CLAIMDB: A Fact Verification Benchmark over Large Structured Data

Michael Theologitis[1], Preetam Prabhu Srikar Dammu[1], Chirag Shah[1], Dan Suciu[1]
[1]University of Washington,
Seattle, WA, USA
mthe@cs.washington.edu

Abstract

Real-world fact-checking often involves verifying claims grounded in structured data at scale. Despite substantial progress in fact-verification benchmarks, this setting remains largely underexplored. In this work, we introduce CLAIMDB, a fact-verification benchmark where the evidence for claims is derived from compositions of millions of records and multiple tables. CLAIMDB consists of 80 unique real-life databases covering a wide range of domains, from governance and healthcare to media, education and the natural sciences. At this scale, verification approaches that rely on "reading" the evidence break down, forcing a timely shift toward reasoning in *executable* programs. We conduct extensive experiments with 30 state-of-the-art proprietary and open-source (below 70B) LLMs and find that more than half score below 55% accuracy. Our analysis also reveals that both closed- and open-source models struggle with *abstention*—the ability to admit that there is no evidence to decide—raising doubts about their reliability in high-stakes data analysis tasks. We release the benchmark, code, and the LLM leaderboard at https://claimdb.github.io.

1 Introduction

Claims based on large-scale structured data are everywhere. They effectively drive and justify the most important decisions of our times. For example, Joe Biden stated on August 29, 2023 in a speech at the White House that the U.S. inflation *"is now down close to 3, the lowest among the world's leading economies."* (2023) at the time of the most aggressive cycle of interest-rate hikes in decades (PBS, 2022). Similarly, two years later, on August 11, 2025, Donald Trump stated that *"Washington, D.C., has 41 homicides per 100,000 people, No. 1 that we can find anywhere in the world."* (The White House, 2025) to justify the historic decision and deployment of the national guard at the U.S. capital (announced later in the same speech).

In both cases, these claims are summaries of large, official, publicly available structured datasets. Biden's inflation claim can be verified against consumer price indices (CPI) published by the Bureau of Labor Statistics in the form of approximately ten separate Excel tables (U.S. BLS). On the other hand, Trump's crime claim can be fact-checked against crime records released by the Metropolitan Police Department of D.C. in the form of a single CSV file containing crime incidents (MPD, 2025).

Despite the central role of structured data in real-world decision-making, fact-verification research has largely focused elsewhere. Prior work has made important progress on evidence *grounded* in modalities ranging from text and documents to Wikipedia tables, info-boxes, and small relational tables (e.g., Chen et al., 2020; Schlichtkrull et al., 2023). These settings have enabled significant advances in fact-checking (e.g., Bazaga et al. 2024), but they share a common simplifying assumption: evidence is *small*—in fact, it is small enough to fit within an LLM's context window. As Biden and Trump's examples showcase, this assumption does not hold for many high-stakes, real-world claims.

In this work, we propose CLAIMDB, a fact-verification benchmark in which claims are grounded in evidence deliberately composed from multiple tables and millions of rows. Successful verification in CLAIMDB implicitly requires some form of *executable* reasoning (Chen et al., 2023) as the evidence is too large to naively "read", consume, and combine. Each claim is paired with a unique database—out of 80 in total—that corresponds to an evidence context of roughly 110M tokens. More specifically, each claim involves 11 tables and 4.5M records on average, reflecting the scale and complexity of real-world fact-checking.

The remainder of the paper is organized as follows. Section 2 reviews existing fact-checking benchmarks. Sections 3–5 describe the full bench-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 34428–34451

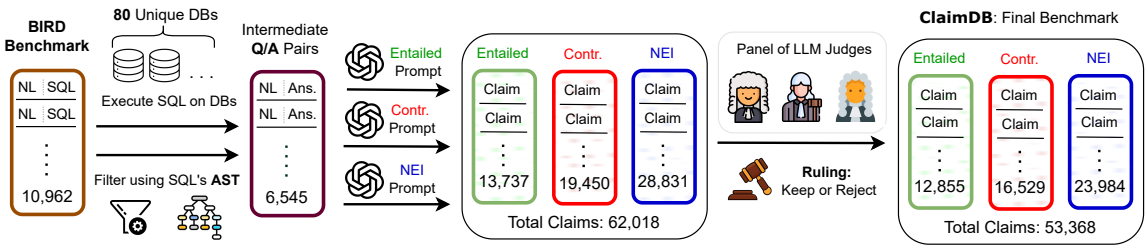


Figure 1: Overview of the CLAIMDB construction pipeline. We start from the NL-to-SQL BIRD benchmark (Section 3.2), execute each query on its respective real-world database, and filter out *low-information* pairs using the query AST (Sections 3.3, 3.4). For each remaining Q/A pair, we prompt `gpt-5` to generate claims grounded in the gold answer—with some additional context for NEI claims (Section 3.5). We then use a panel of LLM judges from Mistral AI, Microsoft, and xAI to retain only high-quality claims (Section 4.1). Finally, we apply embedding-based post-processing to two NEI categories for high-quality sampling (Section 4.2); this step is omitted for space reasons.

mark construction pipeline, including claim generation, quality control with LLM judges, and dataset splits. We then evaluate 30 LLMs under a specific neuro-symbolic architecture in Section 6, before concluding the paper in Section 7.

2 Related Work

Fact verification started with free-form text. The seminal work of Thorne et al. (2018) on FEVER pairs claims with Wikipedia sentences to determine whether they are supported or refuted. LIAR (Wang, 2017) is a complementary benchmark of political claims with fine-grained veracity labels. Since then, fact-verification has evolved beyond text to include structured and semi-structured evidence. FEVEROUS (Aly et al., 2021) extends FEVER with tables, while Bussotti et al. (2023, 2024) automate training-data generation for such settings. Other works explore verification over time-series data (Strong and Vlachos, 2025), Wikipedia info-boxes (Gupta et al., 2020), knowledge graphs (Dammu et al., 2024), financial reports (Zhao et al., 2024), temporal data (Barik et al., 2024, 2025), scientific paper abstracts (Wadden et al., 2020), U.S. court rulings (Putta et al., 2026), and combined text and visual representations of tabular data (e.g., charts) (Zhou et al., 2025).

As the evidence modality naturally evolves to *real-life*, large structured sources like tables, the reasoning starts to become more symbolic. The popular TabFact benchmark (Chen et al., 2020) requires operations like aggregation (e.g., `AVG`, `SUM`, `MAX`) and comparison. Subsequent benchmarks, including SEM-TAB-FACTS (Wang et al., 2021), and SCITAB (Lu et al., 2023) further explored this setting by grounding real-life claims in tables extracted from complex scientific articles.

However, existing fact-verification benchmarks rely on tables that are small enough to fit within an LLM's context window. Evidence is drawn almost exclusively from Wikipedia or scientific articles, where tables are inherently limited in size. Recent work by Devasier et al. (2025, 2026) begins to explore larger structured data, but these approaches remain limited—either to small-scale *pilot* claim sets or to single-table evidence. In CLAIMDB, we go further by designing a benchmark where claims are grounded in databases with millions of records, requiring multi-hop and compositional reasoning across tables. At this scale and complexity, verification can no longer rely on naively "reading" and then reasoning over in-context evidence.

3 CLAIMDB Benchmark

In CLAIMDB, the task is to determine whether a given claim is ① *entailed*, ② *contradicted*, or ③ has *not-enough-info* (NEI) with respect to the evidence in a database. Each claim is paired with exactly one large, multi-table database, which serves as the sole source of evidence for verification.

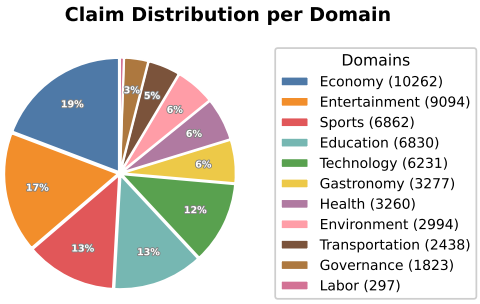
The workflow for creating the CLAIMDB benchmark is shown in Figure 1. In this section, we focus on the first stage of this process—the creation of claims—followed by quality-control in Section 4.

3.1 Overview

The final CLAIMDB benchmark contains 53,368 claims paired with databases from the BIRD benchmark (Section 3.2): 12,855 *entailed*, 16,529 *contradicted*, and 23,984 *not-enough-info*. Table 1 reports a detailed breakdown of label distributions and dataset statistics.

Property	Value
Total Claims	53,368
Label distribution	
# Entailed (E)	12,855
# Contradicted (C)	16,529
# Not Enough Info (NEI)	23,984
NEI subcategories	
# Out-of-Schema	12,644
# Counterfactual	5,786
# Subjective	5,554
Claim Context (Evidence)	
Databases per Claim	1
Avg. Tables per Claim	**11.3**
Avg. Records per Claim	**4.6M**
Avg. Tokens per Claim†	**110M**
Total Databases (DBs)	80

Table 1: Summary statistics of CLAIMDB. †Token counts per claim are computed using the tokenizer of the `gpt-5` series as reference (see Appendix A for details).



Domain	Subdomains
Economy	Finance, World Economies, Retail, Banking
Entertainment	Movies, Music, TV Shows, Games, Cartoons
Sports	Basketball, Olympics, Hockey, F1, Soccer
Technology	Software, IT, Blockchain, Vision
Education	University, Academia, Schools, Language, Books
Gastronomy	Food, Restaurant
Health	Healthcare, Medical, Biology, Chemistry
Environment	Weather, Geography
Transportation	Transit Systems, Airport
Governance	Crime, Law
Labor	Human Resources

Figure 2: Claim distribution and taxonomy. We group the 80 databases into 11 high-level domains (introduced in this work), each comprising multiple subdomains. The subdomains are inherited from Li et al. (2023) with a few minor modifications.

What makes CLAIMDB challenging is the scale of the underlying data. Each claim is paired with a database containing, on average, 11.3 tables and 4.6M records. When converted to text, this amounts to roughly 110M tokens of evidence per claim (Appendix A). This exceeds the context window of modern LLMs by two to three orders of magnitude—for reference, Google's flagship *long-context* models support up to 1M–2M tokens (Doshi, 2025). Thus, common approaches that rely on feeding tables directly into an LLM (e.g., Wang et al., 2024c) are no longer viable.

Instead, successful systems must rely on *executable* programs to carry out the heavy compositional reasoning required by CLAIMDB over large volumes of data—for example, averaging or sorting millions of records (Section 3.3). Promising directions include Program of Thoughts (PoT) (Chen et al., 2023), tool-calling agents (Wang et al., 2024a; Theologitis and Suciu, 2025), and Recursive Language Models (Zhang et al., 2026).

The benchmark uses 80 distinct real-life databases from BIRD (Li et al., 2023), covering a wide range of domains and subdomains, from governance and healthcare to media, education, transportation, and the natural sciences. Figure 2 summarizes the domain taxonomy present in CLAIMDB (full details are in Appendix Table 4).

3.2 BIRD Benchmark

Our starting point is the BIRD (Li et al., 2023) NL2SQL benchmark. Mapping NL to SQL has a long history, dating back to early work on NL interfaces (Warren and Pereira, 1982). More recent systems include WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), and KaggleDBQA (Lee et al., 2021). BIRD is the first benchmark that consists of realistic databases, with multiple, large tables (millions of rows), including "dirty" data such as null values and inconsistent formatting.

BIRD provides us with 11k NL-to-SQL examples (the combined public `train` and `test` splits). Each example is paired with the underlying database and accompanied by *external information* specific to each example. For example, in the `toxicology` database, a label marked with "+" indicates that a molecule is carcinogenic, while "−" indicates the opposite. Throughout all subsequent transformations, we preserve any *external info* intact. Importantly for us, the NL-to-SQL pairs have been carefully curated and vetted by trained, independent annotators, and the SQL queries are produced via a double-blind annotation process with experts in the loop (Li et al., 2023).

3.3 Pre-Filtering

We start from the 11k NL-to-SQL pairs provided by BIRD. As a first filtering step, we retain only pairs where the SQL query combines substantial information in order to answer the NL question. For

example, a query that involves a superlative (e.g., `MAX`) or averaging (`AVG`) summarizes large portions of the database, collapsing millions of values into one. This is highly desirable for us, because verifying claims derived from such pairs implicitly requires compositional reasoning over the database. On the other hand, a lookup query that returns a single record (e.g., *"What is the name of the mayor of Seattle?"*) is of no interest: a verifier could answer by simply inspecting and seeing the *right* data.

Compositional Queries. To identify such cases, we convert every SQL query into an abstract syntax tree (AST) and use it to identify the ones that contain computations over substantial parts of the data. More specifically, a query is retained if and only if its AST contains *at least one* of the following:

- Orderings or superlatives, which introduce comparisons over all data (e.g., `ORDER BY`).
- Aggregate Functions, such as `AVG` and `SUM`, which operate over large sets of records.
- Window functions, which involve complex information flow across partitions of the data.
- Multi-table joins, where three or more tables are combined for the final result.

The common theme is that answering the question (or, later, verifying the claim) requires combining information from large portions of the database, which will far exceed the context size of any LLM. Figures 13–16 in Appendix D provide concrete examples of queries, their ASTs, and the corresponding filtering decisions following the rules above.

Finally, we discard queries whose answers return more than ten records. These answers are later fed to an LLM during claim generation (Section 3.5), and large result sets would make it harder for the model to track the structure (columns and values). This ten-record cutoff is conservative for modern models like `gpt-5`, but helps ensure that errors do not arise from misinterpreting the structured data.

After these two filtering steps we end up with roughly 6.5k NL/SQL pairs, where each SQL query "touches" a large subset of its associated database.

3.4 Question-Answer Pairs over Databases

Next, we execute each SQL query on its corresponding database to obtain the exact answer. This yields Question/Answer pairs grounded in real-world databases. From this point on, the SQL itself is no longer needed and can be discarded.

The answer to each SQL query is a table, i.e. a set of rows, which is not a friendly format for LLMs. Following Singha et al. (2023), we convert each answer from a table to JSON, a format that is friendlier for LLMs. After this step we are left with 6.5k Question/JSON-Answer pairs.

3.5 Claim Generation

Each pair serves as ground truth. Given one such pair, we prompt `gpt-5` to generate claims that are either *entailed* by the answer or *contradicted* by it (we discuss NEI claims later in this section). Figure 3 shows a concrete example of a single Q/A pair together with the claims generated from it.

Domain: California Schools
Question: Which cities have the top 5 lowest enrollment number for students in grades 1 through 12?
Answer: Coulterville; Pinecrest; Shaver Lake; Emigrant Gap; Hyampom.

Generated Claims:

- **Entailed:** In California, the five cities with the lowest K-12 student enrollment are Coulterville, Pinecrest, Shaver Lake, Emigrant Gap, and Hyampom.
- **Contradicted:** The three California cities with the lowest K-12 enrollments are Lee Vining, Trona and Keeler.
- **NEI (Out-of-Schema):** Most families in Hyampom choose homeschooling rather than enrolling their children in public schools.
- **NEI (Counterfactual):** If Pinecrest had opened a new K-12 campus in 2020, its grades 1-12 enrollment would have increased enough that the city would not rank among the five lowest in California.
- **NEI (Subjective):** Among Coulterville, Pinecrest, Emigrant Gap, and Hyampom, Emigrant Gap provides the most *nurturing* learning environment for K–12 students.

Figure 3: Example of claims generated from a single Q/A pair in CLAIMDB (California Schools database). In the generation of NEI claims we also give the database *schema* information along the golden context.

For this step we explored a range of prompting strategies (Kotha et al., 2025), including different task instructions, zero-shot versus few-shot, and varying levels of CoT, while keeping track of failure modes—which we deal with in the next section.

Our final setup uses a single prompt for each label (3 in total). All final prompts are available in Figures 17, 18 and 19 of the Appendix.

For both *entailed* and *contradicted* claims, we found that 1-shot prompting with medium reasoning works best. Given a Q/A pair, the model is instructed to generate between one and three claims.

NEI Claims. Lastly, we generate claims that are *definitely unanswerable* from the database at hand. We draw inspiration from the taxonomy presented by Kirichenko et al. (2025) and the work of

Amayuelas et al. (2024). Given our specific setting, we can safely support three claim categories.

First, we generate claims that fall outside the concepts of the database. To do this, we inspect the database schema metadata offline—including table and column names, and relationships—and provide the full schema information to `gpt-5`. We then explicitly instruct the model (Figure 19 in the Appendix) to produce claims that are based on concepts not represented anywhere in the schema. The model has no trouble doing this effectively: for example, in Figure 3, the generated *out-of-schema* claim refers to *homeschooling*—a concept that is not present in the California Schools database.

Second, we generate *subjective* claims which express opinions or value judgments that cannot be objectively verified. Finally, we generate *counterfactual* claims. These describe imagined or "what if" scenarios that are unanswerable with certainty.

For NEI claim generation, we found that zero-shot prompting with medium reasoning works best. There is a large spectrum of possible and appropriate NEI claims, as they are largely unconstrained by the concepts of the data. As a result, any form of in-context learning biases the model toward specific patterns, limiting generation diversity overall. In contrast, `gpt-5` naturally generates a diverse set of NEI claims in the zero-shot setting.

All in all, we end up with roughly 14k, 19k, and 28k entailed, contradicted, and NEI claims, respectively (Table 1). The discrepancy in counts arises because contradicted claims are not constrained by the answer itself (they can oppose it arbitrarily), while NEI claims are not constrained by either the answer or even the concepts present in the Q/A pair or the database (we address this in Section 4.2).

4 Quality Control

Automatic claim generation is never perfect. Issues can still slip-in such as claims that reference prior context—which is opaque to a verifier—or leak helpful information. While outright label errors are rare (the claim generation task is trivial), we take *conservative* steps to filter problematic claims. This is the second half of the workflow in Figure 1.

4.1 Panel of LLM Judges

Manually annotating 64k examples is infeasible, so we rely on the now widely adopted *LLM-as-a-Judge* paradigm, where an LLM evaluator is used as a proxy for human annotators.

Judge Panel. It is well-known that LLMs tend to prefer answers generated by themselves (Zheng et al., 2023; Ye et al., 2025). To avoid this self-enhancement bias, we therefore exclude OpenAI models entirely. We further follow Verga et al. (2024), who show that replacing a single large evaluator with a *panel of judges* of smaller LLMs leads to more stable results. So, we construct a judge panel from three different model families coming from Microsoft, xAI, and Mistral AI:

(a) `Phi-4` (Abdin et al., 2024)
(b) `grok-3-mini` (xAI, 2025)
(c) `mistral-small` (Mistral AI, 2025c)

Each judge independently evaluates every claim, and we later combine their judgements for the final *ruling*—whether a claim is eliminated or not. Importantly, we also ask the judges to *justify* their reasoning and verdicts in a few sentences in NL.

Rubrics. Before putting the judge panel to work, we must first clearly define the evaluation criteria. Each judge is given the full gold context—namely the NL question, answer, and domain—along with the generated claim and its assigned label. Judges then answer the following binary ("yes" or "no") questions: ① *Is the label correct?*, and ② *Is the claim self-contained?* The second question refers to claims where there are opaque references to previous context (e.g., "the question above", etc.).

NEI claims require additional care. During generation, `gpt-5` had access to the schema as part of the gold context. As a result, schema artifacts occasionally appear directly in the generated claims, which unintentionally helps systems evaluated on the benchmark. For these claims, we therefore ask two additional questions: ③ *Is there schema leakage?*, and ④ *Is the assigned NEI category correct?*

Ruling. If *any* judge flags a claim as problematic under *any* rubric (e.g., incorrect label, lack of self-containment, etc.), the claim is discarded. In other words, a single negative judgment is grounds for elimination. This is intentionally *conservative* as it gives us confidence that the final benchmark is clean—even if that means over-eliminating claims.

Prompt. The evaluator prompt plays a central role in how reliable the judging process is (Ye et al., 2025). We already simplify the task as much as possible (e.g., binary classification rubrics), and we make one more simplifying decision: we are willing to "unfairly" discard borderline claims.

While most LLM-as-a-Judge setups aim to max-

imize agreement with human annotators (Thakur et al., 2024), our goal is slightly different. We want to maximize the *recall* on claims that humans would eliminate. To that end, the prompt includes instructions such as *"If you are unsure, answer no."*

To calibrate the prompt, we manually annotated 300 examples (150 NEI and 150 contradicted or entailed) using our rubrics. We shuffle and split these annotated examples in two halves: one for calibrating the prompt, and the other for testing it.

Prompt Engineering. After calibration, we end up with the prompts shown in Figures 20 and 21 of the Appendix. The former is used for entailed and contradicted claims, while the latter for NEI claims and includes the two additional evaluation rubrics.

Prompt Results. We evaluate the judge panel on the human-annotated test set (75 NEI and 75 contradicted or entailed), which contains 4 claims identified as problematic by PhD annotators. The panel successfully discards all four, achieving 100% recall, and behaves conservatively. We also did not observe any mislabeled claims in terms of veracity in this set. This aligns with our experience during the construction of the benchmark—generating claims from correct Q/A pairs is an easy task, and `gpt-5` is more than capable of doing so reliably.

Finally, we run the full benchmark through the judge panel with all three models evaluated at temperature zero. Overall, 5.6% of claims are flagged for incorrect labels and 1.7% for not being self-contained. Among NEI claims, 11% show schema leakage and 5% have an incorrect category assignment. Taken together, this filtering step removes approximately 14% of the benchmark (Figure 1).

4.2 Grounding NEI Claims

The main reason we have a large number of NEI claims is that they are inherently unconstrained by the database concepts. They can be about almost anything, especially in the case of *out-of-schema* claims. However, this freedom often makes them very obvious to classify. The same issue arises with *counterfactual* claims—given a bit of imagination (`gpt-5` does not lack this quality), it is easy to go overboard. These overboard claims are not bad per se; we simply want a way to avoid sampling them when we construct our main benchmarking test set. In this section, we create a way to filter out claims based on how *far* they are from the underlying database concepts. We later use this signal to construct a more difficult test set (Section 5).

Essentially, we are looking for a way to rank NL claims by how "close" they are to the concepts of the underlying data—we can approximate this using the golden context. We capture this "closeness" using semantic textual similarity (STS). More specifically, we embed[1] the Q/A pair and the generated claim, and compute the similarity between them. The resulting score gives us a simple and effective way to *rank* claims by how closely they stay grounded in plausible concepts of the database.

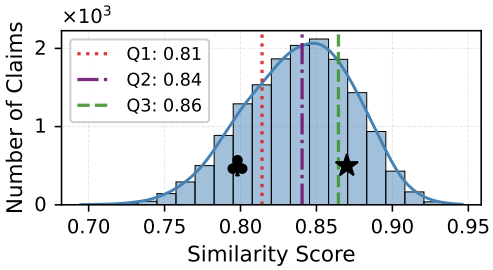


Figure 4: Distribution of similarity scores between generated claims and their gold context for *out-of-schema* claims. Higher scores indicate that claims stay "closer" to the underlying data concepts. The two highlighted claim examples (♣, ★) are discussed in Section 4.2.

Figure 4 shows the distribution of similarity scores for *out-of-schema* claims. To make this concrete, we look at two claims generated from the same Q/A in the Chicago Crime database, which asks about crimes in Chicago's Central district. Although both claims are *out-of-schema*, they fall into very different parts of the similarity distribution:

> (♣) *The commander of Chicago's Central police district holds a law degree from an Illinois university.*
> (★) *Crimes recorded in Chicago's Central district disproportionately involve tourists compared to other districts.*

The second claim is much closer to the database concepts: validating it would require inspecting the data to see whether victim information (e.g., registered residents) is available, which is at least plausible. The first claim, in contrast, drifts far from what the database could reasonably support. This difference is captured by the similarity score: the second claim (★) falls in the top quartile (0.869), while the first (♣) lies in the lowest quartile (0.798).

[1]We use `gemini-embedding-001` (Lee et al., 2025) for the embeddings; at this time, it ranks 4th on MTEB (2023).

5 Benchmark Splits

We divide CLAIMDB into three disjoint splits: ① the training set, ② a *public* test set, and ③ a *private* test set (the labels are hidden). Both test sets contain approximately 1,000 examples each. They are sampled[2] uniformly at random per label and fully balanced, with roughly 333 examples per label. For NEI claims, the three subcategories are also balanced with roughly 111 examples each; for both *out-of-schema* and *counterfactual*, sampling is further restricted to the top quartile of the similarity score distribution (Figures 4 and 10). In contrast, the training split contains the full spectrum of NEI claims. We release similarity scores for all NEI claims, enabling the construction of any custom downstream training split—for example, one that mirrors the test set distribution.

6 Evaluation

6.1 Setup

Claims in CLAIMDB are intentionally derived from insights that *combine* information across millions of records through operations such as averaging, grouping and sorting (Section 3.3). At this scale, verification cannot be done by "reading" the evidence in-context: the claim context contains, on average, 110M tokens of structured information—which is far beyond any LLM's context window. As a result, verifiers must reason in *executable* programs that handle, symbolically, the bulk of the compositional reasoning that is required.

Our evaluation primarily focuses on tool-calling agents (Mialon et al., 2024; Wang et al., 2024a), which we view as a very strong and practical solution for this setting. We use Google's MCP toolbox for databases (Buvaraghan and Egan, 2025) and give each agent a single tool: the ability to execute arbitrary SQL queries against the database associated with the claim at hand. The only constraint we impose is a *generous* limit of 20 tool calls; exceeding this counts as a failed run.

There also exist other setups such as coding agents (Wang et al., 2024b) that interact with data via Python and pandas. However, prior work has shown such approaches to be brittle for the smaller LLMs (below 70B) we consider in this paper (Sun et al., 2024). We therefore leave a systematic study of potential alternatives—from other *tool* choices to different paradigms altogether—for future work.

[2]The test sets are drawn from claims coming from the dev split of BIRD as it has received substantial cleanup effort in prior work (e.g., Wretblad et al., 2024; Liu et al., 2025).

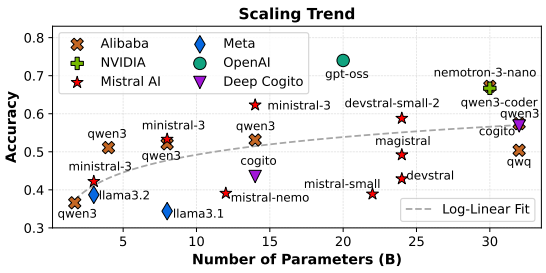


Figure 5: Final accuracy vs. model size (open-source).

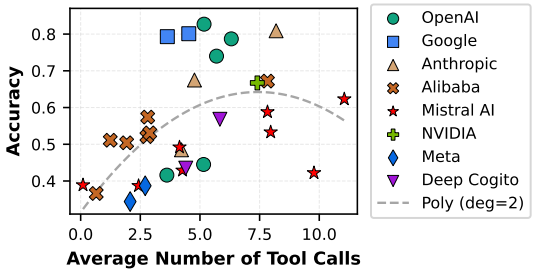


Figure 6: Average number of tool calls per model on the public `test` set of CLAIMDB. Performance generally degrades as LLM-database interactions become longer.

We use a single prompt (Figure 22 in the Appendix) and we conduct extensive experiments with 30 state-of-the-art proprietary and open-source LLMs. Information and hyperparameters (e.g., decoding) are illustrated in Table 3 of the Appendix.

6.2 Metrics

CLAIMDB is a three-way classification benchmark with labels *entailed* (E), *contradicted* (C), and *not-enough-info* (NEI). For each label, we compute precision, recall, and F1 by treating that label as the positive class and the remaining two as negatives.

Precision measures the proportion of predictions for a given label that are correct, while *recall* measures the fraction of true instances of that label that are correctly identified. F1 is their harmonic mean. We report all metrics, but we primarily focus on accuracy and Macro-F1 $= \frac{1}{3}(\mathrm{F1}_{\mathrm{E}} + \mathrm{F1}_{\mathrm{C}} + \mathrm{F1}_{\mathrm{NEI}})$.

6.3 Discussion

Scaling Trends. Figure 5 plots accuracy against model size. We notice the familiar pattern of larger models performing better, but with improvements that are marginal and roughly log-linear.

Long Sessions Hurt. Long LLM-database interactions have inherent fragility: as they grow longer, models increasingly lose focus due to the large

LLM Backbone	Precision			Recall			F1			Macro-F1	Acc.
	Ent.	Contr.	NEI	Ent.	Contr.	NEI	Ent.	Contr.	NEI		
`gpt-4o-mini` (2024)	0.777	0.666	0.679	0.577	0.656	0.107	0.662	0.661	0.185	0.503	0.445
`gpt-4.1-nano` (2025a)	0.647	0.641	0.376	0.165	0.124	0.952	0.263	0.208	0.539	0.337	0.416
`gpt-5-nano` (2025b)	0.815	0.710	0.874	0.742	0.900	0.720	0.777	0.794	0.790	0.787	0.787
`gpt-5-mini` (2025b)	0.875	0.713	0.963	0.754	0.952	0.777	0.810	0.815	**0.860**	**0.828**	**0.827**
`gpt-oss:20b` (2025)	0.851	0.735	0.676	0.669	0.688	0.862	0.749	0.710	0.758	0.739	0.740
`gemini-2.5-flash` (2025)	0.841	0.734	0.830	0.685	0.825	0.869	0.755	0.777	0.849	0.793	0.793
`gemini-3-flash` (2025)	0.754	0.750	0.962	0.799	0.934	0.673	0.776	**0.832**	0.792	0.800	0.801
`claude-3-haiku` (2024)	0.572	0.556	0.433	0.261	0.435	0.756	0.359	0.488	0.550	0.466	0.485
`claude-3-5-haiku` (2024)	0.775	0.626	0.661	0.580	0.743	0.702	0.663	0.680	0.681	0.675	0.675
`claude-haiku-4-5` (2025)	0.836	0.711	0.983	0.796	0.952	0.682	**0.815**	0.814	0.805	0.811	0.809
`qwen3:1.7b` (2025)	0.645	0.533	0.351	0.060	0.048	0.982	0.110	0.089	0.518	0.239	0.366
`qwen3:4b` (2025)	0.530	0.554	0.492	0.450	0.218	0.860	0.487	0.312	0.626	0.475	0.511
`qwen3:8b` (2025)	0.785	0.617	0.450	0.306	0.311	0.940	0.441	0.414	0.608	0.488	0.521
`qwen3:14b` (2025)	0.812	0.744	0.438	0.285	0.360	0.943	0.422	0.485	0.599	0.502	0.531
`qwen3:32b` (2025)	0.732	0.667	0.491	0.393	0.459	0.866	0.512	0.544	0.626	0.561	0.574
`qwen3-coder:30b` (2025)	0.744	0.625	0.660	0.646	0.659	0.711	0.691	0.641	0.685	0.672	0.672
`qwq:32b` (2025)	0.500	0.558	0.472	0.336	0.508	0.667	0.402	0.532	0.552	0.495	0.504
`mistral-nemo:12b` (2024)	0.380	0.446	0.381	0.381	0.211	0.577	0.381	0.287	0.459	0.376	0.391
`ministral-3:3b` (2025b)	0.511	0.500	0.390	0.276	0.163	0.821	0.359	0.246	0.529	0.378	0.422
`ministral-3:8b` (2025b)	0.633	0.557	0.484	0.408	0.396	0.792	0.496	0.463	0.600	0.520	0.533
`ministral-3:14b` (2025b)	0.612	0.638	0.624	0.598	0.580	0.690	0.605	0.608	0.655	0.623	0.623
`mistral-small:22b` (2025c)	0.423	0.526	0.375	0.222	0.060	0.878	0.291	0.108	0.525	0.308	0.389
`magistral:24b` (2025a)	0.682	0.714	0.419	0.309	0.257	0.905	0.426	0.378	0.573	0.459	0.492
`devstral:24b` (2025a)	0.415	0.624	0.399	0.291	0.221	0.771	0.342	0.326	0.526	0.398	0.429
`devstral-small-2` (2025b)	0.602	0.712	0.521	0.610	0.447	0.705	0.606	0.549	0.599	0.585	0.588
`nemotron-3-nano:30b` (2025)	0.714	0.726	0.612	0.652	0.601	0.747	0.681	0.658	0.673	0.671	0.667
`llama3.1:8b` (2024)	0.341	0.433	0.337	0.222	0.079	0.726	0.269	0.133	0.461	0.288	0.344
`llama3.2:3b` (2024)	0.428	0.412	0.366	0.372	0.100	0.685	0.398	0.161	0.477	0.345	0.387
`cogito:14b` (2025)	0.758	0.701	0.384	0.141	0.184	0.973	0.238	0.292	0.551	0.360	0.435
`cogito:32b` (2025)	0.797	0.564	0.506	0.354	0.556	0.792	0.491	0.560	0.617	0.556	0.568

Table 2: We report per-label precision, recall, and F1, along with macro-F1 and accuracy.

amounts of information. In this setting, a single bad decision—a single *careless* query—can flood the model with hundreds of thousands of tokens.

Figure 6 plots, for each of the 30 models, the *average number of tool calls* over the 1,000 `test` examples. We fit a second-order polynomial and observe that the top-performing models typically average roughly 4–8 tool calls. Longer (or shorter) than that leads to degraded performance. The complete tool-call distributions are shown in Figure 12.

Open-Source Models Struggle. Table 2 reports results for all 30 models. Across both accuracy and macro-F1, `gpt-5-mini` performs best, followed by `claude-haiku-4.5` and `gemini-3-flash`. Out of all evaluated models, more than half (17 out of 30) stay below 55% accuracy and Macro-F1.

Open-source models generally struggle, with the exception of `gpt-oss-20b` which performs on par with top proprietary models. Aside from `gpt-oss`, no other open-source model (out of the 20 remaining) exceeds 68% accuracy or macro-F1. Overall, this highlights that there is room for improvement for open-source models—to catch up—in reasoning over large-scale data. The CLAIMDB benchmark is a clear step toward this goal.

No Evidence of Data Contamination. A common concern with modern benchmarks is the risk of data contamination, where models may have seen parts of the data during training, making the benchmark obsolete (Samuel et al., 2025). To investigate this, we evaluate the best-performing model, `gpt-5-mini`, without access to external tools, where it must label each claim using only its parametric knowledge. We find that performance is near random, with a macro-F1 of 0.253 and accuracy of 0.367, suggesting that CLAIMDB cannot be solved using parametric knowledge alone.

NEI is Poorly Handled. We observe that both closed- and open-source models exhibit *opposite but degenerate* behaviors with respect to the NEI

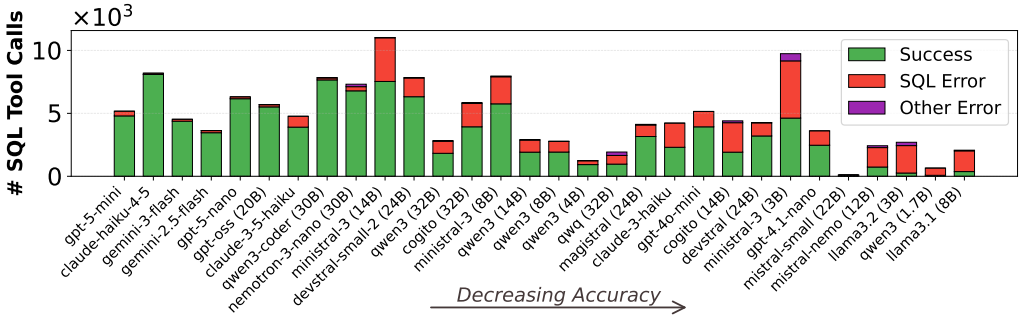


Figure 7: Our agents are equipped with a single tool: the ability to execute SQL queries on the associated database. We report the success rates of the SQL queries per model. More specifically, "Success" denotes queries that execute and return results; "SQL Error" denotes queries with syntax or logical issues; and "Other Error" includes all remaining failures. Notably, top-performing models achieve high query success rates, suggesting that failures are not superficial (e.g., syntax) but instead reflect deeper breakdowns in reasoning over large structured data.

label. Figure 8 shows normalized confusion matrices for two top-performing proprietary models and two strong open-source alternatives.

Both `gpt-5-mini` and `claude-haiku-4.5` are strongly biased *against* predicting NEI. When the ground-truth label is entailed or contradicted, they almost never predict NEI (close to 0%), indicating a strong, inherent reluctance. Even when the correct label *is* NEI, they hedge heavily, spreading probability mass across the other two labels.

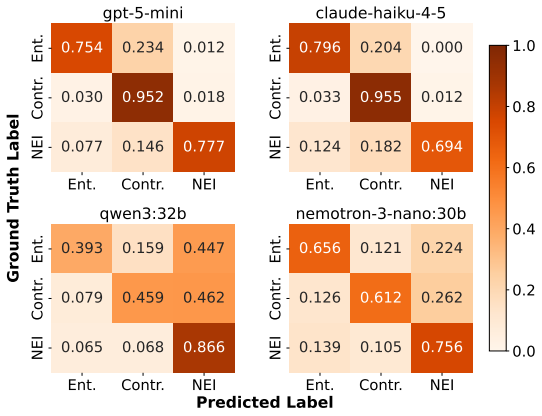


Figure 8: Confusion matrices (normalized) for representative proprietary and open-source models. The primary difference lies in how models handle NEI: proprietary models are *biased* against abstention, while open-source models predict it excessively (full results in Figure 11).

Both `qwen3` and `nemotron-3-nano` display the opposite behavior: they over-predict NEI. When the ground truth is entailed or contradicted, NEI is predicted roughly half the time for `qwen3`, with a milder but similar tendency for `nemotron-3-nano`.

Notably, the performance gap between closed- and open-source models is driven mostly by how NEI is handled. However, neither behavior is desirable: proprietary models effectively avoid abstention, while open-source models over-abstein.

Non-Superficial Failures. Our neuro-symbolic baseline operates as a multi-step process driven by SQL queries over the underlying database. Hence, inspecting these queries provides insight into where model failures occur. In Figure 7, we plot the distribution of whether executed SQL queries succeed or fail. The two best-performing models, `gpt-5-mini` and `claude-haiku-4.5`, achieve SQL query success rates of 93% and 99%, respectively. For both models, the remaining failed queries are evenly distributed across correct and incorrect predictions. This suggests that, for stronger models, failures are not superficial (e.g., syntax), but instead reflect breakdowns in reasoning—specifically, in how the model composes queries to navigate the database.

7 Conclusion

In this work, we introduce CLAIMDB, a fact-verification benchmark over large-scale structured data. It contains 80 unique databases, each with an average of 11 tables and 4.5M records (roughly 110M tokens when converted to text). Each claim is *grounded* in evidence deliberately composed of millions of records across multiple tables. CLAIMDB is a clear step toward real-world claim verification—mirroring the various *challenges* of high-stakes fact-checking in our contemporary world.

Lastly, we conduct extensive experiments with 30 state-of-the-art open- and closed-source models, and find that more than half remain below 55% accuracy. We further uncover systematic failures around *abstention*—the ability to admit there is not enough evidence to decide—raising concerns about the reliability of LLMs in high-stakes verification.

8 Limitations

Reliance on BIRD. Our benchmark is derived from NL-to-SQL pairs in the BIRD benchmark (Section 3.2). As a result, any annotation error in BIRD directly propagates to CLAIMDB. For example, if the NL question is *"How old was the oldest mayor of Seattle when elected?"* but the associated SQL query instead computes the answer for *Portland*, any claims generated from this pair would be incorrectly labeled. To mitigate this risk for public evaluation and model development, we construct our `test` splits exclusively from claims derived from the `dev` split of BIRD, which has undergone substantial subsequent cleanup by prior work (e.g., Wretblad et al., 2024; Liu et al., 2025). Importantly, our dependency on BIRD is not irreversible: as additional errors in BIRD are identified, the corresponding derived claims can be removed from CLAIMDB on-the-fly.

Single Evidence Modality. CLAIMDB deliberately focuses on structured data as its primary evidence source (which may include numerical, categorical, temporal, and short textual fields). While real-world fact verification often involves multi-modal evidence—such as free-form text, reports, news articles, charts, and other unstructured sources—structured data remains a critical yet comparatively underexplored modality in existing benchmarks. By isolating this setting, CLAIMDB enables targeted evaluation of reasoning over complex structured evidence. We acknowledge that this focus alone does not capture the full breadth of real-world verification, and view extension to multi-modal evidence as a natural future direction.

Snapshot Validity and Evaluation Scope. Entailment, contradiction, and NEI are assessed with respect to the fixed database snapshot used to construct the benchmark. Because the underlying databases come from established public dataset releases—and are therefore static and potentially dated—answers may differ if a system incorporates newer external information. For example, a query about the *"bottom-5 enrollment schools"* could change, even if the claim was correct in the snapshot used for claim generation. As with many ML benchmarks, evaluation should be interpreted relative to the provided database state (Li et al., 2023) rather than a live, continuously updated world.

Evaluation Solely on SQL. Our evaluation uses agents equipped with a single tool: the ability to execute SQL queries. Performance therefore depends not only on a model's reasoning ability, but also on its familiarity with SQL. One of the reasons we choose SQL is because, as a language, it makes the required data operations—such as filtering, grouping, aggregation, and ordering—explicit, isolating the core reasoning challenge. But, some models may perform worse than others simply because they are worse at writing SQL (e.g., due to their pre-training data).

9 Ethical Considerations

Data Licensing. CLAIMDB bases all claims on NL-to-SQL pairs coming from the BIRD benchmark which is released under CC BY-SA 4.0 license. CLAIMDB is released under the same license, in accordance with the original terms.

Privacy. We do not anonymize any part of CLAIMDB. CLAIMDB includes *contradicted* claims, which are synthetically generated by perturbing facts and are explicitly labeled as false within the benchmark. These claims are not intended to introduce new real-world information about entities, but solely to support controlled evaluation of real-life fact-verification systems.

10 Acknowledgments

We thank the anonymous reviewers for their constructive feedback on this paper. This work was supported by NSF III 2507117, NSF SHF 2312195, and NSF IIS 2314527. We thank Dean Light for many insightful discussions throughout the review process. We also thank the UW Database Group members for helpful feedback and David Alexander for conversations on benchmark performance.

References

Marah I Abdin, Jyoti Aneja, Harkirat S. Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *CoRR*, abs/2412.08905.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: fact extraction and verification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1,*

NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6416–6432. Association for Computational Linguistics.

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku.

Anthropic. 2025. Introducing Claude Haiku 4.5.

Anab Maulana Barik, Wynne Hsu, and Mong-Li Lee. 2024. Time matters: An end-to-end solution for temporal claim verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 657–664. Association for Computational Linguistics.

Anab Maulana Barik, Wynne Hsu, and Mong-Li Lee. 2025. Chronofact: Timeline-based temporal fact verification. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 8031–8039. ijcai.org.

Adrián Bazaga, Pietro Lio, and Gos Micklem. 2024. Unsupervised pretraining for fact verification by language model distillation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jean-Flavien Bussotti, Luca Ragazzi, Giacomo Frisoni, Gianluca Moro, and Paolo Papotti. 2024. Unknown claims: Generation of fact-checking training examples from unstructured and structured data. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 12105–12122. Association for Computational Linguistics.

Jean-Flavien Bussotti, Enzo Veltri, Donatello Santoro, and Paolo Papotti. 2023. Generation of training examples for tabular natural language inference. *Proc. ACM Manag. Data*, 1(4):243:1–243:27.

Hamsa Buvaraghan and Derek Egan. 2025. MCP Toolbox for Databases: Simplify AI Agent Access to Enterprise Data.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Preetam Prabhu Srikar Dammu, Himanshu Naidu, Mouly Dewan, YoungMin Kim, Tanya Roosta, Aman Chadha, and Chirag Shah. 2024. Claimver: Explainable claim-level verification and evidence attribution of text through knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 13613–13627. Association for Computational Linguistics.

Deep Cogito. 2025. Cogito v1 Preview Introducing IDA as a path to general superintelligence.

Jacob Daniel Devasier, Akshith Reddy Putta, Rishabh Mediratta, and Chengkai Li. 2025. Task-oriented automatic fact-checking with frame-semantics. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 13825–13842. Association for Computational Linguistics.

Jacob Daniel Devasier, Akshith Reddy Putta, Qing Wang, Alankrit Moses, and Chengkai Li. 2026. Frame-guided synthetic claim generation for automatic fact-checking using high-volume tabular data. *CoRR*, abs/2601.17232.

Tulsee Doshi. 2025. Gemini 3 Flash: frontier intelligence built for speed. Google Blog.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10932–10952. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2309–2324. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow,

Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 79 others. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. 2025. Abstentionbench: Reasoning llms fail on unanswerable questions. *CoRR*, abs/2506.09038.

Anoop Kotha, Julian Lee, and Eric Zakariasson. 2025. Gpt-5 prompting guide.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2261–2273. Association for Computational Linguistics.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, and 28 others. 2025. Gemini embedding: Generalizable embeddings from gemini. *CoRR*, abs/2503.07891.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. 2025. Nl2sql-bugs: A benchmark for detecting semantic errors in NL2SQL translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pages 5662–5673. ACM.

Xinyuan Lu, Liangming Pan, Qian Liu, Preslav Nakov, and Min-Yen Kan. 2023. SCITAB: A challenging benchmark for compositional reasoning and claim verification on scientific tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7787–7813. Association for Computational Linguistics.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mistral AI. 2024. Mistral NeMo.

Mistral AI. 2025a. Devstral.

Mistral AI. 2025b. Introducing Mistral 3.

Mistral AI. 2025c. Mistral Small 3.1.

MPD. 2025. Crime Incidents 2025. Accessed: 2026-01-01.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2006–2029. Association for Computational Linguistics.

NVIDIA, :, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, and 340 others. 2025. Nvidia nemotron 3: Efficient and open intelligence. *Preprint*, arXiv:2512.20856.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI. 2025a. Introducing GPT-4.1 in the API.

OpenAI. 2025b. Introducing GPT-5.

PBS. 2022. Analysis: What the Feds largest interest rate hike in decades means for you.

Akshith Reddy Putta, Jacob Daniel Devasier, and Chengkai Li. 2026. Casefacts: A benchmark for legal fact-checking and precedent retrieval. *CoRR*, abs/2601.17230.

Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning.

Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, Yihan Wang, and 80 others. 2025a. Magistral. *CoRR*, abs/2506.10910.

Abhinav Rastogi, Adam Yang, Albert Q. Jiang, Alexander H. Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Anmol Agarwal, Andy Ehrenberg, Andy Lo, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, and 84 others. 2025b. Devstral: Fine-tuning language models for coding agent applications. *Preprint*, arXiv:2509.25193.

Vinay Samuel, Yue Zhou, and Henry Peng Zou. 2025. Towards data contamination detection for modern large language models: Limitations, inconsistencies, and oracle challenges. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 5058–5070. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. 2023. Averitec: A dataset for real-world claim verification with evidence from the web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in LLMs. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Marek Strong and Andreas Vlachos. 2025. TSVer: A benchmark for fact verification against time-series evidence. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29894–29914, Suzhou, China. Association for Computational Linguistics.

Zhihong Sun, Chen Lyu, Bolun Li, Yao Wan, Hongyu Zhang, Ge Li, and Zhi Jin. 2024. Enhancing code generation performance of smaller models by distilling the reasoning ability of llms. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 5878–5895. ELRA and ICCL.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. 2024. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. *CoRR*, abs/2406.12624.

The White House. 2025. President Trump Holds a Press Conference, Aug. 11, 2025.

Michael Theologitis and Dan Suciu. 2025. Thucy: An LLM-based Multi-Agent System for Claim Verification across Relational Databases. *Preprint*, arXiv:2512.03278.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

U.S. BLS. Consumer Price Index (CPI) Databases. Accessed: 2026-01-01.

Pat Verga, Sebastian Hofstätter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. *CoRR*, abs/2404.18796.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7534–7550. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024.

Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval@ACL/IJCNLP 2021, Virtual Event / Bangkok, Thailand, August 5-6, 2021*, pages 317–326. Association for Computational Linguistics.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024c. Chain-of-table: Evolving tables in the reasoning chain for table understanding.

In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

David H. D. Warren and Fernando C. N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Am. J. Comput. Linguistics*, 8(3-4):110–122.

WRAL. 2023. Fact check: Biden says U.S. inflation rate 'lowest' of leading economies.

Niklas Wretblad, Fredrik Gordh Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the effects of noise in text-to-sql: An examination of the bird-bench benchmark. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024 - Short Papers, Bangkok, Thailand, August 11-16, 2024*, pages 356–369. Association for Computational Linguistics.

xAI. 2025. Grok 3 Beta — The Age of Reasoning Agents.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. 2025. Justice or prejudice? quantifying biases in llm-as-a-judge. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Alex L. Zhang, Tim Kraska, and Omar Khattab. 2026. Recursive language models. *Preprint*, arXiv:2512.24601.

Yilun Zhao, Yitao Long, Tintin Jiang, Chengye Wang, Weiyuan Chen, Hongjun Liu, Xiangru Tang, Yiming Zhang, Chen Zhao, and Arman Cohan. 2024. Finver: Explainable claim verification over long and hybrid-content financial documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 14739–14752. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Mingyang Zhou, Lingyu Zhang, Sophia Horng, Maximillian Chen, Kung-Hsiang Huang, and Shih-Fu Chang. 2025. M^2-tabfact: Multi-document multimodal fact verification with visual and textual representations of tabular data. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 26239–26256. Association for Computational Linguistics.

A Tokens Per Database

In this section, we estimate the number of tokens required to represent a database in context. This analysis supports one of our central claims: in ClaimDB, the context of each claim (i.e., its underlying database) exceeds the context window of modern LLMs, which forces solutions to reason in executable programs (e.g., SQL).

To obtain this estimate, we take a straightforward approach. Each database in ClaimDB consists of multiple tables. For each database, we export all of its tables as CSV files (header row followed by all data rows), concatenate them into a single text file, and count the resulting tokens using the tokenizer of the `gpt-5` series. We use this tokenizer as a representative modern LLM tokenizer.

Importantly, this estimate is conservative for two reasons. First, it does not encode the relationships and constraints present in the database (e.g., keys, foreign keys, and other schema-level structure). Second, providing structured data in context to an LLM typically requires an explicit representation—such as JSON (Singha et al. 2023)—that clearly associates each value with its column and preserves structure. This additional encoding would substantially increase the token count. The reported numbers should therefore be interpreted as a lower bound. Figure 9 shows the exact token counts for each of the 80 databases.

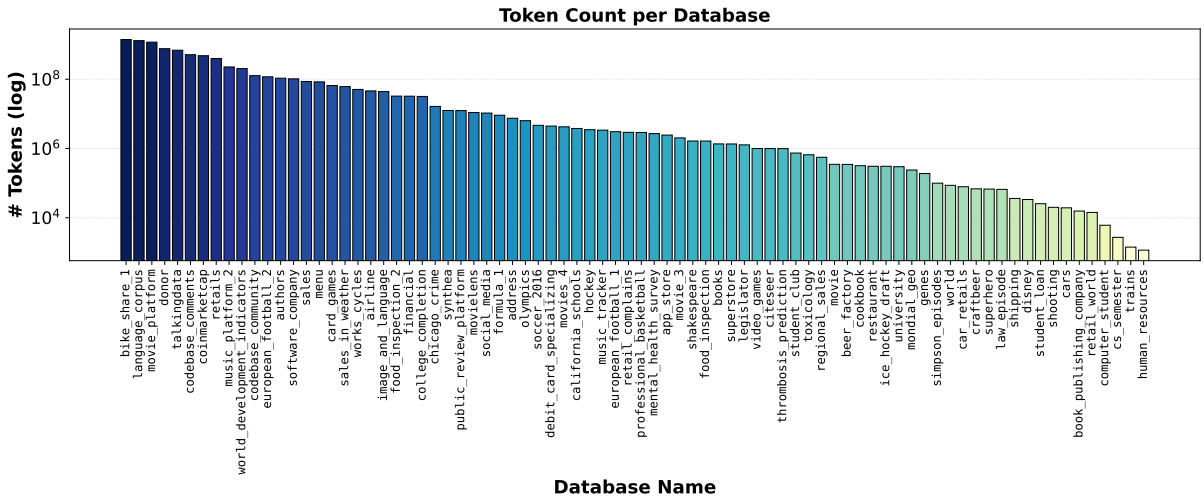


Figure 9: Token count per database (log scale), computed using the tokenizer of the `gpt-5` series. Each bar corresponds to one of the 80 databases in CLAIMDB. For more than half of the databases, naive in-context "reading" of the full evidence is practically impossible. Notably, the reported average of 110M tokens of context per claim in Table 1 reflects the fact that claims are unevenly distributed across databases. In particular, we end up with a lot more claims based on larger databases which means the per-claim token average is higher than a simple average over the 80 databases.

B Evaluation

B.1 Evaluation Setup

Table 3 illustrates the hyperparameters of the evaluation setup.

B.2 Structured Outputs

A few years ago, getting *structured outputs*—responses that adhere to strict user-defined JSON schemas—back from LLMs was a painful process. Making it work would typically require careful prompting and repeated retries to get outputs in the right format. At the same time, this feature was becoming increasingly important for any serious application that relied on LLMs.

Thankfully, the situation has improved substantially. Most major AI providers (e.g., OpenAI, Anthropic, Google) now support some form of *structured outputs*. There are multiple ways to implement this, but one of the most reliable is grammar-constrained decoding (Geng et al., 2023). In this approach, the JSON schema is compiled into a context-free grammar, and during decoding the model's logits are masked so that only grammar-valid tokens can be generated.

Concretely, this means that when a JSON schema is provided, the model is *guaranteed* to produce a syntactically valid response. For example, if we request an `enum` over three possible verdicts (entailed, contradicted, NEI), the output is guaranteed to be one of these values. This greatly simplifies evaluation and removes an entire class of failure modes unrelated to model reasoning.

This level of control is only possible because model providers have direct access to their own decoding process. As a result, most proprietary APIs expose structured output functionality directly. However, when using agentic frameworks (e.g., OpenAI Agent SDK, Pydantic AI, Google ADK, etc.) that connect to third-party or open-source models through adapters such as LiteLLM, this guarantee often breaks down.

For example, some leading agentic frameworks simulate structured outputs by injecting a prompt that forces the model to call a predefined tool for its final answer[3], rather than enforcing the schema at decoding time when the Grammar Constraint Decoding (GCD) option is available (e.g., on open-source models). Aside from bloating the model context, this approach is inherently brittle: the LLM can simply ignore the instruction.

This creates an evaluation asymmetry. Models without guaranteed structured decoding may fail due to formatting errors even when their underlying prediction is correct, while models with GCD will always produce a valid label (which implies a 33% accuracy floor). Since our goal is not to benchmark how well models, vendors and agentic frameworks

[3] https://github.com/pydantic/pydantic-ai/issues/242, https://github.com/openai/openai-agents-python/issues/1778#issuecomment-3316092585

Model	Context Window	temp.	topK	topP
gpt-4o-mini	128K	1.0	–	1.0
gpt-4.1-nano	1M	1.0	–	1.0
gpt-5-nano	400K	1.0	–	1.0
gpt-5-mini	400K	1.0	–	1.0
gpt-oss:20b	128K	1.0	–	1.0
gemini-2.5-flash	1M	1.0	64	0.95
gemini-3-flash	1M	1.0	64	0.95
claude-3-haiku	200K	1.0	–	–
claude-3-5-haiku	200K	1.0	–	–
claude-haiku-4-5	200K	1.0	–	–
qwen3:1.7b	40K	0.6	20	0.95
qwen3:4b	246K	0.6	20	0.95
qwen3:8b	40K	0.6	20	0.95
qwen3:14b	40K	0.6	20	0.95
qwen3:32b	40K	0.6	20	0.95
qwen3-coder:30b	256K	0.7	20	0.8
qwq:32b	40K	1	40	0.95
mistral-nemo:12b	1M	0.0	–	–
ministral-3:3b	256K	0.15	50	90
ministral-3:8b	256K	0.15	50	90
ministral-3:14b	256K	0.15	50	90
mistral-small:22b	128K	0.0	–	–
magistral:24b	39K	0.7	–	0.95
devstral:24b	128K	0.0	–	–
devstral-small-2:24b	384K	0.15	–	–
nemotron-3-nano	1M	1.0	–	1.0
llama3.2:3b	128K	0.0	–	–
llama3.1:8b	128K	0.0	–	–
cogito:14b	128K	0.0	–	–
cogito:32b	128K	0.0	–	–

Table 3: Extra information and hyperparameters. Unavailable or N/A values are shown as "–". Numbers also come from HuggingFace and Ollama.

support structured outputs, but rather how well can models reason over large data, directly penalizing such failures would be unfair.

Our policy is therefore simple. For models where structured outputs are not guaranteed—typically due to limitations of the agentic framework we use—we re-run the test whenever a response violates the expected schema. This occurs in only a small fraction of cases and is usually resolved with one or two re-runs.

C Prompts

In Figures 17, 18 and 19 we showcase the prompts used in creating the claims (Section 3.5). In Figures 20 and 21 we illustrate the two prompts used for the judging process—one for contradicted and entailed claims and one for NEI since the latter has extra rubrics and extra golden context (Section 4.1). Lastly, in Figure 22, we show the single, optimized verifier prompt for all models in our evaluations.

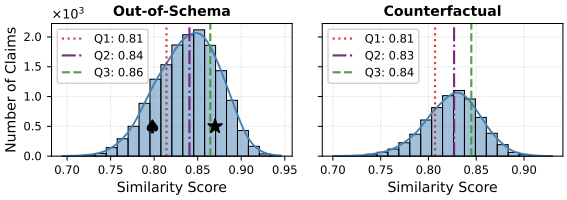


Figure 10: Distribution of similarity scores between generated claims and their gold context for both *counterfactual* and *out-of-schema* claims. These claims can "drift" from the database concepts so we embed them and measure semantic similarity between them and the also embedded golden context. Higher scores indicate that claims stay "closer" to the underlying data concepts. The two highlighted claim examples (♣, ★) are discussed in Section 4.2.

D SQL Queries and ASTs

This section provides concrete examples of SQL queries together with their corresponding abstract syntax trees (ASTs). Figures 13–16 illustrate how the filtering rules described in Section 3.3 are applied in practice.

E Similarity Distributions

We provide the *counterfactual* and *out-of-schema* similarity distributions (Section 4.2) in Figure 10.

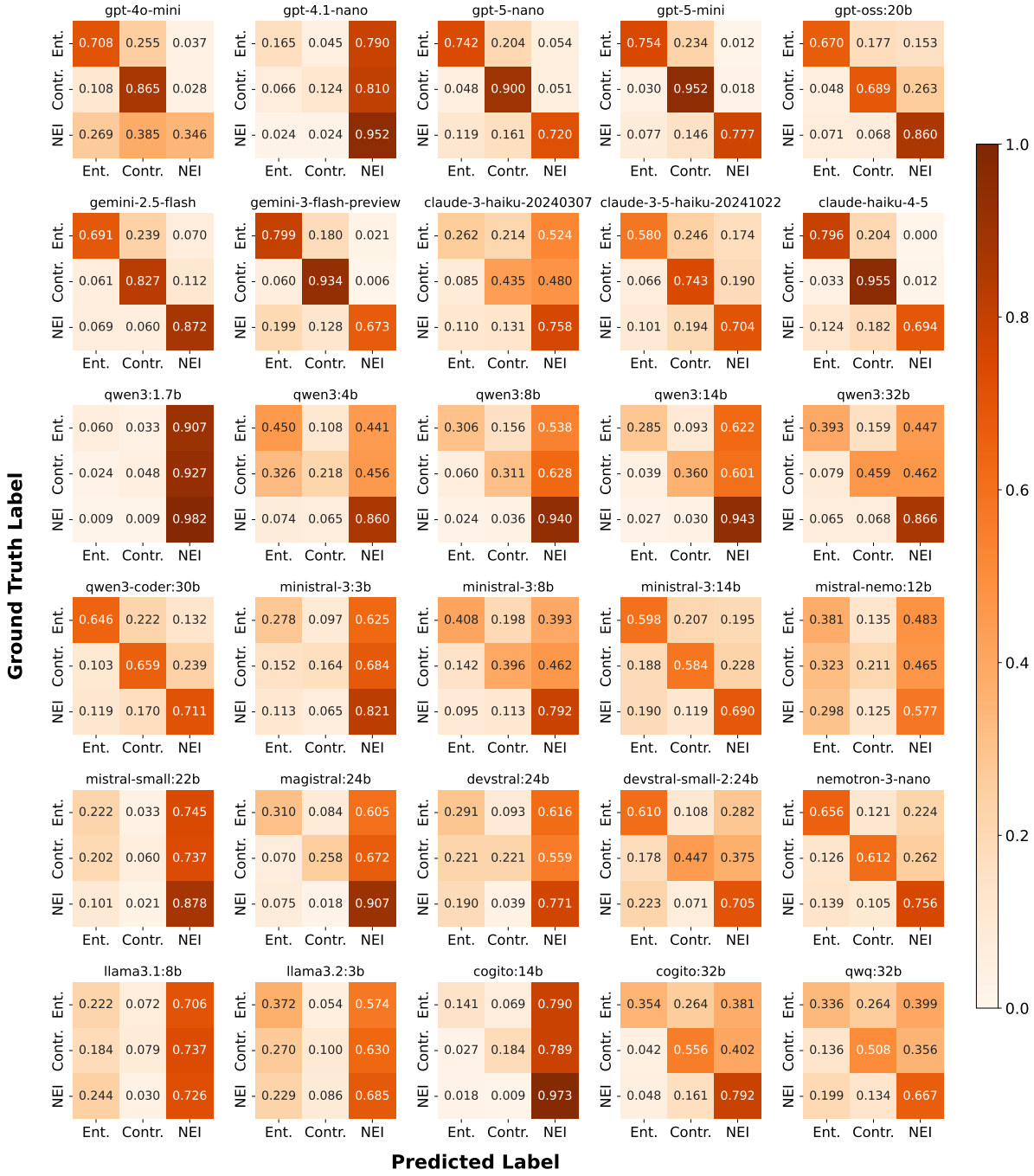


Figure 11: Public `test` set. Confusion matrices (normalized) for all models. One takeaway is that the primary difference in top-performing open- and closed-source models lies in how they handle NEI: proprietary models are *biased* against abstention, while open-source models predict it excessively.

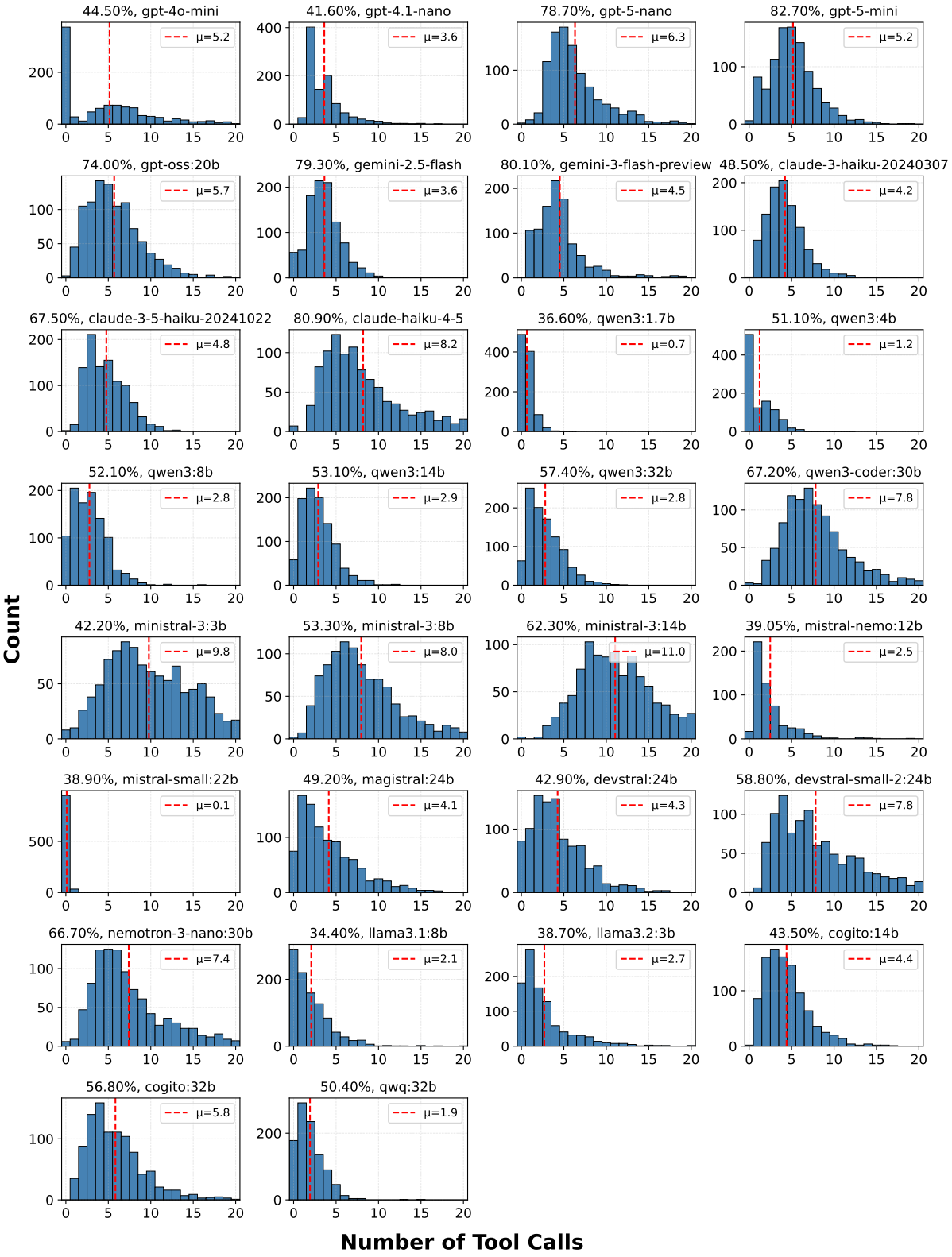


Figure 12: Public `test` set. Distribution of the number of tool calls for each model on the 1,000 examples of the public `test` set of CLAIMDB. The best-performing model is `gpt-5-mini` (0.83 acc.) which gives us insight on *how the distribution of tools calls* should probably look like for a model to do well on our benchmark. Some models often decide not to use a tool call at all, which is obviously a losing strategy as they have no idea of the underlying data environment to answer correctly. We try to avoid this by "forcing" the model to inspect the data first, by including instructions like *"Use the available tools to query the database and gather evidence before making a decision"* and *"You should always start by querying the database for the schema"* (Figure 22) which are ignored.

```sql
SELECT business_id
FROM Business
WHERE state LIKE 'AZ' AND stars = 5
```

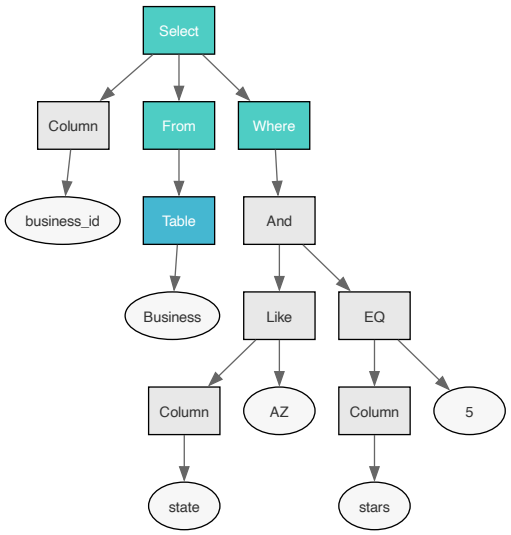


Figure 13: ✗ **Not Compositional**. The SQL query (top) and its AST (bottom). The query is excluded since it only applies local filters (`state`, `stars`) and does not involve aggregations, joins, or other operations that combine information across many records.

```sql
SELECT COUNT(T1.inspection_id)
FROM inspection AS T1 INNER JOIN employee AS T2
    ON T1.employee_id = T2.employee_id
WHERE T2.first_name = 'Lisa' AND
    T2.last_name = 'Tillman' AND
    T1.results = 'Out of Business'
```

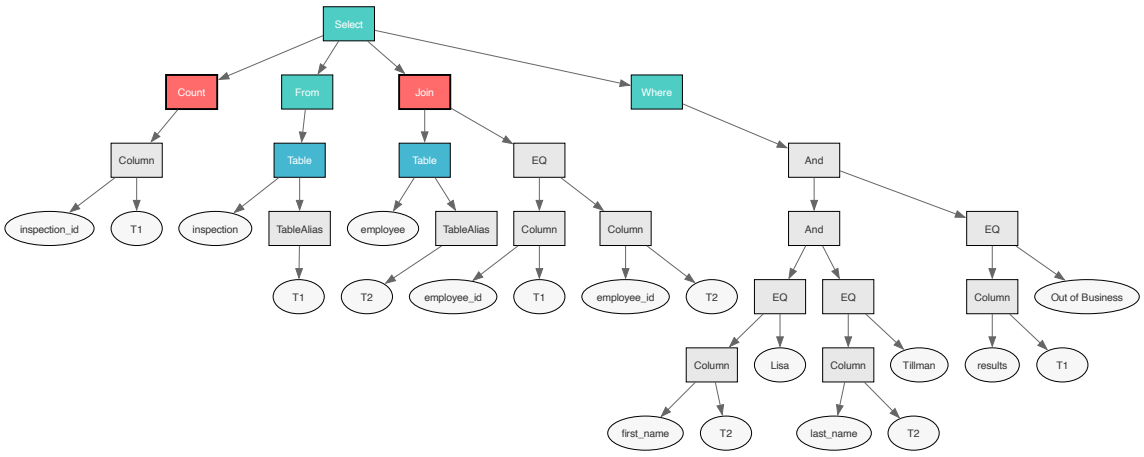


Figure 14: ✓ **Compositional**. The SQL query (top) and its AST (bottom). The query is retained since its AST contains an aggregate function (`COUNT`), which counts a large number of records and collapses them into a single value (the count). Verifying a claim derived from this query therefore requires compositional reasoning over many records, which cannot be performed by naively "reading" the records in-context and counting by an LLM alone.

```sql
SELECT T2.movie_title, T1.user_id,
    T1.rating_score, T1.critic
FROM ratings AS T1 INNER JOIN movies AS T2
    ON T1.movie_id = T2.movie_id
WHERE T1.critic IS NOT NULL
```

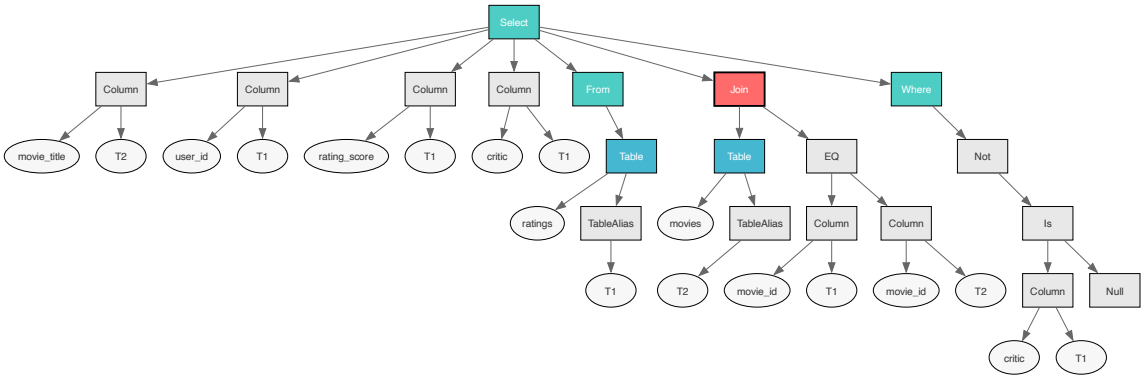


Figure 15: ✗ **Not Compositional**. The SQL query (top) and its AST (bottom). The query is excluded since it does not contain aggregations, orderings, window functions, or joins involving *three or more* tables (see Section 3.3).

```sql
SELECT T1.title
FROM movie AS T1
    INNER JOIN movie_keywords AS T2
        ON T1.movie_id = T2.movie_id
    INNER JOIN keyword AS T3
        ON T2.keyword_id = T3.keyword_id
WHERE T3.keyword_name = 'extremis'
```

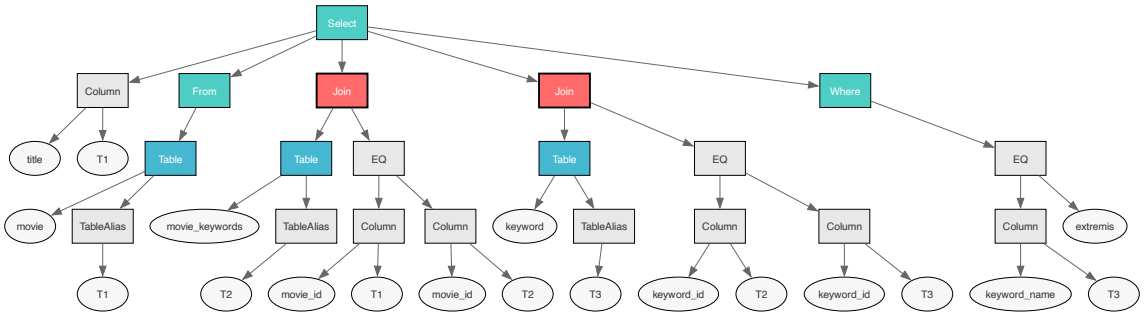


Figure 16: ✓ **Compositional**. The SQL query (top) and its AST (bottom). The query is retained since it joins three tables together—the tables `movie` aliased as `T1`, `movie_keywords` aliased as `T2`, and `keyword` aliased as `T3`—which is a compositional operation as defined in Section 3.3. A verifier must explicitly combine these three tables in order to verify a claim derived from this SQL query.

```python
from pydantic import BaseModel, Field

class ContradictedClaim(BaseModel):
    contradicted_claim: str = Field(
        ...,
        description=(
            "A contradicted claim."
        )
    )

class ContradictedClaimCollection(BaseModel):
    collection: list[ContradictedClaim]
```

```
## Role
You are a **misleading** spokesperson  **in a controlled evaluation
setting**.

## Task
Given the following inputs:
- A question
- Its correct answer
- The data domain
- Optional external knowledge (clarifications)

Your task is to produce natural language claims that are factually
incompatible with the provided answer. In other words, any reader who
knows the correct answer would judge your claim to be false.

## Requirements
- Each claim must be self-contained and must not use opaque references
to earlier context (e.g., "the answer," "the question," "the earlier
claim", etc.). Instead, any needed context should be stated explicitly
within each claim.
- Each claim must contradict or be factually incompatible with the
answer, directly or indirectly.
- Do not restate or explain the external knowledge; assume it is
already known to the reader.
- Produce between 1 and 3 claims.

## Example

### Input
{
  "question": "Which three districts recorded the highest graduation
  rates in 2022?",
  "answer": [
    {
      "DistrictName": "Redwood Coast Unified",
      "GradRate": 0.97
    },
    {
      "DistrictName": "Sierra Vista Union",
      "GradRate": 0.96
    },
    {
      "DistrictName": "Mission Creek Unified",
      "GradRate": 0.95
    }
  ],
  "domain": "California Schools",
  "external-knowledge": "GradRate = Number of graduates / Total number
  of eligible seniors"
}

### Output
Redwood Coast Unified did not lead California's graduation rankings in
2022 — it was Riverbend Joint Unified that posted the top rate.

Sierra Vista Union is no longer among the highest graduation-rate
districts in 2022.

Fairmont Hills Unified surpassed Redwood Coast Unified with 98% of its
eligible seniors graduating in 2022, according to data in CA.
```

Figure 17: Prompt (1-shot) for generating contradicted claims. It is highly optimized for `gpt-5` specifically, by following the work of Kotha et al. (2025). The *output schema* is provided here as python code as we do not know how OpenAI encodes it in the prompt internally. Effectively, the model returns a JSON object that adheres to `ContradictedClaimCollection`, i.e., a collection of contradicted claims. The number of generated claims usually stays between 1-3 as per the instructions.

```python
from pydantic import BaseModel, Field

class EntailedClaim(BaseModel):
    entailed_claim: str = Field(
        ...,
        description=(
            "An entailed claim."
        )
    )

class EntailedClaimCollection(BaseModel):
    collection: list[EntailedClaim]
```

```
## Role
You are an **honest** spokesperson **in a controlled evaluation
setting**.

## Task
Given the following inputs:
- A question
- Its correct answer
- The data domain
- Optional external knowledge (clarifications)

Your task is to produce natural language claims that are consistent
with and supported by the provided answer. In other words, any reader
who knows the correct answer would judge your claim to be true.

## Requirements
- Each claim must be self-contained and must not use opaque references
to earlier context (e.g., "the answer," "the question," "the earlier
claim", etc.). Instead, any needed context should be stated explicitly
within each claim.
- Each claim must follow from or be fully supported by the answer,
directly or indirectly.
- Do not restate or explain the external knowledge; assume it is
already known to the reader.
- Produce between 1 and 3 claims.

## Example

### Input
{
  "question": "Which three districts recorded the highest graduation
  rates in 2022?",
  "answer": [
    {
      "DistrictName": "Redwood Coast Unified",
      "GradRate": 0.97
    },
    {
      "DistrictName": "Sierra Vista Union",
      "GradRate": 0.96
    },
    {
      "DistrictName": "Mission Creek Unified",
      "GradRate": 0.95
    }
  ],
  "domain": "California Schools",
  "external-knowledge": "GradRate = Number of graduates / Total number
  of eligible seniors"
}

### Output
Redwood Coast Unified led California's graduation rankings in 2022 with
a 97% rate.

In 2022, California's strongest graduation results came from Redwood
Coast Unified, which saw 97% of its eligible seniors finish high school.
Sierra Vista Union and Mission Creek Unified followed closely, with
graduation rates of 96% and 95%, respectively.

Mission Creek Unified achieved a graduation rate of 95% in 2022,
placing it among California's top three districts. It ranked just
behind Redwood Coast Unified and Sierra Vista Union.
```

Figure 18: Prompt (1-shot) for generating entailed claims. It is highly optimized for `gpt-5` specifically, by following the work of Kotha et al. (2025). The *output schema* is provided here as python code as we do not know how OpenAI encodes it in the prompt internally. Effectively, the model returns a JSON object that adheres to `EntailedClaimCollection`, i.e., a collection of entailed claims. The number of generated claims usually stays between 1-3 as per the instructions.

```python
from pydantic import BaseModel, Field
from typing import Literal

class NoInfoClaim(BaseModel):
    no_info_claim: str = Field(
        ...,
        description="A NOT ENOUGH INFO claim."
    )
    category: Literal[
        "Out-of-Schema", "Subjective", "Counterfactual"
    ] = Field(
        ...,
        description="The category of the NOT ENOUGH INFO claim."
    )

class NoInfoClaimCollection(BaseModel):
    collection: list[NoInfoClaim]
```

```
## Role
You are a neutral spokespearson **in a controlled evaluation setting**.

## Task
Given the following inputs:
- A question
- Its correct answer
- The data domain
- The schema of the database
- Optional external knowledge (clarifications)

Your task is to produce natural language claims whose truth **cannot**
be determined from the database or the given Q/A. That is, even with
full access to both the database and the correct answer, these claims
cannot be definitively verified or falsified.

## Requirements
- Each claim must be self-contained and must not use opaque references
to earlier context (e.g., "the answer," "the question," "the earlier
claim", etc.). Instead, any needed context should be stated explicitly
within each claim.
- Each claim must *not* be entailed or contradicted by the answer,
directly or indirectly.
- Each claim must fall into at least one of these categories:
  1. **Out-of-schema** — involves concepts the database doesn't store
  or represent anywhere in its schema.
  2. **Subjective/evaluative** — expresses opinions or judgments that
  cannot be objectively verified.
  3. **Counterfactual/hypothetical** — describes an imagined or "what
  if" situation that is not reflected in the actual data.
- Produce between 1 and 5 claims.
- Do not restate or explain the external knowledge; assume it is
already known to the reader.
```

Figure 19: Prompt (zero-shot) for generating NEI claims. It is highly optimized for `gpt-5` specifically, by following the work of (Kotha et al., 2025). The *output schema* is provided here as python code as we do not know how OpenAI encodes it in the prompt internally. Effectively, the model returns a JSON object that adheres to `NoInfoClaimCollection`, i.e., a collection of NEI claims. The number of generated claims usually stays between 1-5 as per the instructions.

```python
from pydantic import BaseModel, Field
from typing import Literal

class ClaimQuality(BaseModel):
    label_correct: Literal["yes", "no"] = Field(
        ...,
        description=(
           'Is the assigned label of the claim (ENTAILED/CONTRADICTED) '
            'correct given the gold information? Answer "yes" if the '
           'label of the claim follows from the gold information; "no" '
            'otherwise. If you are unsure, answer "no".'
        )
    )
    free_of_meta_references: Literal["yes", "no"] = Field(
        ...,
        description=(
            'Does the claim avoid meta-references to the question, '
            'answer, or prior text (e.g., "this question", '
            '"the answer above", "as mentioned earlier")? Answer '
            '"yes" if it is completely free of meta-references; '
            '"no" otherwise. References to provided external '
            'knowledge do not count as meta-references.'
        )
    )
    reasoning: str = Field(
        ...,
        description=(
            'Brief explanation (1-2 sentences) justifying your '
            'evaluation (especially for "label_correct").'
        )
    )
```

```
Your task is to evaluate a natural-language claim across two criteria.
You will be given a gold context composed of a question, its answer,
the domain, and optional external knowledge. Treat the gold context as
the authoritative ground truth.

You will be given a claim labeled as either ENTAILED (supported by the
gold context) or CONTRADICTED (refuted by the gold context). Using this
gold information, assess whether the claim is correctly labeled, and
whether it is free of meta-references. More detailed instructions for
the two evaluation criteria are provided along with the JSON schema
below.

Your answer should be in JSON format, adhering to the following schema:
<SCHEMA GENERATED HERE>
```

Figure 20: Judge prompt used for contradicted and entailed claims. The prompt consists of two parts: a Pydantic schema that specifies the required JSON output (label correctness, self-containment, and a short justification), and a natural-language instruction block that explains the judging task and the available gold context. The Python block defines the output schema, which is injected into the prompt at runtime in the placeholder `<SCHEMA GENERATED HERE>`.

```python
from pydantic import BaseModel, Field
from typing import Literal

class NEIClaimQuality(BaseModel):
    label_correct: Literal["yes", "no"] = Field(
        ...,
        description=(
           'Is the assigned label of the claim (ENTAILED/CONTRADICTED) '
            'correct given the gold information? Answer "yes" if the '
           'label of the claim follows from the gold information; "no" '
            'otherwise. If you are unsure, answer "no".'
        )
    )
    free_of_meta_references: Literal["yes", "no"] = Field(
        ...,
        description=(
            'Does the claim avoid meta-references to the question, '
            'answer, or prior text (e.g., "this question", '
            '"the answer above", "as mentioned earlier")? Answer '
            '"yes" if it is completely free of meta-references; '
            '"no" otherwise. References to provided external '
            'knowledge do not count as meta-references.'
        )
    )
    category_correct: Literal["yes", "no"] = Field(
        ...,
        description=(
            'Is the assigned category of the claim '
            '(OUT-OF-SCHEMA/COUNTERFACTUAL/SUBJECTIVE) correct given '
           'the gold information? Answer "yes" if the category of the '
            'claim follows from the gold information; "no" otherwise.'
        )
    )
    schema_leakage: Literal["yes", "no"] = Field(
        ...,
        description=(
            'Does the claim expose database schema details or '
            'technical artifacts? Answer "yes" if it exposes '
            'schema details (e.g., table names, column names, etc.); '
            '"no" if it does not.'
        )
    )
    reasoning: str = Field(
        ...,
        description=(
            'Brief explanation (1-2 sentences) justifying your '
            'evaluation (especially for "label_correct" and '
            '"category_correct").'
        )
    )
```

```
Your task is to evaluate a natural-language claim across several
criteria. You will be given a gold context composed of a question, its
answer, the domain, optional external knowledge, and the complete
database schema underlying the gold context. Treat the gold context as
the authoritative ground truth.

You will be given a claim with an assigned NOT ENOUGH INFO (NEI) label,
meaning that its truth cannot be determined from the gold context, even
with full access to the database. The claim is also assigned an NEI
category: OUT-OF-SCHEMA (depends on information not stored in the
database), SUBJECTIVE (expresses opinions or judgments), or
COUNTERFACTUAL (describes hypothetical scenarios). Using the gold
context, assess whether the NEI label and category are correct, whether
the claim is free of meta-references, and whether it leaks schema
details of the database. More detailed descriptions for each evaluation
criterion are provided in the JSON schema below.

Your answer should be in JSON format, adhering to the following schema:
<SCHEMA GENERATED HERE>
```

Figure 21: Judge prompt used for NEI claims. Compared to the prompt for entailed and contradicted claims, this prompt includes additional evaluation criteria specific to NEI cases. In addition to label correctness and self-containment, judges assess whether the assigned NEI category is correct and whether the claim leaks database schema details. The instruction block also gives judges access to the full database schema. The Python block defines the output schema, which is injected into the prompt at runtime in the placeholder <SCHEMA GENERATED HERE>.

```
You are a fact-checking assistant operating over structured data. You
will be given a natural-language claim and optional external
information. You will have access to a SQLite database and may execute
arbitrary SQL queries over it using specialized tools.

Your task is to determine whether the claim is "ENTAILED",
"CONTRADICTED", or "NOT ENOUGH INFO" based on evidence you obtain from
the database. The labels are defined as follows:

- ENTAILED: The claim is supported by the database.
- CONTRADICTED: The claim is refuted by the database.
- NOT ENOUGH INFO: The database does not provide sufficient evidence to
decide.

Use the available tools to query the database and gather evidence
before making a decision. Do not ask the user for clarification or
additional information.

You should always start by querying the database for the schema (tables
and columns).

Your answer should be in JSON format, adhering to the following schema:
{
  "properties": {
    "verdict": {
      "description": "Whether the claim is supported, contradicted, or
      undecidable from the database.",
      "enum": [ "ENTAILED", "CONTRADICTED", "NOT ENOUGH INFO"],
      "title": "Verdict",
      "type": "string"
    },
    "justification": {
      "description": "Brief justification (1-2 sentences) of the
      verdict.",
      "title": "Justification",
      "type": "string"
    }
  },
  "required": ["verdict", "justification"],
  "title": "ClaimVerdict",
  "type": "object"
}


Output Example 1:
{
    "verdict": "ENTAILED",
    "justification": "The database shows that the population of France
    is 67 million, which supports the claim."
}

Output Example 2:
{
    "verdict": "CONTRADICTED",
    "justification": "The database indicates that the capital of
    Germany is Berlin, contradicting the claim."
}

Output Example 3:
{
    "verdict": "NOT ENOUGH INFO",
    "justification": "The database does not contain any information
    about the population of Sacramento."
}
```

Figure 22: Prompt for all verifier models in the experiments (Section 6.1). It has been carefully constructed with many iterations and refinements across the different model families. We have tried our best to make the models perform as good as possible. For example, after noticing some models deciding *not* to use tools and hallucinate an answer, we including instructions like *"Use the available tools to query the database and gather evidence before making a decision"* and *"You should always start by querying the database for the schema"*. Furthermore, for a discussion on output format see Appendix B.2. Notably, for the observant readers, there is no explicit explanation of the *tool* the agent has in its disposal because these descriptions are provided by the frameworks internally (most are built this way) by injecting the docstrings and input-output schemas of the tools in the prompt using templates.

Domain	Subdomain	Databases
Entertainment	Movies	movie, movie_3, movie_platform, movies, movies_4, movielens
	Music	music, music_tracker, music_platform_2
	TV Shows	law_episode
	Games	video_games, card_games
	Cartoons	simpson_episodes, superhero, disney
Technology	Software	talkingdata, codebase_community, codebase_comments, social_media, software_company
	IT	public_review_platform, app_store
	Blockchain	coinmarketcap
	Vision	image_and_language
Education	University	student_club, university, cs_semester, college_completion, computer_student
	Academia	authors, citeseer, book_publishing_company
	Schools	california_schools
	Language	language_corpus
	Books	books, shakespeare
Health	Healthcare	synthea, donor, mental_health_survey
	Medical	thrombosis_prediction
	Biology	genes
	Chemistry	toxicology
Economy	Finance	student_loan, debit_card_specializing
	World Economies	world_development_indicators
	Retail	retail_complains, sales, superstore, car_retails, regional_sales, retails, retail_world, works_cycles
	Banking	financial
Transportation	Transit Systems	bike_share_1, shipping, trains, cars
	Airport	airline
Gastronomy	Food	food_inspection_2, beer_factory, food_inspection, cookbook, craftbeer
	Restaurant	restaurant, menu
Governance	Crime	chicago_crime, shooting
	Law	legislator
Environment	Weather	sales_in_weather
	Geography	mondial_geo, address, world
Labor	Human Resources	human_resources
Sports	Basketball	professional_basketball
	Olympics	olympics
	Hockey	ice_hockey_draft, hockey
	F1	formula_1
	Soccer	european_football_1, european_football_2, soccer_2016

Table 4: Domain taxonomy with representative databases (80 total across 11 domains and 36 subdomains).