Hallucinations on the Board: Tool-Augmented Evaluation of LLM Chess Commentary

Anonymous ACL submission

Abstract

Superhuman game engines in domains like chess have made expert-level evaluations easily accessible, yet they communicate *what* is true without the natural-language *why* that makes such expertise educationally useful to expert and non-expert alike. Large language models could, in principle, bridge this gap, but they frequently hallucinate due to a lack of deep domain-specific knowledge, which standard, reference-based LLM-as-a-judge frameworks cannot reliably detect. In this work, we present **ACT-Eval**, an evaluation framework that decomposes chess commentary into atomic claims and routes each to engine-supported tools and expert-annotated gold references for factual verification and quality check. We release a benchmark of 325 positions across pedagogical, tournament, and critical positions, with expert-verified gold reference and a five-class error taxonomy. Evaluating frontier and open-source models, we find that factual hallucinations remain pervasive in chess commentary: GPT-5.4 without tools produces factually incorrect claims 22.0% of the time, while smaller open-source models exceed 50%. Although tool augmentation substantially improves factual correctness and move-quality assessment, coverage of expert strategic and tactical ideas remains limited across all models. Human calibration studies show that ACT-Eval's factual judgments align closely with expert annotators, while its coverage scores correlate strongly with human assessments of strategic completeness.

1 Introduction

Chess has long served as a canonical domain for studying strategic reasoning. Its appeal lies not only in the combinatorial complexity of the game, but also in the rich explanatory tradition that has developed around it. For centuries, chess understanding has been communicated through annotated games, books, lectures, broadcasts, and, more recently, online platforms such as YouTube and Twitch. Strong players make moves intelligible by translating them into plans, threats, tactical motifs, and long-term tradeoffs. The enduring popularity of these formats reflects a simple reality: players and spectators do not only want to know which move is best, but why it works.

Superhuman engines such as Stockfish (The Stockfish developers, 2024; Silver et al., 2016) have transformed this landscape by making expert-level evaluation broadly accessible. On one hand, this creates an unprecedented opportunity for democratizing expertise: even a novice can consult an engine to instantly know whether a move is a blunder or a brilliancy. However, they only provide an objective score, which for beginner and intermediate-level players may not mean much more than determining who is better; even elite grandmaster preparation involves spending hours interpreting engine play into human explanations and practical plans. Engines have largely solved the problem of finding strong moves, but not the problem of explaining them.

The rapid development of Large Language Models (LLMs) offers a promising direction. LLMs excel at multi-step reasoning across many domains (Wei et al., 2022; OpenAI, 2024) and can generate reasoning traces in natural language that humans can understand. This creates the possibility to bridge the gap between superhuman expertise and educational explanations. However, substantial challenges remain before LLMs can reliably fulfill this role in domain-specific reasoning tasks. Taking chess as an example, even state-of-the-art reasoning LLMs still perform far below expert human players in chess-specific reasoning, let alone specialized engines (Kolasani et al., 2025; Wen et al., 2025). Moreover, the spatial structure and long-horizon temporal complexity of chess amplify hallucination issues: models frequently misidentify piece locations, threats or propose

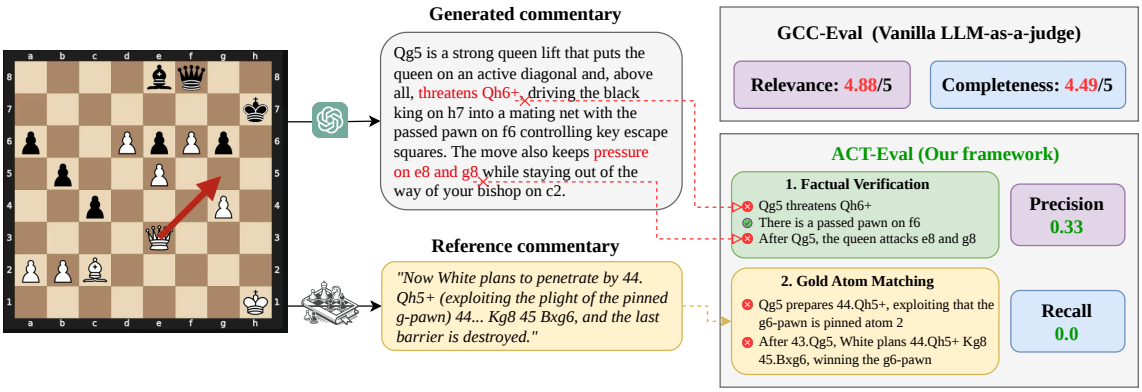

Figure 1: Comparison between GCC-Eval and ACT-Eval on generated chess commentary. The commentary in this example is fluent, but not faithful to the underlying chess position. Nevertheless, GCC-Eval, a naive LLM judge assigns near-perfect relevance and completeness scores. ACT-Eval instead decomposes the commentary into atomic claims, verifies factual claims using chess tools and engine analysis, and measures coverage against expert reference atoms. Here, it correctly determines that only one of three factual claims is correct, and none of the key expert ideas are covered.

illegal move sequences, resulting in commentary that often sounds plausible despite containing factual errors (Figure 1).

Compounding these generation challenges is a fundamental evaluation problem: there is currently no reliable way to measure how well a language model understands specific domains like chess or how accurately it explains the move. Existing benchmarks primarily focus on verifiable capabilities such as puzzle-solving accuracy and Elo-style playing strength (Kolasani et al., 2025; Wen et al., 2025). Standard text generation metrics (Papineni et al., 2002; Lin, 2004) only measure surface similarity to reference commentary. Recent LLM-as-a-judge approaches (Liu et al., 2023; Kim et al., 2025), which use large language models to assess explanations, introduce a circular problem: the judge model often exhibits the same hallucinations as the model being evaluated, making it an unreliable arbiter of domain-specific facts (Szymanski et al., 2025). As Figure 1 illustrates, a hallucinated commentary receives 4.9/5 from GCC-Eval (Kim et al., 2025) but 2 out of 3 main claims contain factual errors.

In this work, we move toward addressing this gap by proposing an evaluation framework for assessing LLM understanding and interpretability in domain-specific reasoning tasks, using chess as a testbed. A key property of chess is that many factual claims such as move legality, attacks, material balance, or board state can be deterministically checked using chess engines and symbolic tools (The Stockfish developers, 2024; Leela Chess Zero; Fiekas, 2022), whereas higher-level explanations about plans, coordination, or positional ideas cannot be fully reduced to computation alone. Built on this observation, we propose **ACT-Eval** (**A**tomi**C** and **T**ool-augmented Evaluation), an atom-grounded, tool-augmented framework for evaluating chess commentary. ACT-Eval decomposes commentary into atomic claims and uses LLMs to route each claim to deterministic tools and check factual correctness. In addition, the LLM judge evaluates strategic and explanatory claims to measure coverage through atomic recall against expert-annotated gold references, capturing whether the explanation identifies the key ideas a strong player would recognize.

Our contributions are summarized as follows:

- We introduce ACT-Eval, an evaluation framework that integrates atomic-claim decomposition with tool-augmented verification for chess commentary. By routing computationally decidable claims to engine and board-state oracles, ACT-Eval removes the LLM judge from the critical path wherever verifiable evidence can be extracted or inferred, and restricts judge reasoning to claims that genuinely resist computational checking.

- We construct a benchmark of 325 positions, covering pedagogical textbooks, modern tournament and meticulously-engineered critical positions. Central to this benchmark is a

subset of positions with expert-verified gold atoms derived from grandmaster-level human annotations. They serve as reference for evaluating conceptual coverage – whether generated commentary captures the strategic and tactical insights behind a move – as a complementary dimension to factual correctness.

- We evaluate frontier and open-source LLMs and show that factual hallucinations remain pervasive in chess commentary, particularly in tactical variations and long move sequences. Tool augmentation substantially improves factual correctness and move-quality assessment, but atomic recall against expert annotations remains limited. Human calibration studies further show that ACT-Eval achieves agreement with expert annotators within the range of inter-human agreement.

Using chess as a testbed for domain-specific explanation, we show that current frontier LLMs fail to translate their strong general reasoning into reliable domain-specific expertise, similar gaps likely exist in other specialized domains too, where existing evaluations cannot reliably detect it. Broadly, ACT-Eval demonstrates a methodological template for evaluating expertise representation in other domains wherever part of the truth is computable and part requires expert judgment. We release the framework and benchmark[1], hoping they serve as a foundation for future work on improving the reasoning quality, interpretability, and educational usefulness of expert AI systems through LLMs.

2 Related Work

Reliable evaluation of generated text has been approached from three directions: similarity to human references (Papineni et al., 2002; Lin, 2004; Zhang et al., 2019), factual verification against a knowledge source (Min et al., 2023; Chern et al., 2023; Wei et al., 2024), and LLM-as-a-judge scoring (Liu et al., 2023). When it comes to domain-specific text generation like chess, each falters for a different reason. Similarity metrics reward fluent text regardless of correctness. Factuality methods assume claims can be verified by retrieval, but verifying a chess claim requires computing against the live position rather than retrieving against a knowledge base. LLM judges share the hallucinations of the model under evaluation (Szymanski et al., 2025), so a fluent but wrong chess claim looks correct to a fluent but equally unreliable judge. The closest prior work in our setting, GCC-Eval (Kim et al., 2025), augments an LLM judge with reference commentary and engine analysis but inherits both noisy references and uninterpreted engine scores. ACT-Eval addresses these limitations by routing factual claims to deterministic chess tools and scoring conceptual coverage against expert-verified gold atoms; we provide an extended discussion in Appendix B.

3 The ACT-Eval Framework

3.1 Background

Chess commentary evaluation. We consider the task of evaluating the quality of chess move commentary. Formally, given a board state s represented in a standard encoding (FEN or ASCII) and a move a played in that state, a commentary $\mathcal{C}$ is a natural language explanation that explains the strategic or tactical purpose of the move. Our goal is to assess the quality of $\mathcal{C}$ along three dimensions: *(1) Factual correctness*, the claims made in $\mathcal{C}$ about the current or subsequent states should be factually correct with respect to the board state. For example, claims about piece placement, attacks, threats, checks, captures, legal moves, and resulting positions should be correct with respect to s and a; *(2) Evaluation accuracy*, $\mathcal{C}$ should correctly characterize the quality of the move a, distinguishing, for example, a strong positional choice from an inaccuracy or blunder; *(3) Conceptual coverage*, $\mathcal{C}$ should identify the key strategic and tactical themes that a human expert would consider most salient in the position.

Why evaluation is hard. Assessing whether the commentary satisfied these properties is non-trivial. Existing reference-based, LLM-as-a-judge approaches (Kim et al., 2025) suffer from two major limitations. First, the reference data the judge relies on are often noisy, containing stylistic artifacts and incomplete analysis. Second, and more critically, LLM judges are insensitive to factual errors: a fluent, chess-specific but *wrong* claim will receive a high score, because the judge exhibits the same domain-specific hallucinations as the model being evaluated (see detailed analysis in Appendix A.4).

To address these limitations, we propose **ACT-Eval** (**A**tomi**C** and **T**ool-augmented Evaluation),

[1] https://anonymous.4open.science/r/act-eval-paper-F5C1/README.md

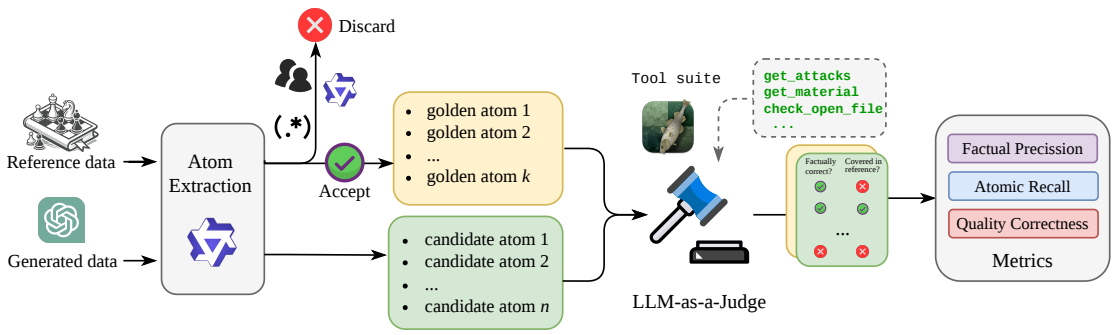

Figure 2: Evaluation Pipeline.

a framework that combines atom-grounded expert annotations as gold reference and tool-augmented fact verification to substantially reduce LLM hallucination.

3.2 Golden Atom Extraction

We first collect reference data from canonical chess books and online forums. Since the quality and accuracy of these sources are highly diverse, we employ regex-based filters and LLMs for preliminary filtering and decomposition, and then human experts conduct rigorous quality review. We apply filters that remove low-content annotations (bare move notation like "19. Nd5+ Bxd5 Qxd5", single-word evaluations like "Forced.", and generic filler phrases like "Obviously" among others). These filters drop approximately 15–30% of raw annotations before LLM processing, ensuring that only substantive commentary reaches the extraction stage. We then use an LLM with few-shot demonstrations to extract factual move explanations from the filtered commentary. If a piece of reference commentary contradicts objective engine evaluation (e.g., claiming a blunder is a good move), the annotation is rejected with the rationale. Similarly, we reject positions where the commentary provides no concrete positional analysis.

From accepted examples, we process the data into a standardized format by extracting factual atoms and stripping stylistic noise. High-quality human annotations often include multi-layered strategic and tactical insights. To standardize the data for evaluation, we parse the filtered reference annotations into a list of "golden atoms". Each atom is a discrete, position-specific factual statement. It is critical that each atom is contextual, i.e., carrying the complete sequence from the root position, to remain independently verifiable.

While strong language models can evaluate text fluency and coherence, they exhibit significant weaknesses in chess-specific knowledge. Our preliminary experiments showed that even state-of-the-art LLMs frequently hallucinate chess facts: claiming pieces exist on squares where they don't, inventing illegal moves, misinterpreting tactical patterns, and incorrectly assessing position quality. This fundamental limitation makes pure LLM-based evaluation unreliable for chess commentary.

To address this, we develop a **tool-augmented verification framework** that grounds fact-checking in deterministic chess engines and programmatic board-state queries. Rather than relying on the judge LLM's internal (and often faulty) chess knowledge, we require it to verify claims by calling specialized tools that provide ground-truth information.

3.3 Verification Architecture

Our fact-checking pipeline operates in three stages (Figure 2):

Stage 1: Atomic Claim Decomposition The generated commentary is first decomposed into **atomic claims** (atoms), following the same principles as golden atom extraction (Sec. 3.2). Each atom is a position-specific factual statement, with full positional context. To enable tool-augmented evaluation, the LLM further breaks down each candidate atom into one or more **verifiable sub-claims** - specific facts that can be independently verified by tools. For example, the candidate atom "Qe4 centralizes the queen and attacks the bishop on b4" decomposes into two sub-claims: (1) "Qe4 centralizes the queen", (2) "Qe4 attacks the bishop on b4". We use a lightweight classifier to assign semantic tags (eg. quality, positional, move continuation) to each sub-claim.

Stage 2: Tool-Augmented Verification Each atomic claim is verified in a single judge call using a specialized prompt conditioned on the extracted tags and equipped with **mandatory access to chess tools**. Within this process, the judge evaluates all associated sub-claims before producing a final atom-level verdict. By grounding verification in deterministic tool outputs rather than relying on parametric model knowledge, we substantially reduce chess-specific hallucination during factual verification. An atomic claim is marked *correct* only if all constituent sub-claims are verified as true; if any sub-claim is incorrect, the entire candidate atom is marked incorrect.

Tool Suite We implement a suite of 15+ chess-specific verification tools spanning board-state queries, attack/defense analysis, move simulation, and engine evaluation; the full tool list is provided in Appendix A.2. The combination of these tools enable the judge to ground factual verification in deterministic outputs rather than internal chess knowledge. We demonstrate an example with one atomic claim below:

Multi-Tool Verification Example

Generated commentary: "Qe4 centralizes the queen and creates a dual threat: it attacks the bishop on b4 and prepares to infiltrate Black's position via the e-file."

Atomic Claim 1: Qe4 centralizes the queen and creates a dual threat: it attacks the bishop on b4 and prepares to infiltrate Black's position via the e-file. [× Incorrect - 1/3 sub-claims verified]

✓ "Qe4 centralizes the queen" — `get_piece_at("e4")`, `get_squares("queen")`, confirm queen moved from b1 to central e4. **Verified.**

× "Qe4 attacks the bishop on b4" — `get_attacks("e4")` → `["b1", "c2", "c4", "d3", ...]` does not include b4. **Incorrect.**

× "Qe4 prepares to infiltrate via the e-file" — `get_attacks("e4")` shows e-file blocked by Black's e5 pawn; no penetration possible. **Incorrect.**

Result: Tool-based verification catches hallucinations (false claims about attacking b4 and e-file infiltration) that pure LLM evaluation would miss.

Stage 3: Gold Atom Matching After verification, the judge compares verified atoms from the generated commentary against the golden atoms (Sec. 3.2) to measure coverage of important positional concepts.

3.4 Evaluation Metrics

Our evaluation computes three complementary metrics from the verification pipeline:

- **Factual Precision:** The fraction of atomic claims verified as factually correct. An atomic claim is factually correct if and only if all of its sub-claims are verifiably correct. Claims that are purely strategic or otherwise not verifiable through tool calls are excluded. We also report sub-claim error rate, the fraction of verifiable sub-claims that fail tool-based verification.
- **Atomic Recall:** The fraction of golden atoms (from reference annotations) that are covered by the generated commentary, computed from Stage 3 matching. This measures coverage - whether the model identifies important positional concepts.
- **Quality Correctness:** A binary indicator of whether the commentary's move-quality assessment (good / inaccurate / blunder) matches the engine-derived label. This measures whether the model accurately assesses position evaluation.

4 Experiments

4.1 Experimental Setup

Dataset	Size	Source	Gold Atoms
Textbook	75	Chernev (2003)	✓
Candidates50	50	Lichess-Study	✓
Critical	200	Lichess+Maia2	×
Total	**325**		

Table 1: Evaluation datasets. Textbook and Candidates50 include expert-verified gold atoms and support all three metrics; Critical positions have no reference annotations and evaluate factual correctness and move quality only.

Datasets. Table 1 summarizes the three evaluation datasets (n=325 total). Textbook and Candidates50 include expert-verified gold atoms and support all three metrics; Critical positions are generated programmatically using Maia2 (Tang et al., 2024) and evaluate only factual correctness and move quality. Dataset construction details are provided in Appendix A.3.

Models. We evaluate four models: GPT-5.4, Claude Opus 4.7, Qwen3-32B, and Qwen3-8B. For GPT-5.4 and Claude Opus 4.7, we test both

with and without access to chess tools during generation. Qwen models are run in No-Think mode (reasoning traces disabled), as preliminary experiments showed that the thinking mode produced degenerate outputs.

Commentary generation. All models are prompted to act as a chess instructor explaining a move to an intermediate player. The model receives only the board state (as a FEN string) and the move played. When tool access is enabled, the model may invoke the Stockfish-backed chess tool suite (Table 3) to analyze the position before writing the explanation. When tool access is disabled, the model must rely entirely on its internal chess knowledge. All generations use temperature 0.3 with the same base system prompt, which instructs concise, position-specific prose without filler or generic commentary. The full prompt is reproduced in Appendix C.

Evaluation pipeline. All generated commentary is evaluated using the ACT-Eval framework (Section 3) with GPT-5.4 as the judge model, equipped with the full tool suite. Stockfish 14.1 at depth 18 serves as the backend engine for all tool calls.

4.2 Main Results

Table 2 presents results aggregated across all three datasets (n=325 position-move pairs). We found that providing tool access to models substantially improves factual correctness and move-quality assessment, but atomic recall against expert annotations remains low across all models.

LLMs hallucinate frequently in chess commentary. Without tool access, even frontier models produce factually unreliable commentary. GPT-5.4 fails verification on 22.0% of sub-claims; Claude Opus 4.7, 20.5%. Smaller models fare far worse: Qwen3-32B and Qwen3-8B fail on 53.8% and 59.0% of sub-claims, respectively. These results are consistent with the well-documented finding that LLMs lack reliable chess-playing ability (Kolasani et al., 2025): generating good commentary requires not just fluency but the ability to identify key ideas in a position, which in turn demands accurate board comprehension - recognizing piece locations, legal moves, tactical patterns, and threats. Errors in this basic spatial reasoning propagate directly into the explanations.

Tool access substantially improves factual correctness. Equipping models with chess tools, including Stockfish evaluation and variations, legality checks, piece location queries, and attack/defense analysis - mitigates a large share of these errors. GPT-5.4's sub-claim error rate drops from 22.0% to 9.7% with tool access, more than halving the rate of verification failures. Claude Opus 4.7 shows a similar improvement, from 20.5% to 13.8%. The key insight is that tools disentangle chess-playing ability from chess explanation ability. LLMs need not internally compute whether a square is attacked or a move is legal, they can offload this to deterministic tools and focus on what they are better suited for: structuring coherent, contextual explanations around verified facts.

Quality assessment benefits from engine access. Without tools, models frequently misjudge move quality: GPT-5.4 correctly categorizes moves only 80.0% of the time, and Claude Opus 4.7 achieves 82.2%. With engine access, both models reach near-perfect quality correctness (99.1% and 97.3%, respectively). This is expected: with direct access to Stockfish evaluations, models can calibrate their assessments to objective analysis rather than relying on their own (often faulty) positional intuition. Smaller models without tool access perform notably worse (Qwen3-32B: 65.8%, Qwen3-8B: 69.3%), further underscoring the difficulty of move quality assessment from internal knowledge alone.

Recall remains low across all models. Even the best configuration, Claude Opus 4.7 with tools, covers only 58.3% of expert-identified gold atoms. GPT-5.4 with tools achieves 44.0%, and the Qwen models fall below 16%. This indicates that current LLMs miss a substantial fraction of the positional and tactical ideas that a human expert would consider important. The effect of tools on recall is model-dependent: GPT-5.4's recall is essentially unchanged by tool access (44.0% vs. 44.6%), suggesting that tools help it say things *correctly* without changing *what* it chooses to say. Claude Opus 4.7, by contrast, shows a notable recall increase with tools (45.3% $\rightarrow$ 58.3%), suggesting that access to verified facts enables it to build on confirmed observations and surface additional themes. Regardless, the low absolute recall across all models highlights that identifying the critical ideas in a chess position remains a significant challenge.

Table 2: **Combined ACT-Eval Results** across all datasets (n=325). *Error Rate* is the fraction of sub-claims that fail verification; *Precision* is the fraction of atoms that are factually correct; *Recall* measures the fraction of gold reference atoms covered by the generated commentary; and *Quality* calculates the fraction of correctly categorized commentary (see Section 3.4 for detailed metric definitions). Recall is computed only on datasets with gold reference annotations (n=125). Our human calibration experiments indicate that both the error rate, and recall are lower bounds. NT = No-Think mode.

Model	Tools	Err Rate	Precision	Recall	Quality
GPT-5.4	✓	**9.7%**	53.9%	44.0%	**99.1%**
	×	22.0%	51.0%	44.6%	80.0%
Claude Opus 4.7	✓	13.8%	53.6%	**58.3%**	97.3%
	×	20.5%	**54.9%**	45.3%	82.2%
Qwen3-32B (NT)	×	53.8%	23.3%	16.0%	65.8%
Qwen3-8B (NT)	×	59.0%	19.7%	13.9%	69.3%

5 Human Calibration

To validate that the ACT-Eval judges align with expert human assessment, we conduct two calibration studies targeting the two judge-dependent metrics: (i) factual correctness (Stage 2) and (ii) atomic recall against gold references (Stage 3).

Participants. Four chess players with Lichess blitz ELO between 1800-2200 participated in the study. All 4 completed Study 1, while 2 completed Study 2. Annotators worked through a purpose-built interface that presented the board diagram, FEN, and engine analysis alongside the LLM commentary or claim under review, and they were blinded to the judge's verdicts.

5.1 Study 1: Claim Verification

We assess whether the judge's per-atom verdict agrees with expert human judgment. We draw 50 atomic claims from the evaluation corpus using stratified sampling across different models and judge verdicts (30% judge-correct, 70% judge-incorrect; fixed seed) and place them in randomized order. For each item, annotators (a) issued a verdict on a four-point scale (Correct / Mostly correct / Incorrect / Cannot determine) and (b) reported confidence (low/medium/high).

Since the judge produces binary labels and marks an atom incorrect whenever any sub-claim fails verification (Section A.2), mostly correct human verdicts are mapped to incorrect for agreement analysis, consistent with the judge's own aggregation rule.

Judge agreement approaches inter-human variability. We measure agreement between the judge's binary verdict and human annotators on the 50 sampled claims. Per-annotator agreement with the judge ranges from 69% to 83%, with a macro-average of 74.5% across the four annotators. Pairwise agreement between annotators ranges from 73% to 91%. This indicates that the judge falls within the range of inter-human agreement on the task.

We additionally report pairwise Cohen's κ, which measures agreement corrected for chance: values range from 0.107 to 0.697 across annotator pairs, and Fleiss' κ (the multi-annotator extension) is 0.285. Both fall in the *fair* range by standard conventions; the wide pairwise range partly reflects a statistical artifact : the sample is 70% incorrect by design, which inflates expected chance agreement and suppresses κ irrespective of true disagreement.

The judge is conservative but reliable at detecting errors. When the judge marks a claim *incorrect*, annotators agree 83–94% of the time: errors flagged by the judge are almost always real errors. When it marks a claim *correct*, agreement drops to 14–57%. The judge is therefore reliable at catching errors but misses some that human experts identify, meaning the atom error rates in Table 2 are themselves lower bounds on the true error rate.

5.2 Study 2: Commentary Coverage

We evaluate whether the judge's recall metric aligns with expert human assessment over 25 position–move pairs. The task proceeds in two phases. In the *completeness check*, annotators are shown the gold atoms and judge whether they cover the most important observations about the move (fully / partially / no). Annotators can optionally add missing key points, which are appended to the gold set. This provides a sanity

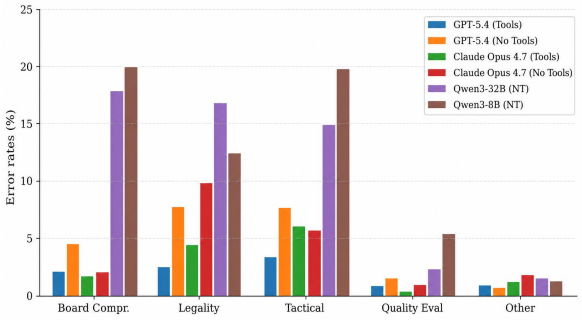

Figure 3: Factual error breakdown by category across models, normalized by total decisive sub-claims.

check on the quality of our extracted gold atoms.

Next, in the *coverage annotation*, annotators are shown an LLM-generated commentary (sampled across models) and indicate, for each gold atom (including any additions), whether it is covered by the commentary. Human recall is computed as the fraction of (augmented) gold atoms judged covered.

Gold atoms serve as adequate coverage targets. Gold atoms are extracted from grandmaster commentary, filtered, and decomposed via the pipeline of Section 3.2. Across the 25 sampled positions, the two annotators rated the gold set as fully complete in 92% and 88% of cases respectively; the remaining items were marked partially complete, typically with a single additional point appended. This validates the extracted gold atoms as adequate coverage targets.

Judge recall correlates strongly with expert humans. The judge's per-position recall correlates with the two annotators at Pearson $r = 0.732$ and $r = 0.749$. The annotators themselves agree at $r = 0.835$, indicating that the judge-human gap is small relative to the inherent variability of the task.

The judge applies a stricter coverage threshold than human experts: mean judge recall over the sample is 0.33, compared to 0.41–0.47 for the two annotators, with the gap consistent across positions. Recall scores in Table 2 should accordingly be read as lower bounds on true coverage.

6 Analysis

6.1 Error Analysis

Figure 3 breaks down factual errors across five categories: board comprehension, legality, tactical reasoning, quality evaluation and others (defined in Appendix B.1), normalized by the total number of decisive sub-claims. Smaller open-source models are dominated by board comprehension and legality failures. These include hallucinating pieces on incorrect squares, misidentifying attacks and defenders, and proposing illegal continuations. Such errors indicate weak internal representations of the board state itself.

Frontier models substantially reduce these static board-understanding failures, but still exhibit frequent errors in multi-move variations when tool access is disabled. In particular, GPT-5.4 and Claude Opus 4.7 often generate plausible-looking continuations that become inconsistent after several moves. This suggests that, despite strong linguistic fluency, LLMs struggle to reliably maintain and update an internal chess state over long-horizon tactical sequences.

Tool augmentation shifts the error distribution significantly. Access to legality checks, board-state queries, attack analysis, and engine evaluations largely eliminates many low-level hallucinations about piece locations and legal moves. As a result, the remaining errors are concentrated in tactical reasoning: models incorrectly infer whether a tactical idea works, overlook defensive resources, or hallucinate forcing continuations. Quality-evaluation errors become relatively rare with tool access, consistent with the large Quality Correctness gains in Table 2.

6.2 Ablation Study on Tool Augmentation

ACT-Eval combines atomic claim decomposition with tool-augmented verification. To isolate the contribution of tool augmentation, we keep decomposition fixed and evaluate the same sub-claims using GPT-5.4 without tool access, forcing the judge to rely entirely on its internal chess knowledge.

We evaluate on 29 claims from the S1 study (Section 5) on which all four annotators agreed *incorrect* with high confidence : this is a clean set of unambiguously confirmed errors. On this set, ACT-Eval misses 21% of confirmed errors, compared to 48% for the no-tool ablation. This significant gap confirms that the addition of tool augmentation substantially improves the LLM judge. The 21% miss rate also reinforces our finding that ACT-Eval's reported atom error rates are themselves lower bounds; expanding the tool suite and refining verification prompts are natural avenues for closing the remaining gap.

Limitations

Tool-suite coverage. Our verification framework relies on a fixed set of chess-specific tools. Human calibration shows that ACT-Eval still misses some factual errors. These missed errors are not confined to a single category; they occur across board comprehension, legality, tactical reasoning, and evaluation-related claims. Reported atom error rates should be interpreted as lower bounds on the true error rate. Expanding the suite, particularly for long-horizon variation checking and endgame-specific patterns, is a natural direction for closing this gap.

The framework still inherits LLM-judge limitations. Although tools reduce reliance on the judge's internal chess knowledge, the judge is still responsible for interpreting claims, selecting tools, and applying prompt guidelines. In practice, it may occasionally misread the claim, ignore instructions, or apply the verification rubric inconsistently, making calibration an ongoing challenge.

LLM-based decomposition can introduce errors. ACT-Eval relies on an LLM to split commentary into atomic claims. This step can produce incorrect or under-contextualized atoms, especially when claims depend on earlier sentences, implicit move sequences, or surrounding tactical context. If an atom loses necessary context, the verification stage may incorrectly reject it.

Gold atoms are incomplete approximations of expert understanding. The expert-verified gold atoms capture many salient tactical and strategic ideas, but chess explanations are open-ended and multiple valid analyses may exist for the same move. Atomic recall should therefore be interpreted as coverage relative to our annotation set, not as an absolute measure of all possible expert insight.

References

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, and 1 others. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Irving Chernev. 2003. *Logical Chess: Move by Move: Every Move Explained*. Batsford, London.

Esin Durmus, He He, and Mona Diab. 2020. Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 5055–5070.

Alexander Richard Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. Qafacteval: Improved qa-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601.

Niklas Fiekas. 2022. python-chess: a chess library for Python. GitHub repository.

Jaechang Kim, Jinmin Goh, Inseok Hwang, Jaewoong Cho, and Jungseul Ok. 2025. Bridging the gap between expert and language models: Concept-guided chess commentary generation and evaluation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9497–9516.

Sai Kolasani, Maxim Saplin, Nicholas Crispino, Kyle Montgomery, Jared Quincy Davis, Matei Zaharia, Chi Wang, and Chenguang Wang. 2025. Llm chess: Benchmarking reasoning and instruction-following in llms through chess. *arXiv preprint arXiv:2512.01992*.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 9332–9346.

Leela Chess Zero. Leela chess zero: Open source neural network based chess engine. https://lczero.org/. Accessed: 2026-05-25.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. 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*, pages 12076–12100, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Learning to reason with LLMs. OpenAI o1 system card and technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Annalisa Szymanski, Noah Ziems, Heather A Eicher-Miller, Toby Jia-Jun Li, Meng Jiang, and Ronald A Metoyer. 2025. Limitations of the llm-as-a-judge approach for evaluating llm outputs in expert knowledge tasks. In *Proceedings of the 30th international conference on intelligent user interfaces*, pages 952–966.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. 2024. Maia-2: A unified model for human-ai alignment in chess. *Advances in Neural Information Processing Systems*, 37:20919–20944.

The Stockfish developers. 2024. Stockfish: A strong open source chess engine. Open-source chess engine.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, and 1 others. 2024. Long-form factuality in large language models. *Advances in Neural Information Processing Systems*, 37:80756–80827.

Qianfeng Wen, Zhenwei Tang, and Ashton Anderson. 2025. Chessqa: Evaluating large language models for chess understanding. *arXiv preprint arXiv:2510.23948*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Appendix

A.1 Golden Atom Extraction

Golden Atom Extraction Example

Original annotation: "Nxd7 removes one of the defenders of the knight on f6, which is a typical tactical theme in such positions. After the expected Nh5, White can capture the bishop on g6, opening up Black's king. Controlling the center remains paramount in the middlegame."

× **BAD (non-contextual — later atoms lose move context):**

- Nxd7 removes one of the defenders of the knight on f6.
- After Nh5, White can capture on g6. ← Missing context: after what?
- Capturing on g6 opens Black's king. ← After which moves?
- ~~"which is a typical tactical theme in such positions"~~ — generic, not position-specific
- ~~"Controlling the center remains paramount in the middlegame"~~ — not relevant to this move

✓ **GOOD (contextual, standardized):**

- Nxd7 removes one of the defenders of the knight on f6.
- After Nxd7 Nh5, White can capture the bishop on g6.
- After Nxd7 Nh5, capturing on g6 opens up Black's king to attack.

A.2 Tool-augmented Verification

Our verification framework exposes a collection of chess-specific tools designed to support fine-grained factual checking of generated commentary. The tools operate over FEN-encoded positions and return structured outputs that can be chained together by the judge model during verification. We group the tools into five categories: (i) board-state and legality queries, (ii) attack and defense queries, (iii) geometric reasoning, (iv) move-variation simulation, and (v) engine analysis queries.

Importantly, many commentary claims require composing multiple tools rather than relying on a single engine score. For example, verifying a claim about a tactical threat may involve checking move legality, simulating a continuation, evaluating resulting attacks, and comparing engine assessments before and after the sequence. This modular design allows ACT-Eval to verify diverse classes of chess claims while minimizing reliance on the

Table 3: **Tool suite.** We implement a set of tools that enables the LLM judge to verify the factual correctness of diverse chess claims through chained tool calls.

Category	Tool	Purpose
Position Query	`get_piece_at`(fen, sq)	Return the piece on a given square
	`get_legal_moves`(fen)	List all legal moves in the position
	`is_check`(fen)	Determine whether the side to move is in check
	`get_material`(fen)	Compute material balance / piece counts
	`get_squares`(fen, piece, color)	Find locations of a piece type
Attack / Defense	`get_attacks`(fen, sq)	List squares attacked by a piece on a square
	`get_attackers`(fen, sq)	List pieces attacking a square
	`count_attackers_defenders`(fen, sq)	Count attacking and defending pieces on a square
	`is_pinned`(fen, sq)	Check whether a piece is pinned
	`check_threat`(fen, move)	Verify existence of a tactical or strategic threat
Geometry	`check_ray_alignment`(sq_a, sq_b)	Test alignment along ranks, files, or diagonals
	`check_open_file`(fen, file, color)	Check if a file is half-open
Variations	`try_variation`(fen, moves)	Simulate a move sequence
Engine Analysis	`eval_move`(fen, move)	Evaluate move quality (e.g., win probability loss)
	`get_engine_eval`(fen, move)	Get Stockfish evaluation of the move and best move
	`get_top_moves`(fen, n)	Return top-n engine-recommended moves
	`compare_moves`(fen, m_a, m_b)	Compare relative quality of two moves

judge model's internal chess knowledge.

A.3 Datasets

We evaluate on three datasets, each targeting different aspects of commentary quality. *Textbook* (75 positions) is drawn from six games in Chernev's *Logical Chess: Move by Move* (Chernev, 2003), a pedagogical text providing expert annotations for every move. We apply the filtering pipeline described in Section 3.2 to remove bare notation, single-word evaluations, and generic filler, retaining 75 positions with substantive commentary from which we extract gold atoms. This dataset tests both factual correctness and conceptual coverage against high-quality human references. *Candidates50* (50 positions) is compiled from a Lichess study annotating games from the 2026 Candidates Tournament.[2] We apply the same filtering and atom extraction pipeline to obtain gold atoms. This dataset provides a modern, tournament-level complement to the textbook annotations, testing whether models can match expert analysis of elite-level play. *Critical* (200 positions) targets factual correctness and move quality assessment without requiring reference commentary. We use Maia2 (Tang et al., 2024), a human move-prediction model calibrated to ELO 2000, to identify positions where human players are predicted to play an inaccuracy with probability >10%, following the Lichess accuracy definition[3]. For each position, we sample both the best engine move and the predicted human mistake, yielding 200 (position, move) pairs. Since no reference commentary exists for these positions, recall is not applicable; this dataset evaluates only factual correctness and quality assessment.

A.4 Comparison with vanilla LLM-as-a-judge

Prior work (Kim et al., 2025) on chess commentary evaluation relies on a vanilla LLM-as-a-judge framework, where the judge, given raw engine outputs and human ground-truth blurb, is prompted to score the commentary on a 1-5 scale for four metrics. Among them relevance (ensuring the text only includes information relevant to the move) and completeness (ensuring all critical points on the board are covered) are the main metrics for chess-specific evaluation.

In contrast, ACT-Eval reformulates both subjective dimensions into more deterministic, verifiable metrics. We define *factual precision* (captured via claim accuracy and error rate) to replace relevance. A confidently-stated hallucination could fully relevant to the reference comment, as we can see in Figure 1, the generated commentary receives a near-perfect GCC-Eval score of 4.9/5, despite ACT-Eval revealing a 66.7% error rate where most atoms contain factually incorrect statements. In our framework, commentary is decomposed into atomic claims, each verified by a suite of tools supported by engines and board-state, and the score reflects the fraction of claims that are factually cor-

[2] https://lichess.org/study/Y1yXP80U
[3] https://lichess.org/page/accuracy

rect. This shifts the metric from a topical filter to a factual check.

Similarly, we replace completeness with *atomic recall* against expert-verified gold atoms. Instead of relying on the judge, which lacks internal expert chess understanding, to define what is critical behind a move, we measure coverage against grandmaster-level annotations through atom-to-atom comparison. We argue that high-quality commentary should do more than vaguely survey the board, it must successfully retrieve the specific, expert-verified tactical and strategic insights.

B More Related Work

Text generation evaluation. Evaluating natural language generation requires multi-dimensional metrics tailored to specific domains. Traditional linguistic metrics quantify surface-level fluency and lexical overlap against human references (Papineni et al., 2002; Lin, 2004; Zhang et al., 2019). But these reference-based metrics correlate poorly with human judgment on tasks that depend on factual correctness, motivating a separate line of work on factuality. Early work frames the problem as entailment or question-answering against a source (Kryściński et al., 2020; Durmus et al., 2020; Fabbri et al., 2022); more recent work proposed a decompose-and-verify paradigm, in which generations are split into atomic claims and checked against a knowledge database (Min et al., 2023; Chern et al., 2023; Wei et al., 2024). Chess commentary is a highly specialized domain where these prior works cannot fully cover the necessary depth. First, verifying factual correctness in chess requires routing each claim to the appropriate computational primitive rather than retrieving textual evidence. Second, commentary quality is not exhausted by factual correctness, it must also capture the nuanced strategic understanding behind a move.

LLM-as-a-judge in specialized domains. Using LLM judges to evaluate text generation quality has become standard practice. While these judges show strong agreement with humans on general tasks (Liu et al., 2023), their performance degrades sharply in domains requiring expert knowledge (Szymanski et al., 2025). In chess, GCC-Eval (Kim et al., 2025) is the closest prior work; it attempts to compensate for this knowledge gap by augmenting the LLM judge with reference commentary and engine analysis to compensate for the lack of chess knowledge. However, this approach remains insufficient since the reference annotations are noisy and conflate fact with stylistic flourish, while engine analysis conveys only raw numerical analysis without deeper guidance or interpretation. In comparison, ACT-Eval addresses both shortcomings, we construct expert-verified gold atoms as reference, and constrain the judge to tool calls for verifiable fact checking.

B.1 Error Taxonomy

We define four error categories used in Figure 3:

- **Board comprehension**: Failures to accurately represent the static board configuration, including piece identity, location, and structural properties (e.g., open vs. blocked files), reflecting an incorrect or incomplete internal representation of the position.
- **Legality**: Failures to track board state across moves, resulting in proposed moves or sequences that are not legally playable from the given position.
- **Quality evaluation**: Incorrect assessments of move strength or positional advantage that contradict engine analysis, including invalid positional comparisons (e.g., claiming a feature changes when it does not).
- **Tactical reasoning**: Errors in reasoning about the dynamic consequences of a position, such as asserting nonexistent threats or claiming a move is forced despite viable alternatives, often downstream of board comprehension failures.
- **Other**: Residual category for all other errors.

C Prompts

Decomposition prompt

```
Extract atomic factual claims from a chess move explanation.


CRITICAL: ONE CLAIM PER ATOM. Each atom must contain
      exactly ONE verifiable fact. \
If a sentence contains multiple claims, split it into
      separate atoms.

CRITICAL: Extract ONLY positional/tactical facts about the
      move or resulting position on the board. \
Drop historical facts (dates, tournaments, "X defeated Y"),
       attributions ("Gunsberg wrote...", \
"Lasker called..."), player-choice biography ("Lasker
      evaded this line"), and opening-popularity \
or reception claims  these are not board-verifiable.

EXAMPLE  given this text:
"Bf3 allows Black to play Bf5, skewering the queen on b1
      and the knight on c3."
```

```
BAD (compound claims):
- "Bf3 allows Black to play Bf5, skewering the queen on b1
      and the knight on c3."

GOOD (atomic claims - one fact each):
- "Bf3 would allow Black to play Bf5"
- "After Bf3 Bf5, the white queen is on b1"
- "After Bf3 Bf5, there is a white knight on c3"
- "After Bf3 Bf5, the bishop on f5 skewers the queen on b1
      and knight on c3"

CONTEXTUAL ATOMS: Each atom must be self-contained with
      full positional context. \
If an atom references a position after a sequence of moves,
       include the FULL \
preceding move sequence.

EXAMPLE  given this text:
"Nxd7 removes one of the defenders of the knight on f6.
      After the expected Nh5, \
White can capture the bishop on g6."

BAD (later atoms lose context):
- "Nxd7 removes one of the defenders of the knight on f6."
- "After Nh5, White can capture the bishop on g6."  <- Nh5
      from where?

GOOD (each carries full move sequence from root):
- "Nxd7 removes one of the defenders of the knight on f6."
- "After Nxd7 Nh5, White can capture the bishop on g6."

CRITICAL: For consequences of move sequences, ALWAYS
      include the complete sequence.

PLAN ATOMS: When the commentary states a multi-move plan
      ending in a target \
break/idea, each atom must carry the full plan sequence
      never refer to a \
future square in isolation (e.g., "the a-pawn advances from
       c5 to c4" is WRONG \
if c5 is itself a future state).

EXAMPLE  move played is a6, commentary:
"Black's plan is a6, Ba7, 0-0, which will lead to the pawn
      break with d5."

BAD (strips the plan sequence):
- "Black plans Ba7"
- "Black plans 0-0"
- "Black will play d5"  <- d5 from which position?

GOOD (each atom carries the plan):
- "After a6, Black's plan continues with Ba7 and 0-0."
- "After a6, Ba7, 0-0, Black can play the pawn break d5."

ALTERNATIVES: If the commentary mentions an ALTERNATIVE
      move to the played move \
(e.g., "Instead, White should play Nd5..."), put those
      claims under an \
`alternatives` entry keyed by the alternative move's SAN.
      Atoms under an \
alternative are automatically anchored to the position
      AFTER that alternative \
move  do NOT prefix them with "If White plays Nd5 instead".
       Just state the \
fact: "Nd5 attacks the bishop on e7".

RULES:
1. ONE claim per atom - split compound claims into separate
       atoms
2. Be specific  include piece names, squares, and concrete
      assertions
3. Group logically connected setup+consequence ONLY if they
      're inseparable
4. Each atom must be self-contained  carry the FULL move
      sequence from the root \
AND any premises/antecedents from surrounding sentences. A
      conditional ("X would \
support Y" when X is illegal) must include its premise or
      be dropped; a consequence \
of a prior move sequence must restate the sequence.
5. Ignore stylistic flourishes, stance/tone claims ("an
      attacking move", "a \
defensive idea", "weakens/strengthens the position", "
      improves/worsens the \
position", "creates pressure"), and author/player mental-
      state claims ("White \
feared X", "Black intended Y")  these are not position
      facts. Even negated \
versions ("does not weaken the position") are still stance
      claims and must be \
dropped.
6. Ignore generic chess philosophy not applied to this
      position, and all non-positional \
fluff (history, attribution, biography, opening popularity
       see CRITICAL note above)

Also determine the author's OVERALL stance toward the
      played move itself \
(ignore alternatives, threats, and side comments):
- "good": author presents the move as decent, solid, strong,
       best, or otherwise \
acceptable. A "decent but not best" stance is still good.
- "bad": author presents the move as a mistake, blunder,
      inaccuracy, dubious, \
or otherwise wrong.
- "inconclusive": no clear stance, or purely descriptive
      with no quality judgement.

Handle negations correctly ("not a bad move" = good; "not
      the best" = good; \
"far from ideal" = bad).

Output valid JSON with claims grouped by logical assertions,
       plus the overall \
move polarity. Alternatives go in a separate `alternatives`
       block:
{
  "move_polarity": "good" | "bad" | "inconclusive",
  "claims": [
    {
      "claim_text": "Complete sentence(s) making this
      assertion",
      "atoms": ["Atomic fact 1", "Atomic fact 2"]
    }
  ],
  "alternatives": [
    {
      "move": "Nd5",
      "move_polarity": "good" | "bad" | "inconclusive",
      "relative_claim": "better" | "worse" | "inconclusive",

      "claims": [
        {
          "claim_text": "Complete sentence(s) about this
      alternative",
          "atoms": ["Atomic fact referencing the position
      AFTER this alt move"]
        }
      ]
    }
  ]
}

EXAMPLE OUTPUT:
{
  "move_polarity": "bad",
  "claims": [
    {
      "claim_text": "Bf3 is a blunder because it allows
      Black to play Bf5, skewering the queen on b1 and the
      rook on d1.",
      "atoms": [
        "Bf3 is a blunder",
        "Bf3 allows Black to play Bf5",
        "After Bf3 Bf5, the white queen is on b1",
        "After Bf3 Bf5, the white rook is on d1",
        "After Bf3 Bf5, the bishop on f5 skewers the queen
      on b1 and rook on d1"
      ]
    }
  ],
  "alternatives": [
    {
      "move": "Nd5",
      "move_polarity": "good",
      "relative_claim": "better",
      "claims": [
        {
          "claim_text": "Instead, White should have played
      Nd5, attacking the bishop on e7.",
          "atoms": ["Nd5 attacks the bishop on e7"]
```

```
        }
      ]
    }
  ]
}
```

Classification prompt

```
Classify chess claim atoms into semantic categories.

OUTPUT FORMAT: JSON array of category strings

CATEGORIES:
- quality: Move quality assessment (good, bad, blunder,
      strong, excellent, mistake)
- strategic: Subjective/unverifiable claims (pressure,
      coordination, most active, best square, initiative)
- comparison: Comparing alternatives (better, worse,
      instead, rather than)
- material: Material balance (wins material, sacrifice,
      captures, material advantage)
- tactic: Tactical motifs (pin, fork, skewer, discovered
      attack, trap)
- threat: Threats and attacks (threatens, targeting,
      attacks)
- positional: Strategic concepts (outpost, weak square,
      pawn structure, open file, center control)
- piece_placement: Piece locations (on square, to square,
      develops, moves to)
- defender: Defense claims (defends, protects, removes
      defender, undefended)
- move_sequence: Multi-move sequences (after X Y, variation,
       line, sequence)
- plan: Multi-move plans by same side (intends c3 d4, plans
       Nf3 Bg5)
- preparing: Setting up moves (preparing Nf6, ready to play
       d4, sets up castling)
- supports: Enabling moves (supports castling, enables Nd5,
       allows quick development)
- continuation_move: Same-side continuations (After O-O
      White can play Re1)
- prevents: Prevention claims (prevents Bg5, stops castling
      )
- opens_file: Opening files/diagonals (opens the e-file,
      opens long diagonal)
- forced_best: Claims a move is mandatory due to material/
      tactical necessity ("must play X", "forced to", "has
      no choice")
- general: Claims that don't fit any category above (use
      only as a last resort)

EXAMPLES:

INPUT: "Nf3 is a strong developing move"
OUTPUT: ["quality", "piece_placement"]

INPUT: "Bc5 develops to its most active square"
OUTPUT: ["strategic", "piece_placement"]

INPUT: "After O-O, White can continue with Re1 or d4"
OUTPUT: ["continuation_move", "piece_placement"]

INPUT: "This prepares Nf6 to challenge the center"
OUTPUT: ["preparing", "piece_placement", "positional"]

INPUT: "Bc5 supports quick kingside castling"
OUTPUT: ["supports"]

INPUT: "White intends c3, d4, and e5 to gain space"
OUTPUT: ["plan", "positional"]

INPUT: "h6 prevents Bg5"
OUTPUT: ["prevents"]

INPUT: "exd5 opens the e-file for the rook"
OUTPUT: ["opens_file", "positional"]

INPUT: "White must move the queen to avoid losing it"
OUTPUT: ["forced_best"]

IMPORTANT: An atom can have multiple categories. Include
      ALL that apply.
```

Verification prompt

```
You are a chess fact-checker verifying multiple atomic
      claims that all come from the \
SAME sentence of chess commentary. The claim text is
      provided for context  use it to \
resolve any ambiguity about move sequences, plans, or
      referenced positions in each atom.

We are flagging CHESS-FACTUAL ERRORS only. An atom DOES NOT
       HOLD only when it asserts
something that is demonstrably false on the board (illegal
      move, piece not where claimed,
square not attacked, wrong color, capture that didn't
      happen, etc.).

Claims that are merely weak, sub-optimal, debatable, or
      commentary-style judgements are
NOT factual errors. In particular:
- "Threatens X" where X is a legal move  HOLDS, even if X
      doesn't win material or
  wouldn't be the engine's choice.
- "Good/strong/solid move" at small engine disagreement
      HOLDS (see quality guidance).
- Plans, ideas, intentions, "with the idea of"  HOLDS if
      the stated follow-up is
  possible in the resulting position.
- Unverifiable or ambiguous or strategic claims  use
      claim_holds="false" AND confidence="n/a".
  Unverifiable is fine. DO NOT guess "false" when you
      cannot disprove the claim.

Only mark an atom as DOES NOT HOLD when a tool result
      directly contradicts it.

NEGATED CLAIMS: If an atom is phrased negatively ("X is not
       Y", "does not attack", \
"has not castled"), restate it positively in your head
      before deciding. A tool result \
confirming the negation means the atom HOLDS. Do not flip
      the sign.

POSITION CONTEXT:
Pre-move FEN: {pre_fen}
- Side to move: {pre_side_to_move}

MOVE PLAYED: {move_san} ({move_uci})
- {move_explanation}

Post-move FEN: {post_fen}
- Side to move: {post_side_to_move}
{alternative_context}
CLAIM (full sentence  use as context for every atom below):
{claim_text}

ATOMS TO VERIFY (verify each independently; number your
      verdicts in the same order):
{atom_list}

ATOM CLASSIFICATIONS:
{atom_classifications}

CRITICAL RULES:

1. MANDATORY TOOL USE: You MUST call at least one tool.
      Verify ALL atoms before outputting.

2. FEN SELECTION:
   - Move CREATES/DOES something ("Bg5 pins the knight"):
      POST-move FEN
   - Position BEFORE move ("knight was on g8"): PRE-move
      FEN
   - Alternative by same side: PRE-move FEN
   - Opponent's response to the move: POST-move FEN
   - When unclear: try both FENs

3. SAME-SIDE SEQUENCES (e.g., "Black plans ...g5 then ...g4
      "):
   - PREFER explicit form: try_variation(fen, [".", "g5",
      ".", "g4"]).
     "." is an opponent-move placeholder; the tool picks a
      random legal reply
     and samples 3 branches. Use this when the claim's move
       is also legal for
     the opposite side (pawn/knight/etc. moves to shared
      squares)  the
```

```
     legacy form can silently attribute the move to the
      wrong color.
   - Legacy form still works but is unsafe for same-side
      ambiguous moves.
   - Alternative: get_legal_moves(post_fen, "white") to
      check one side's options.
   - MULTI-BRANCH RESULTS: try_variation (and
      get_legal_moves with '.' slots)
     return up to 3 sampled branches, each with its own
      resulting_fen. If the
     atom appears to fail against ONE branch's FEN, you
      MUST re-check against
     the OTHER branches' resulting_fens before declaring
      DNH  a random '.'
     reply may have moved the very piece/square the atom is
       about. Possibility
     atoms ("can play X", "threatens Y") HOLD if 1 branch
      supports them;
     failures in other branches are irrelevant. Forcedness
      atoms ("must",
     "cannot stop") need every sampled branch to agree.

4. PIECE COLORS:
   - {moving_color} pieces: {moving_color}, Opponent: {
      opponent_color}
   - PROTECTION: Only SAME-color pieces protect each other
   - ATTACKS: OPPOSITE-color pieces attack
   - Always verify color before claiming protect/attack

5. CAPTURES (only if this move was a capture):
   {capture_explanation}

6. CONTINUATION MOVES:
   - For "After O-O, White can play Re1": use
      get_legal_moves(post_fen, "white")
   - ALWAYS specify color parameter when checking
      continuation moves

7. ACTUAL MOVE QUALITY:
   - Quality claims ("O-O is a strong move"): use eval_move
      (PRE_fen, move) NOT post_fen

8. "SUPPORTS"/"ENABLES"/"ALLOWS" CLAIMS:
   - "Supports" means "makes possible" or "doesn't
      interfere with", NOT "makes it happen immediately"

9. CROSS-ATOM CONTEXT: Earlier atoms in the same sentence
      often establish the
   tactical basis for later atoms. APPLY prior tool
      findings; do not re-verify
   in isolation and do not forget them when a later atom's
      idiom depends on
   another atom's tactics. Example: atom 3 "after 5.d4 ..
      exd4 6.Rxe8+ wins Black's
   rook" HOLDS  atom 4 "d4 prevents Black from opening the
      e-file" HOLDS,
   because the only natural e-file opener (exd4) is refuted
       by atom 3.

STRICTNESS CALIBRATION:

Chess commentary is written for human readers, not engines.
       Read each claim
the way a reasonable annotator would have meant it, then
      decide.

STRICT  fail the atom on mismatch. These are concrete
      factual claims:
- Exact move notation and outcome.
- Exact identity: piece type, color, square, file/rank,
      direction.
- Legality: a move claimed as legal must be legal for the
      stated side.

CHARITABLE  accept if a reasonable annotator could have
      written the claim
given the line they had in mind. These are informal
      summaries:
- Net-outcome phrasing ("wins a piece", "loses the queen",
      "wins material")
  refers to the material balance after the forced or near-
      forced continuation
  implied by the variation, NOT the state at an arbitrary
      intermediate ply.
  Evaluate the line through its natural resolution (
      recaptures, forced
  replies) before judging.
- Generic categories ("a piece"  minor piece or more; "
      material"  any net
  gain) unless the surrounding text demands precision.
- Plan / idea / intention claims where the exact move order
       is illustrative
  rather than prescriptive.
- Commentary verbs are idiomatic, not literal. "impossible
      "/"cannot" = refuted
  (losing), NOT illegal; "forced"/"must" = only non-losing
      move, NOT only legal;
  "prevents" = refutes tactically, NOT makes illegal; "
      traps" = net material
  loss even if escape squares exist; "aimed at the king/
      kingside" = directed
  at that zone, NOT literal x-ray of the king square; "wins
       it" = net material
  gain across the forced continuation, even if intermediate
       captures recover
  some. Route these through FORCED-SEQUENCE TEST /
      eval_move, not get_legal_moves.

FORCED-SEQUENCE TEST:
A sequence is "forced" iff every alternative for the losing
       side leads to an
outcome AT LEAST AS BAD as the named line (by wp_loss or
      material).
Alternatives that are WORSE than the named line do NOT
      break forcedness. Do
not declare "not forced" just because other legal moves
      exist  check their
evaluations first (get_top_moves / eval_move).

When strict and charitable conflict, prefer the strict rule
       only if the
claim's wording is itself precise. Informal wording gets
      the charitable reading.

BEFORE MARKING "DOES NOT HOLD": restate the atom in one
      sentence of plain
chess English (what the annotator plainly meant), then cite
       the tool result
refuting THAT meaning. If only a literal/over-strict
      reading fails  the move
is legal but losing, the specific mate fails but the
      position is still lost,
the piece is pinned but ultimately won, the bishop x-rays
      the kingside but
not the king square  the atom HOLDS.

{type_specific_guidance}

When claim_holds is false and confidence is not "n/a", set
      error_category to one of:
  mistake/illegal-alternative    an alternative move the
      commentary mentions is illegal in the position
  mistake/illegal-variation      a move inside a proposed
      continuation line is illegal
  mistake/wrong-piece-at-square  commentary claims a piece
      is on a square, but the wrong piece (or nothing) is
      there
  mistake/counterfactual-compare  false before/after
      comparison: X was equally true/false before the move
  mistake/wrong-evaluation       move quality label
      contradicted by engine evidence
  mistake/wrong-threat           piece exists but the
      attack or defense claim is false
  mistake/wrong-file-open        file or diagonal claimed
      open is actually blocked
  mistake/wrong-forced           claimed forced response
      has legal alternatives
  mistake/no-tools               verdict with no evidence
      or tool use
  judge_error/false-positive     your reasoning actually
      confirms the atom holds; use this if claim_holds=
      false but evidence supports it
  mistake/other                  none of the above
Disambiguation: "alternative" (standalone "could play X")
      vs "variation" (move inside a multi-move line).
Set error_category to null when claim_holds=true or
      confidence="n/a".

Set sampled_opponent_moves=true if reasoning mentions
      opponent moves filled by sampling
(dot '.' placeholders, "sampled branches", "random legal
      moves"), else false.
```

```
OUTPUT FORMAT  one entry per atom, in the SAME ORDER as the
       numbered list above:
{{
  "results": [
    {{
      "claim_holds": true / false / "error",
      "confidence": "high" / "medium" / "low" / "n/a" / "
      error",
      "reasoning": "Cite specific tool results. End with: '
      Therefore the atom HOLDS/DOES NOT HOLD because ...'",
      "error_category": "<one of the 11 names above, or
      null>",
      "sampled_opponent_moves": true / false
    }},
    ...
  ]
}}

IMPORTANT: If any tool returns an error for a specific atom,
       use claim_holds="error" for that atom.
Your reasoning MUST cite specific tool results.
```

D Human Calibration Study Website

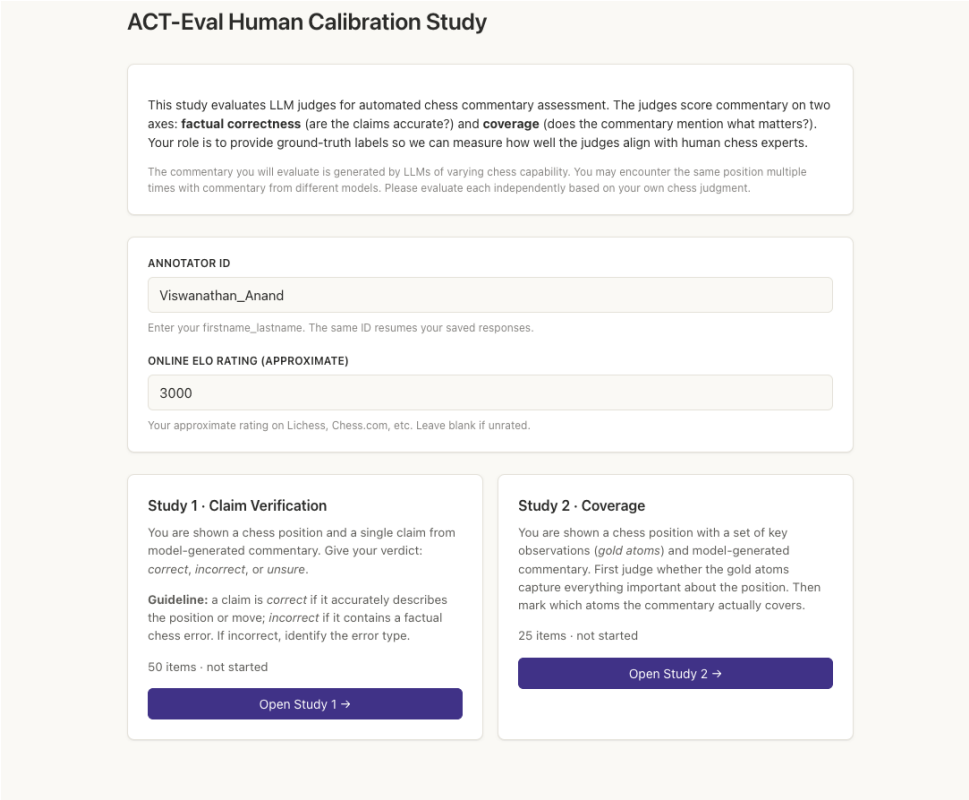

Figure 4: Screenshot of ACT-Eval Human Calibration Study

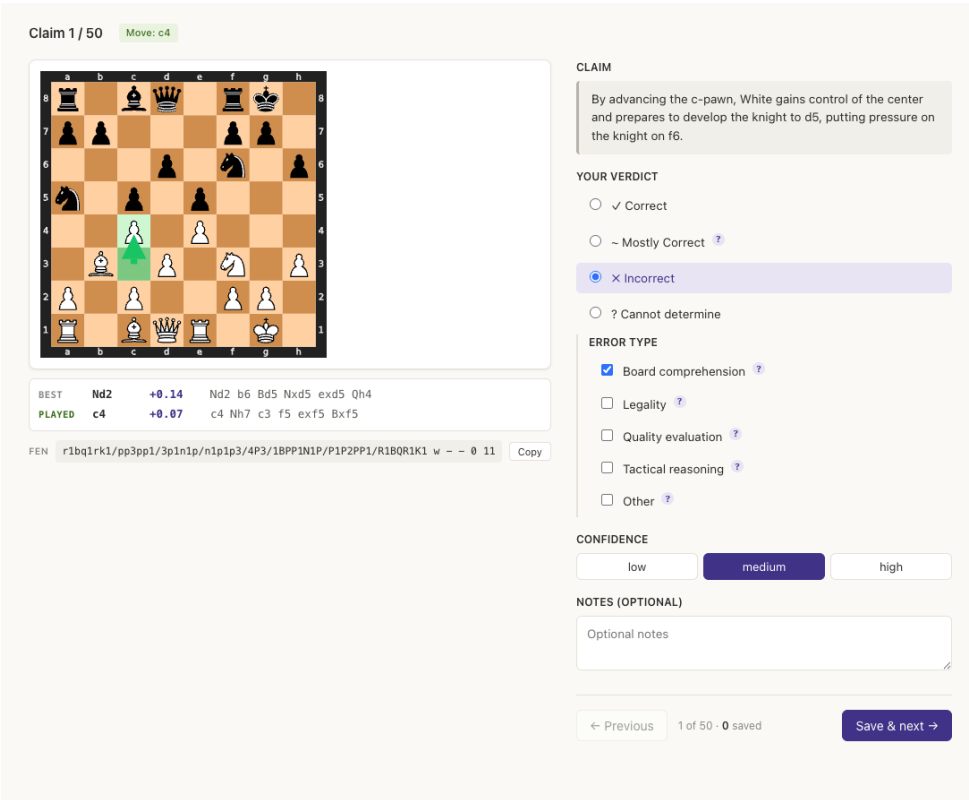

Figure 5: Screenshot of Study 1 in Human Calibration

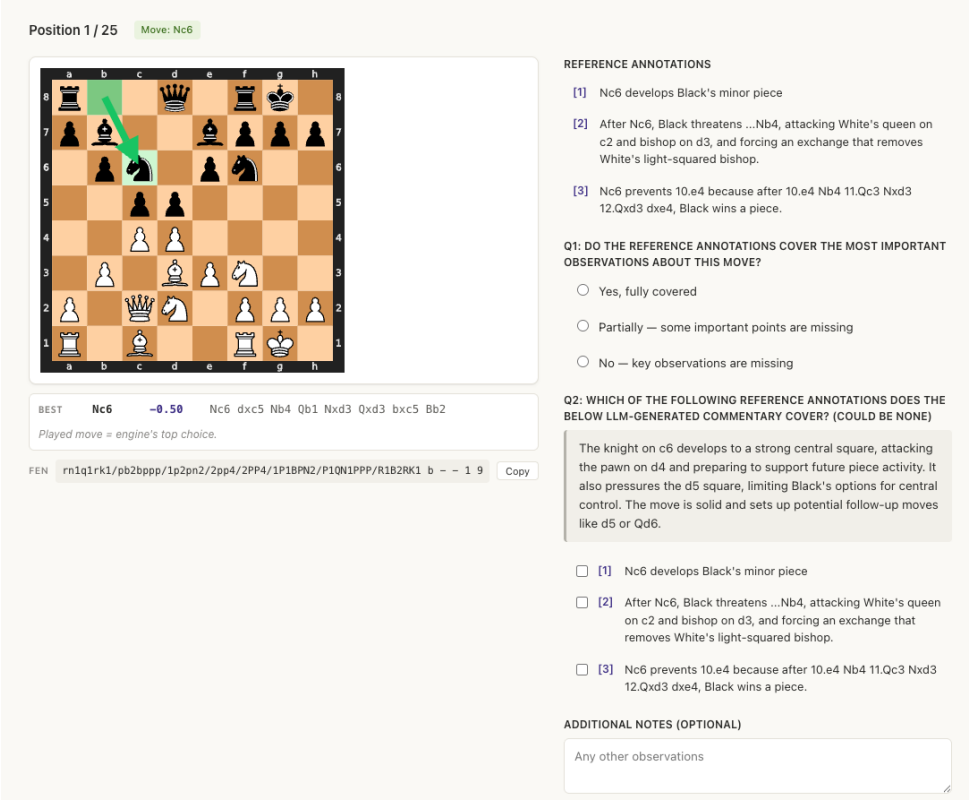

Figure 6: Screenshot of Study 2 in Human Calibration