrees" from "the data cannot tell".

2.4. Scope of this study

The study fixes one generator (Qwen3-30B-A3B, run locally), one domain (NHL skater statistics), one language (Polish prompts and generations, with claim normalization to an English-keyed schema), one rewrite prompt, and one judge (post-hoc claim extraction followed by graph verification). The results are evidence about this configuration. Whether they transfer to other models, languages, or rewrite procedures is an open question, which Section 8 returns to.

3. Related Work

We group prior work into graph-augmented generation, claim-level evaluation of generated text, and knowledge-graph-based correction; Table 1 summarizes the comparison. Hockey analytics itself has an established model-based tradition, for example expected-goals models [12], but it is the source of our data rather than the subject of this paper.

3.1. Graph-augmented generation

Surveys map the many ways in which structured knowledge and LLMs can be combined [13, 14, 5]. KAPING [6] retrieves triples for the entities in a question and prepends them to the prompt. Think-on-Graph [7] lets the LLM explore the graph iteratively and returns the reasoning path. Reasoning on Graphs [8] plans relation paths that are then retrieved from the graph. GraphRAG [4] builds an LLM-extracted entity graph over a corpus and answers from community summaries. These methods reduce hallucination and, in the case of Think-on-Graph and Reasoning on Graphs, make the reasoning traceable. Their limitation for our purpose is twofold: the graph is consulted before or during generation only, so the final text is never compared with it, and the evaluation is answer accuracy (Hits@1, F1, or LLM preference judgments), which does not measure whether the accompanying explanation is faithful. Our generation layer is a plain member of this family (subgraph retrieval plus serialization) and is kept simple on purpose.

Table 1
Related approaches and this work. *Check* states whether the generated text is compared with a reference after generation and what the reference is; *Judge* states how the verdict is produced; KGQA = knowledge-graph question answering; QFS = query-focused summarization.

Approach	Knowledge use at generation	Check / Judge	Verdict	Evaluation	Limitation w.r.t. our setting
KAPING [6]	KG triples in prompt	none	—	answer accuracy on KGQA	output never compared with the KG
Think-on-Graph [7]	iterative KG exploration by the LLM	none (reasoning path is exposed)	—	Hits@1 on KGQA	path shown, not checked; answer-level metric
Reasoning on Graphs [8]	relation-path plans retrieved from KG	none	—	Hits@1, F1 on KGQA	as above
GraphRAG [4]	LLM-built entity graph, community summaries	none	—	LLM-judged preference on QFS	graph is itself LLM-extracted
SelfCheckGPT [15]	none	self-consistency of samples	sentence score	AUC-PR on WikiBio	no external reference
RAGAS [16]	retrieved text in prompt	against retrieved context / LLM	binary	agreement with human labels	reference is what the generator saw
FActScore [9]	optional	against Wikipedia / retrieval + LM	binary	precision of atomic facts (biographies)	text reference, LLM verdict, binary
KGR [10]	KG facts retrieved for draft claims	against KG / LLM, then refine	per claim (LLM)	end-task accuracy on factual QA	LLM verdict; faithfulness of revised text not measured
This work	serialized subgraph in prompt (ablated)	against property graph / deterministic query	three-way	faithfulness, contradicted rate, claims/response, top-scorer coverage	one model, one domain, one language

3.2. Claim-level evaluation of generated text

SelfCheckGPT [15] detects hallucinated sentences from the inconsistency of several sampled responses and needs no external reference, which is also its limitation: a consistently repeated fabrication passes. RAGAS [16] extracts statements from a RAG answer and asks an LLM whether the retrieved context supports them; the reference is the context the generator saw, so an answer that faithfully reproduces a wrong or incomplete context passes, and a system that receives no context cannot be scored at all. FActScore [9] decomposes long-form text into atomic facts and verifies each against Wikipedia with a retrieval-augmented language model, reporting the precision of supported facts on biographies; verdicts are binary. Rashkin et al. [11] formalize attribution to an identified source and underpin the reference-relative reading of verdicts used here. Our verification layer follows the atomic-fact decomposition of FActScore, with four differences: the claims are numeric and comparative, the reference is a property graph rather than text, the verdict is computed by a graph query rather than by an LLM, and a third verdict separates contradiction from lack of coverage.

3.3. Knowledge-graph-based correction

Agrawal et al. [17] survey knowledge-graph-based hallucination mitigation and note that most work targets open-domain question answering. Knowledge-graph retrofitting (KGR) [10] is the closest design to our rewrite condition: claims are extracted from a draft answer, matched with knowledge-graph facts, judged by an LLM, and the draft is refined, with reported gains on factual QA benchmarks. Two differences matter for interpreting our results. KGR's verification is itself an LLM judgment over retrieved facts, whereas ours is a deterministic query; and KGR reports end-task accuracy, whereas we measure the faithfulness of the revised text with a judge that is held fixed across all conditions, including those without any rewrite. This fixed judge is what lets us observe that a one-pass rewrite

Table 2
Property graph schema. Statistics on PLAYED_IN come directly from official boxscores; xg is a reserved slot for a play-by-play-derived model; LINEMATE is a derived co-occurrence proxy.

Element	Type	Attributes
`Player`	node	playerId, fullName, position, sweaterNumber
`Team`	node	teamId, abbrev, name
`Game`	node	gameId, date, season, gameType, scores
`(Player)-[:PLAYS_FOR]->(Team)`	rel.	season
`(Player)-[:PLAYED_IN]->(Game)`	rel.	goals, assists, points, toi, shots, hits, blocked, pim, plusMinus, opponentAbbrev, xg
`(Game)-[:HOME_TEAM|:AWAY_TEAM]->(Team)`	rel.	—
`(Player)-[:LINEMATE]->(Player)`	rel. (derived)	gamesTogether

can lower faithfulness.

4. Data and Graph Model

4.1. Data source

All data comes from the official NHL web API[1] through four endpoint families: daily schedules, per-game boxscores, per-game play-by-play, and team rosters. The boxscore parser was validated field by field against a live game (2025 Stanley Cup Final, game 2025030416): parsed skater goals summed to the official final score, and every extracted attribute (player identity, position, sweater number, goals, assists, points, plus/minus, penalty minutes, hits, blocked shots, shots on goal, time on ice) matched the official record.

4.2. What the graph holds

Boxscores provide the per-game statistics listed above; time on ice is parsed from MM:SS into minutes. Two advanced signals are declared extension points rather than silent approximations, in line with the reference boundary of Section 2.3.

- **Expected goals (xG).** Play-by-play carries shot coordinates, shot type, game strength, and zone for every shot event, which is sufficient input for an xG model in the spirit of [12]. The schema reserves an xg attribute on the player–game edge (NULL by default), so the claim vocabulary already covers xG claims; the model itself is not part of this study.
- **Line chemistry.** Play-by-play does not expose the full on-ice six for every event, so true shared ice time requires the separate shift-chart endpoint. Until that enrichment is added, LINEMATE is a declared proxy: the number of games two players appeared in for the same team.

4.3. Schema

Table 2 summarizes the schema of the property graph [18]. Per-game performance lives on the PLAYED_IN relationship rather than on nodes, which makes per-game claims and season aggregates equally natural to query.

5. System Architecture

Figure 1 shows the pipeline. The graph is used by two layers that share no state. The generation layer (Mode A) retrieves a subgraph and serializes it into the prompt. The verification layer (Mode B) extracts claims from the finished justification and adjudicates each one by querying the graph again; it never

[1] https://api-web.nhle.com/v1; no API key is required.

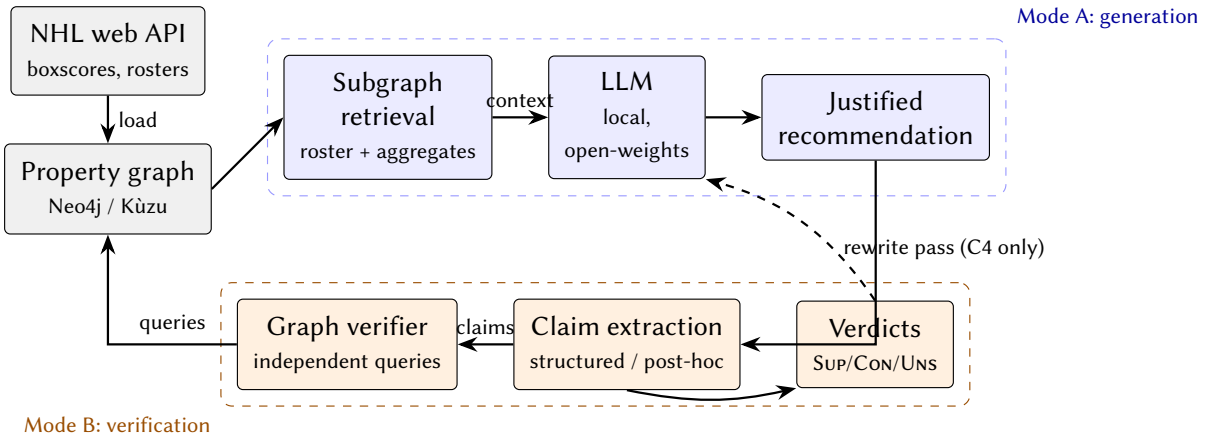


Figure 1: Pipeline. The property graph grounds generation (Mode A) and, separately, adjudicates the claims extracted from the finished justification (Mode B). The dashed edge is active only in condition C4, where failed verdicts are fed back for one rewrite pass.

sees the prompt context, so claims are checked against the graph itself rather than against the excerpt the model received.

5.1. Generation layer

Given a query, retrieval fetches the team roster and, for each eligible player (exclusions removed), season aggregates over a fixed statistic set: totals of points, goals, assists, and shots on goal, and per-game average time on ice. The subgraph is serialized as a compact table (one line per player) and prepended to an instruction that asks the model to recommend a forward lineup and to justify each choice with concrete statistics, using only the provided context. Prompts and generations are in Polish; the normalization layer of Section 5.2 maps Polish surface forms to the claim schema. No English counterpart was run, so the study makes no claim about language effects.

5.2. Atomic claims

Both acquisition strategies (Section 5.4) produce the same intermediate representation: a validated schema of atomic claims, each answerable by one graph query. Four claim kinds are supported: `player_stat` (player X has aggregate $a \in \{\text{total}, \text{avg}\}$ of statistic s standing in relation c to value v), `player_vs_team_stat` (as above, restricted to games against a given opponent), `player_position` (X plays position p, with F accepted as any of C/L/R), and `played_together` (X and Y appeared in games together). Comparators $c \in \{\geq, \leq, =, \approx\}$ have explicit semantics. The approximate comparator, which extractors are instructed to use for rounded or hedged numbers, accepts $|o - v| \leq \tau \cdot \max(|v|, 1)$ with relative tolerance $\tau = 0.05$, so that "around 25 minutes of ice time" is SUPPORTED by an observed 24.7. Statistic names are whitelisted ({goals, assists, points, toi, shots, hits, blocked, pim, plusMinus, xg}), which constrains the extractors and prevents claim fields from injecting arbitrary property names into graph queries. A normalization layer maps surface forms to the schema before validation: Polish and abbreviated statistic names (*punkty*, *gole*, *+/-*), numeric strings with units ("30 pkt"), and aggregation synonyms. Claims that fail validation are dropped and counted. Listing 1 shows a normalized claim and its verdict.

5.3. Verification layer

For each claim the verifier resolves the entities, executes the corresponding semantic graph query, and issues a verdict. Name resolution first attempts an exact match on the full name; failing that, it matches on the surname (final token, at least three characters, dots treated as spaces, because models

Listing 1: A normalized atomic claim (top) and the verdict issued for it by the verification layer (bottom); a contradicted or unsupported claim carries the same structure with the reason recorded.

```
{"kind": "player_stat", "player": "Connor McDavid",
 "stat": "points", "aggregation": "total",
 "comparator": ">=", "value": 30,
 "raw_text": "McDavid ma ponad 30 punktow w sezonie"}

{"status": "SUPPORTED", "observed": "32.000",
 "reason": "Graph: total(points)=32.000 satisfies >= 30"}
```

routinely expand the "C. McDavid" form shown in the context to "Connor McDavid") and disambiguates same-surname candidates by first-name initial. An unresolved entity yields UNSUPPORTED, never a guess. The verdict vocabulary is three-way:

SUPPORTED the graph holds the relevant data and it satisfies the claim;

CONTRADICTED the graph holds the relevant data and it violates the claim (a hard hallucination);

UNSUPPORTED the graph cannot adjudicate: unknown entity, or no data for the requested statistic (for example an xG claim while the xG slot is unpopulated).

Numeric claims are read open-world: absent data yields UNSUPPORTED. `played_together` is read under a local closed-world assumption over the loaded game window: zero co-occurrences within the window count as CONTRADICTED.

5.4. Claim acquisition

Obtaining the claims is the error-prone part of a claim-checking pipeline. The implementation supports two strategies. In *structured generation* the model emits its justification directly as a JSON array of schema-conformant claims; there is no extraction step, at the cost of less natural text and the risk that part of the reasoning is left outside the schema. In *post-hoc extraction* the model first writes free prose, and a second model pass decomposes it into claims under a strict instruction to extract only what is stated, never to guess numbers, and to skip qualitative statements that do not fit the schema; generation stays natural, but the extractor is itself an LLM inference and can miss or distort claims. To cope with the deployed model's observed failure modes, the extractor runs a three-stage cascade: a structured-output call that forces a JSON array (the model otherwise tended to return a single object), then plain JSON mode with lenient coercion (single objects and wrapper keys are rescued), then free-form output from which JSON is mined; the first non-empty result wins, and total failure yields an empty claim list rather than a failed run. All experiments reported here use post-hoc extraction, both as the judge and as the source of claims for the rewrite step. Structured generation is implemented but not evaluated in this paper (Section 8).

5.5. Verification-driven rewrite

In condition C4 the verdicts are fed back once. Claims are extracted from the grounded justification and verified; if any claim is CONTRADICTED or UNSUPPORTED, the model receives its own text together with the failed claims and their verdicts, and is instructed to revise the justification using only confirmed data and without introducing new facts. If all claims pass, the text is kept unchanged. This is a single rewrite round with a single prompt. It tests whether the verdicts, used as feedback, improve the text. It does not test verification in isolation: verification is applied to every condition as the measurement step (Section 6.3).

Table 3
Ablation conditions. Every condition is scored by the same judge: post-hoc claim extraction from the final justification followed by graph verification.

Condition	Run name	Context	Graph-grounded generation	Verdict feedback
C1 no knowledge	`C1_no_knowledge`	none	no	no
C2 flat text	`C2_knowledge_flat`	flat text dump	no	no
C3 structured	`C3_graph_generation`	serialized subgraph	yes	no
C4 C3 + rewrite	`C4_graph_gen_and_verify`	serialized subgraph	yes	yes (one pass)

5.6. Implementation

The pipeline is implemented in Python behind a semantic `GraphStore` interface: the loader and the verifier never emit raw Cypher, and each backend implements a fixed set of semantic operations (upserts plus the small query set needed by the claim checker) in its own dialect. Production runs target Neo4j[2] in Docker Compose; continuous-integration runs execute the identical logic on the embedded engine Kùzu [19], so the test suite needs no server. End-to-end tests check that graph aggregates match ground truth computed independently from fixtures, that verdicts and metrics are correct on known inputs, and that the benchmark loop survives per-call failures. Generation uses Qwen3-30B-A3B [20], a mixture-of-experts model that activates 3B of its 30B parameters per token, served by Ollama[3] at temperature 0.2 on a single NVIDIA DGX Spark workstation. No external LLM API is involved, which matters for reproducibility and for settings in which sensitive graphs cannot leave the premises. Benchmark runs are logged with full per-query artifacts. The complete implementation is available at https://github.com/SpecAgentRM/NHL-Assistant.

6. Experimental Design

6.1. Four conditions of increasing graph usage

The main experiment is an ablation over how much of the graph the system may use (Table 3). C1 measures the parametric prior. C2 is an information-matched flat-text baseline: the same retrieved aggregates as C3, rendered as an unstructured run-on paragraph rather than a table, so that C2 vs. C3 isolates the effect of context *structure* rather than of information access. C4 adds the rewrite of Section 5.5 to C3, so that C3 vs. C4 measures the effect of one verification-driven rewrite pass. All four conditions are scored by the same procedure, described in Section 6.3.

6.2. Benchmark queries

Queries are generated from the graph in four types: *basic* (team only; one per NHL team), *vs-opponent* (against a named rival), *exclude* (one or two named players unavailable), and a combined *vs-opponent with exclusions* type. Because teams and players are drawn from the graph, every query refers to entities the reference contains, so UNSUPPORTED verdicts measure claim quality rather than query malformedness. The full released set contains 1,512 queries spanning all 32 NHL teams. The campaign uses a stratified, seed-deterministic subset of 332 queries (100 per non-basic type plus all 32 basic queries), shuffled so that any prefix stays approximately stratified, with the exact set recorded alongside the results. The subset size follows the power analysis in Section 6.4.

6.3. Measurement protocol and metrics

All four conditions are scored by the same judge: the post-hoc extractor of Section 5.4 decomposes the final justification of each condition, and the verifier of Section 5.3 adjudicates every claim against the

[2] https://neo4j.com, community edition 5.25.
[3] https://ollama.com

graph. Over the resulting verdicts we report:

faithfulness fraction of claims SUPPORTED, the primary trust axis;

contradicted rate fraction CONTRADICTED (hard hallucinations);

unsupported rate fraction UNSUPPORTED;

adjudication coverage fraction the judge could decide either way (SUP + CON), a property of the judge, reported so that faithfulness cannot hide behind an undecidable judge;

claims per response the guard against evasion (failure mode iii): a system that stops making checkable statements is faithful vacuously, so the winning condition must not collapse on this axis;

top-scorer coverage the guard against degraded recommendations (failure mode iv): how many of the team's top-5 scorers by graph-recorded points the justification mentions (surname matching).

Failures of individual LLM calls (timeouts, malformed output) are logged, excluded pairwise, and reported.

6.4. Statistical protocol

Rates are micro-averaged over claims; uncertainty is quantified by bootstrap 95% confidence intervals that resample *queries* (2,000 resamples), the unit of analysis. Differences between adjacent conditions in the hierarchy $\mathrm{C1} \to \mathrm{C2} \to \mathrm{C3} \to \mathrm{C4}$, plus the overall C1 vs. C4 contrast, are tested with an exact two-sided paired sign test on per-query faithfulness, since the same queries are answered under every condition. The effect of the rewrite is additionally quantified as the change in the total number of CONTRADICTED claims from C3 to C4. The query budget was sized for the weakest expected step, C2 vs. C3: a paired sign test at $\alpha = 0.05$ with 80% power needs about 194 untied pairs to detect a 60% win rate and about 85 for 65%. Allowing for an anticipated 30–40% tie rate, 332 queries power the pooled comparisons for small-to-moderate effects; per-type analyses (100 queries per non-basic type) are powered for moderate-to-large effects only and are otherwise reported descriptively.

6.5. Hypotheses

- **H1** Faithfulness increases monotonically with graph usage: $\mathrm{C1} < \mathrm{C2} < \mathrm{C3} < \mathrm{C4}$.
- **H2** The gain of C4 is not bought by evasion or by weaker recommendations: claims per response and top-scorer coverage do not degrade relative to C3.
- **H3** The rewrite removes a substantial fraction of the hard hallucinations present in C3.

7. Results

Table 4 reports the verdict-level metrics with query-level bootstrap intervals and the paired sign tests for all 332 queries, and Figure 2 places every condition on the faithfulness–quality plane. Adjudication coverage, not shown in the table, is 0.888 for C1 and at least 0.995 for the grounded conditions.

Grounding accounts for most of the improvement. The ungrounded baseline behaves as the hallucination literature predicts: only 21.5% of its claims are supported and 67.2% are contradicted, that is, confident numeric statements the graph directly disproves. Most of them concern real players (just 11.2% of claims are unsupported, largely unknown entities): the model knows from pretraining who plays in the NHL (top-5 coverage 47.5%) but invents their statistics. Any form of grounding changes the regime. Faithfulness rises to 85.5% (flat text) and 88.4% (structured subgraph), hard hallucinations fall by a factor of 4–6, and the paired contrasts are unambiguous: grounding wins on 321 (C2) and 323 (C4) of 332 queries against C1 ($p < 10^{-83}$).

Table 4
Main results: four conditions under the same judge, $N = 332$ queries; brackets are 95% bootstrap CIs (2,000 resamples over queries). *faith* = faithfulness, *contra* = contradicted rate, *unsup* = unsupported rate, *cl/resp* = claims per response, *top-5* = top-scorer coverage. The last column is the exact two-sided paired sign test on per-query faithfulness against the preceding condition (wins/losses/ties for the later condition; the C1 vs. C4 contrast is 323/6/3, $p = 3.1 \times 10^{-87}$).

Condition	faith [95% CI]	contra	unsup	cl/resp	top-5 [95% CI]	sign test vs. previous
C1 no knowledge	21.5% [18.5, 24.5]	67.2%	11.2%	22.4	47.5% [45.2, 49.7]	—
C2 flat text	85.5% [84.1, 86.8]	14.3%	0.2%	33.3	**95.2%** [94.1, 96.2]	321/8/3, $p = 5.9 \times 10^{-84}$
C3 structured	**88.4%** [87.0, 89.7]	**11.6%**	0.0%	27.9	88.7% [87.2, 90.1]	154/112/66, $p = 0.012$
C4 C3 + rewrite	83.8% [82.0, 85.4]	15.7%	0.5%	28.1	83.2% [81.4, 85.0]	120/135/77, $p = 0.38$

Context structure has a small, separable effect. C2 and C3 receive the same retrieved aggregates, so the C2→C3 step isolates the effect of structure alone: $+2.9$ points of faithfulness with non-overlapping confidence intervals and a significant paired test (154 wins vs. 112, $p = 0.012$). The effect is second-order next to grounding itself. It is also not free: C3 justifications contain fewer claims (27.9 vs. 33.3 per response) and mention fewer of the top-5 scorers (88.7% vs. 95.2%), so the structured context yields shorter and more selective justifications. The pattern is uniform across the four query families (C3 faithfulness ranges from 87.9% to 89.2% across basic, vs-opponent, exclusion, and combined queries), so no single query type drives the result.

The one-pass rewrite is harmful. Contrary to H1, C4 sits below its uncorrected counterpart C3 on faithfulness (83.8% vs. 88.4%); the paired direction is negative but not significant on its own (120 wins vs. 135, $p = 0.38$). The surrounding evidence is consistent. The absolute number of CONTRADICTED claims *rises* by 36% (1,078 in C3 vs. 1,466 in C4), and per query the rewrite worsens the hard-hallucination count more often than it improves it (136 vs. 109 queries, 87 unchanged). The unsupported rate climbs from 0.0% to 0.5%, so the rewritten text also contains claims the graph cannot decide at all. Recommendation quality drops: top-scorer coverage falls from 88.7% to 83.2%, and the paired test is decisive (55 wins vs. 113, $p = 9.0 \times 10^{-6}$). Claims per response are unchanged (28.1 vs. 27.9), so the failure is not evasion. The model does not delete or fix the flagged statements; it regenerates the justification, and regeneration introduces new errors faster than it removes old ones.

The judge decides almost every claim. Adjudication coverage is at least 99.5% for all grounded conditions (0.888 for C1, where invented entities cannot be resolved). The claim vocabulary, the Polish-language normalization, and surname-based entity resolution are sufficient for the verifier to reach a decisive verdict for practically every extracted claim.

Hypothesis verdicts. H1 holds through C3, where every grounding step is significant, and fails at the C4 step. Of H2, the evasion half passes (claims per response stable), while quality preservation is refuted (-5.5 points of top-scorer coverage, $p < 10^{-5}$). H3 is refuted: the one-pass rewrite adds hard hallucinations on net instead of removing them.

8. Discussion and Limitations

What the results add. Three points follow from the campaign. First, in this setting the graph earns most of its value as context: grounding accounts for almost the entire gain, and structure adds a small but measurable increment. Second, using the verifier as a fixed measurement instrument, applied identically to conditions with and without feedback, is what exposed the regression of C4; a loop that trusted its own rewrite step would have reported success. Third, the negative rewrite result is evidence against the assumption that a verification signal automatically improves the text it is fed back into: KGR [10] reports gains from iterative retrofitting with LLM-judged facts on open-domain QA, whereas

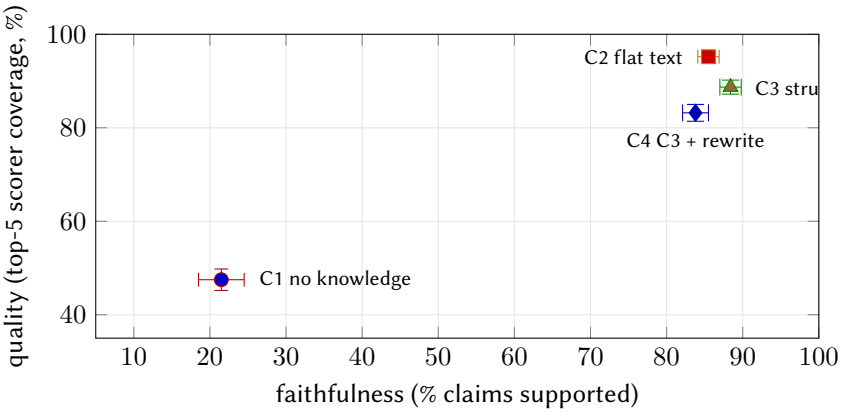


Figure 2: Faithfulness vs. recommendation quality ($N = 332$; error bars are 95% bootstrap CIs, symmetrized). Grounded conditions cluster in the upper right; the ungrounded baseline is far from them on both axes. The rewrite condition C4 trails its uncorrected counterpart C3 on both axes.

one pass with a 30B-class local model and deterministic verdicts made the justification worse. The two settings differ in model, judge, task, and number of rounds, so the contrast motivates further experiments rather than settling the question.

Why the rewrite fails. The rewrite prompt asks the model to *revise* its justification; the model complies by regenerating it, and regeneration introduces contradicted and even undecidable claims at the rate of ordinary generation, while the number of claims stays constant. Natural refinements are to restrict revision to deleting or substituting the flagged claims, to re-verify the revised text and accept it only if it improves (a gated verify-after-revise step), or to iterate to a fixed point; each trades additional LLM calls for convergence. None of these is evaluated here, so the negative result is limited to the one-pass, unconstrained rewrite actually tested.

Oracle circularity. The graph is the context source, the feedback signal in C4, and the measurement judge. C4 therefore optimizes the quantity we measure, which would let a degenerate system win by parroting the context or by saying nothing. The risk is mitigated rather than eliminated: the judge is identical across all conditions, including the two that never see the graph, and the two guard metrics penalize silence and weaker recommendations. The outcome also argues against judge-gaming, since the one condition that directly optimizes the measured quantity ended up below its counterpart on it. A fully independent evaluation (held-out games, human judgments) remains to be done.

The extractor is an error source. Post-hoc extraction is itself an LLM inference and can miss, distort, or invent claims, so the measured faithfulness is a joint property of the generator and the extractor; dropped-claim counts from schema validation are one observable of extractor health. The structured-generation strategy of Section 5.4 bounds the risk from the other side (no extraction step, but constrained generation). Comparing the two strategies on the same queries would check measurement validity and is the most important experiment we have not yet run.

Scope. The results are for one model (Qwen3-30B-A3B), one language (Polish, with no English counterpart, so the effect of language on extraction and verification is unmeasured), one domain, one rewrite prompt, and one judge. LINEMATE is a co-occurrence proxy pending shift-level enrichment. Queries are generated from the same graph that serves as the reference, so there is no temporal hold-out (grounding on window T, questions from $T+1$) that would detect leakage of parametric knowledge. Verdicts are relative to the loaded game window, as stated in Section 2.3.

Directions. On the data side, deriving expected goals from play-by-play shot geometry and shared ice time from shift charts would turn the reserved `xg` slot and the LINEMATE proxy into measurements. On the Semantic Web side, the property graph maps naturally onto RDF with an ontology for players, teams, and games; expressing the claim schema as shape constraints, in the spirit of SHACL, would let the verification layer interoperate with standard tooling and extend verdicts with ontology-based reasoning [14]. On the evaluation side, the priorities are an English-language counterpart of the campaign, other local open-weights generators (to test whether the C3/C4 regression is model-specific), the structured versus post-hoc claim-acquisition comparison, gated multi-round revision, and human assessment of recommendation quality.

9. Conclusion

We described a reproducible pipeline in which a property graph built from official NHL data both grounds LLM-generated lineup recommendations and serves as an independent reference for checking the claims in their justifications, with a three-way verdict that separates hard hallucinations from unverifiable statements. On 332 queries, grounding was decisive (faithfulness $21.5\% \rightarrow 88.4\%$, $p < 10^{-83}$), the structure of the context mattered on its own ($p = 0.012$), and a single verification-driven rewrite was harmful: it added contradicted claims and lowered recommendation quality. The last finding was visible only because measurement was kept separate from the rewrite. For graph-backed decision-support systems we therefore recommend treating verification against a declared reference as a measurement step first, and feeding verdicts back into generation only under a procedure whose effect has itself been measured.

Acknowledgments

This research was carried out at the Faculty of Mathematics and Natural Sciences of Cardinal Stefan Wyszyński University in Warsaw.

Declaration on Generative AI

During the preparation of this work, the authors used Anthropic LLMs for text rewriting with human review of some parts of this article. The authors take full responsibility for the content of this publication.

References

[1] 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) 248:1–248:38. doi:10.1145/3571730.

[2] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu, A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, ACM Transactions on Information Systems 43 (2025) 42:1–42:55. doi:10.1145/3703155.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, pp. 9459–9474.

[4] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, J. Larson, From local to global: A graph RAG approach to query-focused summarization, 2024. doi:10.48550/arXiv.2404.16130, arXiv:2404.16130.

[5] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, S. Tang, Graph retrieval-augmented generation: A survey, 2024. doi:10.48550/arXiv.2408.08921, arXiv:2408.08921.

[6] J. Baek, A. F. Aji, A. Saffari, Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, 2023. doi:10.48550/arXiv.2306.04136, arXiv:2306.04136; presented at the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE@ACL 2023).

[7] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, J. Guo, Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, in: The Twelfth International Conference on Learning Representations (ICLR 2024), 2024.

[8] L. Luo, Y.-F. Li, G. Haffari, S. Pan, Reasoning on graphs: Faithful and interpretable large language model reasoning, in: The Twelfth International Conference on Learning Representations (ICLR 2024), 2024.

[9] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, H. Hajishirzi, FActScore: Fine-grained atomic evaluation of factual precision in long form text generation, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), Association for Computational Linguistics, Singapore, 2023, pp. 12076–12100. doi:10.18653/v1/2023.emnlp-main.741.

[10] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, L. Sun, Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 18126–18134. doi:10.1609/aaai.v38i16.29770.

[11] H. Rashkin, V. Nikolaev, M. Lamm, L. Aroyo, M. Collins, D. Das, S. Petrov, G. S. Tomar, I. Turc, D. Reitter, Measuring attribution in natural language generation models, Computational Linguistics 49 (2023) 777–840. doi:10.1162/coli_a_00486.

[12] B. Macdonald, An expected goals model for evaluating NHL teams and players, in: Proceedings of the 2012 MIT Sloan Sports Analytics Conference, Boston, MA, USA, 2012.

[13] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 3580–3599. doi:10.1109/TKDE.2024.3352100.

[14] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, 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) 71:1–71:37. doi:10.1145/3447772.

[15] P. Manakul, A. Liusie, M. J. F. Gales, SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), Association for Computational Linguistics, Singapore, 2023, pp. 9004–9017. doi:10.18653/v1/2023.emnlp-main.557.

[16] S. Es, J. James, L. Espinosa-Anke, S. Schockaert, RAGAs: Automated evaluation of retrieval augmented generation, in: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (EACL 2024), Association for Computational Linguistics, St. Julians, Malta, 2024, pp. 150–158. doi:10.18653/v1/2024.eacl-demo.16.

[17] G. Agrawal, T. Kumarage, Z. Alghamdi, H. Liu, Can knowledge graphs reduce hallucinations in LLMs? A survey, in: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 3947–3960. doi:10.18653/v1/2024.naacl-long.219.

[18] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: Proceedings of the 2018 International Conference on Management of Data (SIGMOD 2018), 2018, pp. 1433–1445. doi:10.1145/3183713.3190657.

[19] G. Jin, X. Feng, Z. Chen, C. Liu, S. Salihoglu, Kùzu graph database management system, in: 13th Conference on Innovative Data Systems Research (CIDR 2023), Amsterdam, The Netherlands,

2023.
[20] Qwen Team, Qwen3 technical report, 2025. doi:10.48550/arXiv.2505.09388, arXiv:2505.09388.