COMPILED TOKENIZERS: TRAINING LANGUAGE MODELS ON SCHEMA-SPECIFIC ACTION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Constrained decoding is commonly used to make language models generate valid structured output, such as SQL, by masking inadmissible tokens at inference time. We introduce Type-Constrained Tokenization (TCT), which compiles the database schema into the tokenizer and training target. The resulting representation lets the model select tables and columns from a candidate list that reflects the current query scope, while the decoder supplies decisions with only one admissible option. Since invalid schema references have no encoding, the model learns to choose among the references and structural alternatives that the schema permits. We compare the models on the 87% of test questions that pass the compiler's encoding and rendering checks. On this set, a 1B compiled model produces 98% fewer outputs that fail to execute than the same model fine-tuned on plain SQL, without reducing execution accuracy. On the full test set, counting the 11% of questions outside the grammar as compiled failures still cuts non-executions by two fifths, while the accuracy difference is not significant. A text control trained to write the same schema choices in ordinary text answers 8% more questions correctly than the compiled model, although 5% of its outputs fail to execute, compared with less than 1% for the compiled model. Given the question, schema, candidate SQL, and a preview of each distinct executable result, a separate 8B instruction model, used without further training, chooses among outputs from models trained to emit plain SQL, schema choices as text, or compiled actions. It answers 10% more questions than the best single model. The same judge answers fewer questions with three rehearsal seeds of either text model, and its accuracy degrades when the compiled candidate is removed.

1 INTRODUCTION

Language models are often asked to produce valid structured output: SQL for a database, tool calls for an API, or configuration files that match a schema. The standard remedy is *constrained decoding*, which masks tokens that the grammar does not admit (Willard & Louf, 2023; Dong et al., 2025; Ugare et al., 2025; Poesia et al., 2022). A syntax-only grammar does not check whether tables and columns exist or are in scope, but constrained decoders can add these checks. PICARD validates names and scope (Scholak et al., 2021), and Synchromesh enforces semantic constraints (Poesia et al., 2022). Masking alone does not guarantee completion within a finite decoding budget. It can also change the sampled distribution (Park et al., 2024). For a model trained on SQL text, inference-time constraints leave the training representation unchanged. We therefore study how to incorporate the schema into the tokenizer and training target. Our compiler, TCT (Type-Constrained Tokenization), derives an action vocabulary from the database catalog, which specifies its tables, columns, and keys. Within this vocabulary, the model selects tables from the catalog and columns from the tables in scope instead of spelling their names. We call the list of allowed tables or columns at each reference decision a *menu*. When only one action is allowed, the decoder supplies it instead of asking the model to predict it. A compiled transition function determines the admissible actions at each position, making references outside the catalog unrepresentable. Decoding still restricts scores to the admissible set, and we measure how often the trained model requires this mask (Sections 3 and D).

To evaluate this representation, we employ Spider 1.0 (Yu et al., 2018) for controlled comparisons, without claiming competitiveness with specialized text-to-SQL systems (Section 4). From one 1B base model we train a *text baseline* that emits SQL, the *compiled model*, and a *menu-text control*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that writes menu indices and identifier names as ordinary text (Section 5.4). The held-out test set has 2,147 questions from 40 unseen databases. The grammar supports 1,908 questions, reduced to 1,863 by a later render check. Our main result is the reduction in execution failures (Section 5.1). Counting the 239 unsupported questions as compiled failures, the compiled model has 245 execution failures across all 2,147 test questions, compared with 405 for plain SQL. The compiled model answers 12 more questions correctly, but the difference is not significant (Section A). We make four contributions.

- **A compiled training representation.** Scope tracking, omission of forced decisions, and native literal spans share one transition function (Section 3). The guarantee covers references inside accepted streams (Proposition 1).
- **An accuracy–execution tradeoff.** Menu text is more accurate than compiled decoding, which fails to execute less often. Keeping the menu-text answer when it executes and using the compiled answer otherwise gives 1,273 correct answers and one execution failure on the common 1,863-question set. We tried this combination after the test evaluations, using answers the models had already generated (Table 1). Compilation also produces more answers that execute but are wrong (Section 5.1). We did not observe the 1B accuracy lead over plain SQL in one 8B development comparison under our main scorer (Section 6).
- **Verifier selection across representations.** An 8B judge selects among the three executed answers and solves 1,365 questions, gaining 92 over menu text with compiled output as a fallback. The same judge solves fewer questions with three rehearsal seeds of either text representation or nine plain-SQL candidates (Section 5.2 and Figure 2).
- **Attribution, confidence, and robustness.** Compiled confidence better distinguishes correct from incorrect answers than plain-SQL confidence (Section 5.2). A text control using the same menu indices shows that the accuracy gain does not require compiled actions (Section 5.4). Compared with plain SQL, compiled decoding has higher accuracy when schema names change, no detectable difference across the pooled Dr.Spider paraphrase sets, and lower accuracy when database content changes (Section 5.6).

2 Related Work

Decode-time engines constrain an existing vocabulary (Willard & Louf, 2023; Moskal et al., 2025; Dong et al., 2025; Zheng et al., 2024; Ugare et al., 2025; Koo et al., 2024; Park et al., 2025). JSONSchemaBench benchmarks structured generation (Geng et al., 2025). Masking can alter the sampled distribution (Park et al., 2024; Li et al., 2026b), while restrictive formats and tokenization affect task performance (Tam et al., 2024; Banerjee et al., 2025; Beurer-Kellner et al., 2024; Li et al., 2026a). PICARD checks catalog names, aliases, and scope (Scholak et al., 2021), and Synchromesh uses completion engines for syntax, scope, and typing (Poesia et al., 2022). A related line of work changes the training representation itself. TRANX predicts actions that construct an abstract syntax tree (Yin & Neubig, 2018), while GrammarCoder adds grammar-rule tokens to a pretrained vocabulary (Liang et al., 2025). Building on these approaches, TCT combines catalog-derived menus, scope tracking, omission of forced decisions, and native literals in one tokenizer. Our contribution is this combination, rather than semantic constraints or grammar-action training alone. Schema linking connects questions to catalog elements (Wang et al., 2020). IRNet also trains an intermediate representation (Guo et al., 2019). BRIDGE uses FROM-first decoding and masks column pointers by selected tables (Lin et al., 2020). Selecting elements by index builds on pointer networks (Vinyals et al., 2015). Our menus do not link question values to database contents. Spider evaluates cross-domain text-to-SQL (Yu et al., 2018; Zhong et al., 2020). BIRD emphasizes database values, and Spider 2.0 enterprise workflows (Li et al., 2023; Lei et al., 2025). Annotation errors have been documented in BIRD and Spider 2.0-Snow (Jin et al., 2026). We test Dr.Spider, Spider-Syn, and Spider-Realistic perturbations (Chang et al., 2023; Gan et al., 2021; Deng et al., 2021). Other methods run candidate programs and use their outputs to choose an answer. MBR-Exec groups programs by execution result (Shi et al., 2022), CodeT uses agreement with generated tests (Chen et al., 2023), and LEVER trains a verifier on programs and results (Ni et al., 2023). Self-consistency improves accuracy by aggregating sampled reasoning paths (Wang et al., 2023), while repeated sampling can increase the chance of finding a correct candidate (Brown et al., 2024). LLMs can also judge model responses (Zheng et al., 2023). Here we apply a judge to executed SQL candidates to test whether different representations provide more useful candidates than seeds or samples of one representation.

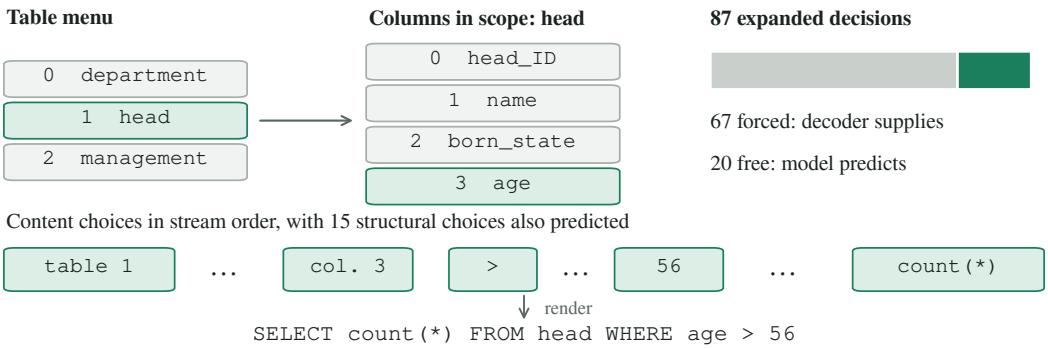


Figure 1: **From scoped choices to SQL.** Table index 1 selects `head`. At the column reference, the decoder supplies the table choice because only `head` is in scope, and the model selects column index 3 (`age`). Boxes label choices, not token ids. The stream is FROM-first, and the renderer restores SQL keyword order. Five content choices are shown from the 20 model decisions. The decoder supplies the other 67 fixed transitions (Table 3).

Our comparison of output representations makes no competitiveness claim against small-model text-to-SQL systems built on synthetic data (Li et al., 2025; Sheng & Xu, 2025).

3 Compiling Schemas into Tokenizers

TCT compiles a fixed database catalog into an action alphabet that represents schema references as bounded menu indices (Figure 1). The compiled alphabet contains 321 ids: 64 for structural decisions, a shared pool of 256 payload ids, and one reserved padding id that is never admissible or supervised. The meaning of a payload id depends on the transition state: it can represent a literal byte, a bounded integer, or a menu index (Section C). At a table-selection site, the menu contains the catalog's tables. At a column-selection site, the available columns are restricted to the tables already introduced into the current query scope. To select a column, the model first chooses a table already in scope and then a column in that table. Our trained model supports at most 256 tables and 256 columns per table. An alternative numbers all in-scope columns in one list. We tested it only on large artificial database schemas and did not train a model to use it (Section C). The indices refer to positions in each catalog's menus, so the same token layout serves all databases, including the 40 unseen test catalogs. Compilation rejects catalogs that exceed its capacity with a typed error. Decoding rejects indices outside the current menu, making references outside the catalog unrepresentable.

When the next action is fixed, the decoder supplies it. The model predicts only choices with more than one allowed action. An earlier census using an 833-id codec found 660,901 of 881,626 expanded positions (74.96%) forced across 6,108 supported training rows and 924 supported development rows. This census predates the 321-id factored representation and native literal spans used by the trained models (Section C). Although omitting fixed actions shortens the compiled stream, the remaining decisions use a different representation from SQL text. It therefore does not establish fewer model predictions per query than the text baseline, which uses the base model's native bpe-compressed tokenizer (Section B). Since menu indices depend on catalog structure rather than identifier spellings, renaming tables and columns without changing that structure leaves the target stream unchanged. The new names still appear in the model's prompt and may lead it to choose different indices (Section 5.5).

Training and decoding. The grammar marks where a string value begins, letting the model write it with its pretrained tokenizer in a *native literal span* that ends with a compiled token. Integers use compiled digit tokens instead, but the model must choose the right value in either case (Section 5.6). We extend pretrained MiniCPM5-1B with 320 action tokens and three control tokens, adding a row for each new token to both its input embeddings and output layer. Where a new token has an assigned text spelling, we tokenize it and average the corresponding pretrained vectors to initialize both new rows. Otherwise, we use the mean pretrained vector with small noise. We then fine-tune all model parameters (Section B). At compiled decisions, the decoder masks out actions forbidden by the

grammar, a safeguard that still prevents errors after training (Section D). Most pretrained tokens remain available while writing a string, and a separate stop token can end an incomplete output. The deployed system also narrows some decisions. When generating a number in a comparison such as `age > 30`, it allows only numbers mentioned in the question, but leaves the number after LIMIT unrestricted. It also uses foreign keys or shared column names to narrow join choices. When either rule cannot narrow the choices, decoding uses the original menu, although an applicable rule may still exclude the benchmark's reference SQL query (Section B). The supported SQL subset includes single-table and joined SELECT, filtering, grouping, ordering, aggregation, DISTINCT, and uncorrelated subqueries. A table can enter each scope only once, excluding self-joins, and set operations would need a second query root. Supporting either feature would require changes to the encoded output and new training and evaluation. Questions whose reference SQL uses unsupported features are omitted from comparisons restricted to supported questions, but count as compiled failures in results for the full question set (Sections A and 4). The compiler also accepts JSON Schemas and Go grammars, but the trained-model evaluations in this paper are limited to SQL.

Proposition 1 (In-menu references in accepted streams)**.** *For a catalog $\mathcal{C}$, every reference decision accepted by the compiled transition function selects a menu entry permitted by the current scope. A complete accepted stream therefore contains no table or column reference outside its menu. This invariant concerns internal references, not the SQL text emitted by the renderer.*

This does not guarantee completion, faithful rendering, dialect compatibility, execution, or semantic correctness. Defects in the evaluated renderer included misquoted keyword identifiers and NUL literals rejected by SQLite (Section C).

4 Experimental Design

Spider 1.0 separates 140 training databases from 20 development databases containing 1,034 questions (Yu et al., 2018). We fine-tune on the training databases. We used reference SQL from both splits to choose keyword spellings for 36 structural action tokens and initialized their input and output rows from the corresponding pretrained vectors. Training queries alone give the same spellings, and no test queries entered this initialization (Section B). For each result, we report the question set, checkpoint, scoring rule, and whether the comparison was planned before evaluation or assembled afterward from saved predictions. We count a generated query as correct when it returns the same result as the reference query on one database, although this does not prove the queries have the same meaning (Zhong et al., 2020). Coverage matters because the compiler does not accept every Spider query. Development results retain the original 914-question subset because the ten additional questions later found to be encodable were not reevaluated. An earlier matched-budget checkpoint pair, distinct from the deployed models, answered 580 of all 1,034 development questions correctly with plain SQL and 551 with compiled SQL, counting exclusions as compiled failures. On the held-out test set, the grammar supports 1,908 of 2,147 questions (88.9%). A later check excluded 45 queries whose rendered SQL differed under a canonical-AST comparison, leaving 1,863 for the three-model comparison (Section A). The compiled model leads plain SQL by 48 correct answers on the original 1,908 supported test questions, but by only 12 across all 2,147. Counting the 239 unsupported test questions as compiled failures gives 245 non-executions for the compiled model and 405 for plain SQL. Choosing compiled output where the reference SQL fits the grammar and plain SQL elsewhere leaves 74 non-executions. This is only a hindsight calculation: a new question has no reference SQL to guide that choice.

Models and representations. The three output models start from this same augmented MiniCPM5-1B checkpoint and use the same prompt format, three training stages (schema literacy, Spider, and rehearsal), and number of optimizer steps. At stage two, plain SQL trained on 6,106 examples, which is 771 more than the 5,335 used by compiled output and menu text. The smaller set resulted from setting aside examples for a database-disjoint holdout, and we did not retrain plain SQL on it. The formats also differ in output length and in which token positions are trained, so this is not a comparison that isolates the effect of the output format (Section B).

The text baseline. We fine-tune the base model to emit SQL as ordinary text.
The compiled model (TCT). We fine-tune on the compiled token stream (Section 3) and decode under its action space.

Table 1: **Accuracy and execution on the common held-out basis.** All rows use the same 1,863 questions and saved predictions, scored by the same procedure. Bars show correct answers (green, left), wrong but executing answers (amber, middle), and execution failures (red, right), on a common 0–100% scale. Arrows denote fallback on execution failure. Compiled decoding uses both narrowing channels. These counts were assembled after evaluation from single checkpoints. The judge test was specified beforehand (Section 5.2). The 45 exclusions from the primary 1,908-question set contain 13 text failures and no compiled failures.

System	Correct	Did not run	Outcome distribution
Plain SQL	1,108	324	
Menu-index text	1,246	93	
Compiled variant	1,153	6	
Plain SQL → compiled	1,232	1	
Menu text → compiled	1,273	1	
Judge over the three arms	1,365	1	

The menu-index text control. We fine-tune on menu indices and scope structure written as ordinary text, including each reference's index and identifier name, and decode freely (Section 5.4).
The verifier. Without further training, Qwen3-8B selects among distinct executable answers from the plain-SQL, compiled, and menu-text models (Figure 2 and Section E). It reads the question, schema, candidate SQL, and result previews, but not the reference query or result, model identity, or confidence scores.

We render each output as SQL, run it on the question's database, and record whether it executes and matches the reference query's result. Compiled and menu-text outputs use different renderers, but all arms share this scoring procedure.

Evaluation and checkpoint selection. Except for the sampling experiments in Sections E and 5.2, we use greedy decoding: the model chooses the highest-scoring allowed token at each step. Before the first held-out evaluation, we fixed the deployed system and the original set of 1,908 supported questions from 40 unseen databases. Questions from the same database with the same reference query are closely related, so we grouped them into 1,038 clusters for the main accuracy test. The two-sided sign-flip test repeatedly reverses each cluster's net advantage between the models and asks how often a gap at least as large in either direction appears. We report exact McNemar results alongside it but use the cluster test for the main decision (Section B). After the rendering check, we also formed a separate 1,863-question comparison from saved predictions of the three models. We specified what would count as success or failure before each held-out evaluation and report both kinds of outcome. The test set was used in seven completed evaluations, and one planned 8B test was withdrawn. The compiled 1B system was decoded on test only in the first evaluation, while later analyses reused its saved predictions. We chose model checkpoints and decoding settings without using test results (Section B). We did not measure latency or FLOPs and make no cost or efficiency claim. To compare against spending more decoding effort on text alone, we include baselines that sample several text queries and vote among those that execute (Section 5.2).

5 Results

5.1 Execution failures and wrong answers

We first compare how often the generated SQL executes. On the held-out test split, the compiled variant executes on 1,902 of the 1,908 supported questions (99.69%), compared with 1,571 (82.34%) for the plain-SQL baseline (Table 2 in Section A). Only compiled output executes on 336 questions, while only plain SQL executes on five. The compiled variant leads on 38 of 40 databases. To test whether this gap could arise with no model advantage, we randomly reverse the model labels for each of the 1,038 reference-query clusters in 200,000 draws. No draw gives a gap as large in either direction. The standard add-one correction counts this as one exceedance in 200,001 trials, giving an estimated two-sided p-value of about 5.0×10^{-6}, the smallest value this test can report. These figures exclude unsupported questions. Counting them as compiled failures changes the disagreement to 336 against 176, with 28 databases in favor, 9 against, and 3 tied: text executes on 171 of the 239

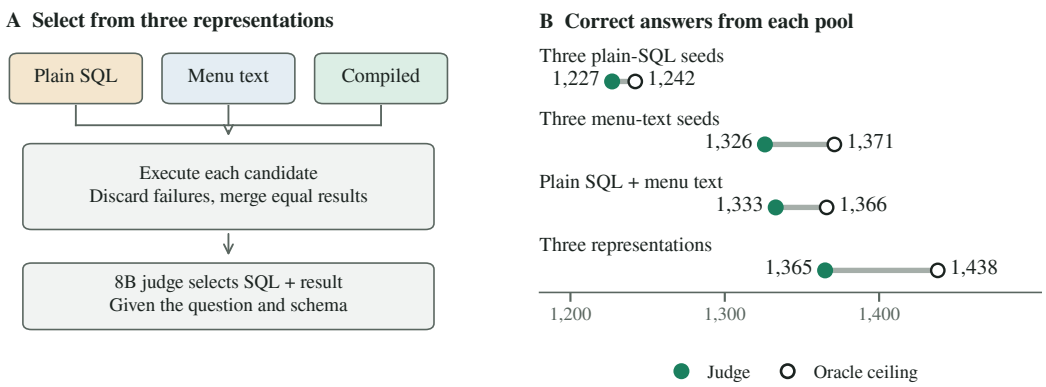


Figure 2: **Execution-based verification.** The judge selects among distinct executing results (left). Three representations provide more correct candidates than three rehearsal seeds of either text representation (right). Filled markers count correct selections. Hollow markers count questions with any correct candidate. All pools use the same 1,863 questions and judge. The two-text pool omits compiled output (Section 5.2).

excluded questions. Across all 2,147 test questions, the compiled model has 245 execution failures against 405 for plain SQL, while its 99.69% execution rate applies only to the supported subset. We have not tested a way to choose among models on all test questions without knowing the reference query. The fallback rules in Table 1 cover only questions supported by all arms. Among answers that execute on the 1,908 supported test questions, plain SQL is correct 71.0% of the time and the compiled variant 61.2% (Table 2). The outcome bars in Table 1 show the same tradeoff: compilation removes most execution failures, but many answers remain wrong. Of the compiled variant's 739 incorrect scored predictions on the supported test set, 344 name a different set of tables from the reference query (Section A).

The grammar mask does matter: it removes actions that the current query cannot accept. When we stop filtering those actions but keep tracking the grammar, the model chooses a forbidden action on 49 of 914 development questions. With the two extra rules disabled in both runs, execution failures rise from 7 with the grammar mask to 53 without it. Under teacher forcing, the mask changes the greedy choice at only 0.2% of compiled decisions, so the model usually chooses a permitted action on its own, but the mask still catches errors (Section D). On the development set, plain SQL fails to execute on 18.1% of questions, menu text on 5.4%, and compiled output on 0.9% (Table 2). Of menu text's 49 failures, 26 contain a qualified name, such as `table.column`, that the schema cannot resolve. The compiled decoder rules out this kind of reference in accepted token streams by allowing only entries from the current menu. The original held-out test measured whether answers were correct. We calculated execution-failure rates afterward from the same saved predictions and report them with the grammar's coverage.

5.2 Confidence and execution-based fallback

We ask whether the models assign higher confidence to correct answers than to wrong ones. For compiled output, we average the log-probabilities of the actions the model chose at its decision sites. Five of the 1,908 supported held-out test questions have no compiled confidence score. On the remaining 1,903, this score has an AUROC of 0.838, against 0.712 for plain text. The database-clustered interval for the compiled variant is $[0.807, 0.866]$, and the paired difference is 0.126, $[0.092, 0.163]$. These scores are not directly comparable: the text score covers the whole output, while the compiled score covers only decision sites. We therefore also score each model's entire output and compare answers within groups of similar output length. On the 1,858 test questions shared by all three arms after excluding those same five, this gives AUROCs of 0.795 for the compiled variant, 0.733 for menu text, and 0.707 for plain text. The compiled decision-site score also does better than this alternative score for the same model. The menu-text scores were obtained by teacher forcing, and each arm contributes one checkpoint seed (Section E). On the 911 development questions with scores for all three models, every confidence interval for the length-controlled pairwise AUROC

differences includes zero, so these data do not establish which model ranks answers better. A check planned before evaluation restricts the comparison to 1,744 questions whose re-encoded outputs have unchanged length. There the compiled-minus-menu-text difference is $+0.046$, with an interval of $[-0.0002, 0.0932]$. We cannot explain why this interval includes zero while the main three-arm comparison favors compiled output. AUROC measures discrimination between correct and incorrect answers, rather than calibration of their probabilities. Moreover, the models generate and score different answers, so this comparison does not establish a causal effect of compilation. Among the 20% most confident answers on those 911 development questions, compiled output is correct 97.3% of the time and plain text 84.1% of the time. We did not fix an abstention threshold for held-out evaluation, so this development result does not establish a rule for deciding when a 1B model should abstain. Section E reports the base rates, scores for entire outputs, a run without the extra narrowing rules, and the restricted comparisons.

We use plain SQL with compiled output as a fallback: if the plain-SQL query fails to execute, we take the compiled query. This rule needs no trained selector. For comparison, vote@K samples K plain-SQL answers, discards those that fail to execute, and selects the most common result (Section E). On the 1,863 held-out questions shared by all three representations, plain SQL with compiled output as a fallback answers 1,232 questions, compared with 1,155 for vote@4 and 1,181 for vote@8. Menu text with compiled output as a fallback answers 1,273 (Table 1 and Section E).

Verifier selection. The fallback that starts with menu text uses its answer whenever it executes, even if another model returns a different result. The verifier instead runs all three answers, discards those that fail, and asks the judge to choose when different executable results remain. It calls the judge on 679 of the 1,863 held-out questions. On the others, there is only one distinct executable result or no query executes. At least one model gives a correct answer on 1,438 questions, and the verifier answers 1,365 correctly (Table 1). Compared with the menu-text fallback's 1,273 correct answers, the verifier corrects 129 errors but makes 37 new ones, a net gain of 92 (exact McNemar $p = 3.9 \times 10^{-13}$). The test fixed before evaluation, which groups questions sharing a reference query, also finds a difference ($p = 1.0 \times 10^{-4}$). The verifier answers 119 more questions than menu text alone. Both gains exceed the thresholds set before evaluation: 40 additional correct answers over the fallback and 50 over menu text. To see how much the judge helps without mixing representations, we also give it nine answers from the plain-SQL model: one greedy answer and the eight samples used by vote@8. On the same 1,863 questions, it selects a correct answer 1,232 times, compared with 1,181 for vote@8. Menu text alone answers 1,246, so the three-representation result depends in part on which models supply the candidates. We did not run the corresponding judge over repeated menu-text samples (Section E). We next ask whether using three representations helps more than using three versions of one representation trained with different rehearsal seeds. With the same judge and questions, three plain-SQL models give 1,227 correct answers and three menu-text models give 1,326, against 1,365 for the plain-SQL, menu-text, and compiled models together (Figure 2). The gains of 138 and 39 meet the thresholds set before testing (exact McNemar $p = 8 \times 10^{-18}$ and $p = 0.005$, with cluster tests in Section E). With the compiled candidate removed, the judge answers 1,333 questions, 32 fewer than with all three representations ($p = 0.002$). This gain also meets its threshold, although the corresponding gain of 17 on the development set does not. The compiled model thus contributes useful candidates in this held-out comparison. These results do not show that compiled output makes the judge better at choosing among candidates (Section E). The judge has database access and is an 8B model selecting among outputs from 1B models, so these results establish no cost or efficiency advantage. On this held-out split, Qwen3-8B answers 1,365 questions using one prompt and one fixed shuffle of the candidates. Given the same candidates, Qwen2.5-Coder-7B-Instruct answers 1,344. It still beats the fallback's 1,273, but the 21-answer difference shows that the size of the gain depends on the judge (Section E).

5.3 HELD-OUT ACCURACY

On the 1,908 held-out questions supported by the grammar, the deployed compiled model answers 1,164 (61.01%) correctly, compared with 1,116 (58.49%) for plain SQL, a lead of 48 questions. Spider's official scorer gives the same lead. The difference narrowly meets the statistical criterion chosen before evaluation, but not when we treat databases as independent units or use the 1,863-question set retained after the render check. We therefore cannot establish an accuracy advantage across new schemas from this result. The lead is notable because compiled output used 771 fewer

stage-two examples than plain SQL. It does not establish a data-efficiency gain or isolate the output format's effect: both models took the same number of optimizer steps, their training targets differ, and the compiled checkpoint was selected from five rehearsal runs, while the plain-SQL anchor was fixed beforehand (Section B). The tests and sensitivity checks are detailed in Section B.

5.4 The role of representation and tokenization

To see whether the deployed model's accuracy gain over plain SQL requires compiled actions, we compare it with the menu-text control. The control writes the same scope-tracked choices as ordinary text, including an index and a name for each reference, and decodes without a compiled grammar mask. It shares the compiled model's stage-two and stage-three training examples (Sections B and 4). On the development set, it answers 618 of 914 questions, against 599 for the compiled variant and 551 for plain SQL (Table 2). Its 19-question lead over the compiled variant is not significant ($p = 0.153$), while its 67-question lead over plain SQL is ($p = 9.4 \times 10^{-7}$), despite plain SQL having more second-stage training examples. Menu text thus reproduces the accuracy gain without compiled actions, but this comparison does not show that compiled actions lower accuracy. Under the decision rule set before evaluation, this result does not support an accuracy benefit from compiled actions. The menu-text model uses that format on 905 of 914 development questions, while the other nine outputs fall back to plain SQL. This exceeds the 95% format requirement and shows that the accuracy result is not mainly due to plain-SQL fallback. This is a one-seed result for 1B models. Removing the index prefixes gives 600 of 914, an 18-question drop that is not significant under the planned query-cluster test ($p = 0.208$). Even without indices, the control differs from plain SQL in output format, second-stage training data, and seed sequence, so its remaining lead cannot be attributed to indices alone (Section B).

5.5 Constrained-decoding baseline

We also tested a grammar mask that tracks catalog scope on an earlier plain-SQL model. On 685 supported development questions, it leaves accuracy nearly unchanged on clean schemas (62.19% versus 62.04%, exact $p = 1.0$), and 31 masked outputs do not finish. The mask also fails to preserve validity when schema names change. Because this grammar covers fewer SQL forms, uses an earlier checkpoint, and was not tested on the held-out split, these results are not a matched comparison with TCT or a general verdict on constrained decoding. The full results are in Section D.

5.6 Robustness to schema and content changes

Menu indices keep table and column names out of the model's output. Renaming those identifiers leaves the indices unchanged, although the new names still appear in the prompt and may change which indices the model chooses. Figure 3 compares the compiled model with plain SQL on Dr.Spider changes to schema names, question wording, and database content. These tests use the development-selected compiled checkpoint and the fixed plain-SQL anchor (Section B). These Dr.Spider suites are separate perturbation tests, not the Spider held-out test split. The figure shows accuracy on the changed questions, not the drop from clean accuracy. The compiled model leads on both schema-name sets but trails on the content-equivalence set, which changes database contents (exact query-cluster $p < 5 \times 10^{-5}$ in all three comparisons). Across nine paraphrase sets, the test detects no difference ($p = 0.757$ over 901 clusters, with four sets favoring compiled). These comparisons cannot tell whether the schema-name gains come from compiled tokens or from menu-based choices (Section D).

Leading on changed questions does not by itself show that a model handles the change better: it may have started ahead on the original questions. We therefore compare how much each arm's accuracy changes between the original and altered versions. These suites alter questions from the Spider development set. This later test, fixed before evaluation, uses Spider-Syn and Spider-Realistic with the same compiled and plain-SQL checkpoints plus the single trained menu-text checkpoint (Gan et al., 2021; Deng et al., 2021). Neither suite establishes a compiled advantage. Across 749 Spider-Syn pairs, the compiled model's change differs by -10 correct answers from plain SQL ($p = 0.57$) and by $+30$ from menu text ($p = 0.051$). Neither difference is significant under the planned row-level sign-flip test, which does not mean the models respond equally to paraphrases. On 305 Spider-Realistic pairs, the compiled variant loses 50 correct answers after rewriting, against 26 for plain SQL. Its additional loss of 24 ($p = 0.0097$) passes the planned row-level test even after

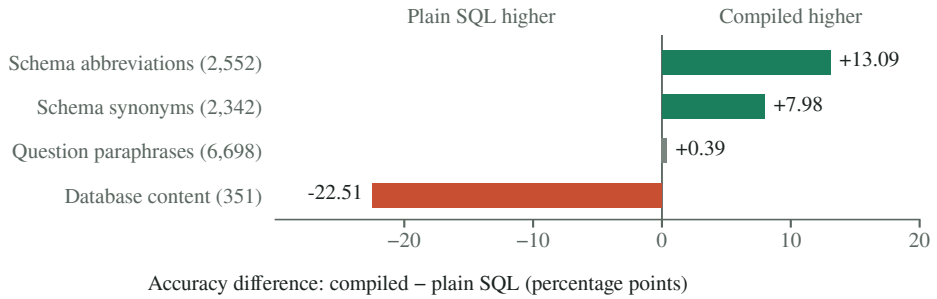


Figure 3: **Accuracy under different perturbations.** Compiled decoding leads on schema-name changes and trails on changed database content. Bars show raw accuracy differences for one fixed checkpoint pair. Parentheses give question counts. The paraphrase row pools nine sets. These are separate perturbation sets, not the Spider held-out test split. These gaps do not measure the change from clean to perturbed accuracy.

correction for four comparisons. The -18 difference from menu text is not significant ($p = 0.060$). A later check that groups questions by database does not pass the same four-test correction (Section D). The reversal under changed database content may point to a limitation of the representation. Menu indices help the model choose tables and columns, but it still chooses query values without a way to copy them from database cells (Section 2). A value-copy mechanism might help, but we have not tested one. We have also not tested this result on BIRD or Spider 2.0. None of these robustness tests was repeated across training seeds.

6 Limitations

Each new database schema must be compiled before use. The current SQL grammar does not support self-joins or set operations. We train and evaluate TCT to generate complete queries, so we make no claim about editing or completing partial programs (Section C). At 1B, menu text is more accurate than compiled output, so we cannot credit the accuracy gain to compiled tokens alone (Section 5.4). This comparison uses one way of writing menu choices and one training seed. Menu text alone answers 14 more questions than plain SQL with compiled output as a fallback, but this does not establish that menu text is more accurate (Table 1 and Section E). In a separate experiment on Spider's 914-question development set, the 8B compiled variant answers 696 questions against 719 for plain SQL under our main scorer, with one training seed per model. The direction changes with scoring: the compiled variant trails by eight under Spider's single-instance evaluator but leads by six under its test-suite metric (Section D). No 8B SQL generator was evaluated on the Spider test split, so these results establish no accuracy trend with model size.

7 Conclusion

TCT brings schema constraints into both the tokens predicted during training and the choices available during decoding. On the common 1,863-question held-out set, the 1B compiled model produces 6 queries that fail to execute, compared with 324 for plain SQL. This is a large gain within the grammar's coverage, but the model can still choose the wrong valid table, column, or value. Menu text answers more questions correctly, showing that compiled token IDs are not required for the observed 1B accuracy gain. The smaller compiled lead over plain SQL weakens under sensitivity checks and does not appear when we compare 8B models on the Spider development set under our main scorer. The models' different answers are useful when combined. An 8B verifier that selects among their executable results answers 10% more questions than the best single model, and removing the compiled candidate reduces its score. This gain requires a separate judge and does not establish an accuracy advantage for compiled tokens alone. Future work should test whether column types, question context, and database values improve query choice, measure learning efficiency across seeds and model sizes as SQL coverage grows, and use our JSON Schema and Go frontends to test whether the validity gain transfers beyond SQL.

Reproducibility Statement

The question sets and scoring rules are described in Sections A and 4, the training and checkpoint choices in Section B, and the verifier procedure in Section E. We will provide model-training and evaluation code, together with essential result data, at https://github.com/anonymous-tct-authors/tct-models/. The retained metadata records completed fine-tuning runs, but does not attest historical checkpoint hashes or the exact compiler runtime. The saved predictions reproduce the reported judgments, but they do not show that training was independently replicated.

AI Use Statement

AI assistants helped formulate hypotheses, design and review experiments, implement and test the compiler and the model-training, evaluation, and analysis code, interpret results, find related work, and draft and edit this paper. The authors reviewed and tested the AI-assisted work and take responsibility for the final text, code, results, and claims.

References

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. CRANE: Reasoning with constrained LLM generation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2836–2857. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/banerjee25a.html.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding LLMs the right way: Fast, non-invasive constrained generation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3658–3673. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/beurer-kellner24a.html.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. URL https://arxiv.org/abs/2407.21787.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Wang, Zhiguo Wang, Vittorio Castelli, Bing Xiang, and Patrick Ng. Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Wc5bmZZU9cy.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1337–1350, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.105. URL https://aclanthology.org/2021.naacl-main.105/.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems*, volume 7. MLSys, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/5c20ca4b0b20b0bd2f1d839dc605e70f-Paper-Conference.pdf.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2505–2515, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.195. URL https://aclanthology.org/2021.acl-long.195/.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025. URL https://arxiv.org/abs/2501.10868.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1444. URL https://aclanthology.org/P19-1444/.

Tengjun Jin, Yoojin Choi, Yuxuan Zhu, and Daniel Kang. Text-to-SQL benchmarks are broken: An in-depth analysis of annotation errors. In *Proceedings of the 16th Annual Conference on Innovative Data Systems Research (CIDR 2026)*, 2026. URL https://www.vldb.org/cidrdb/2026/text-to-sql-benchmarks-are-broken-an-in-depth-analysis-of-annotation-errors.html.

Terry Koo, Frederick Liu, and Luheng He. Automata-based constraints for language model decoding. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BDBdblmyzY.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida I. Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *Proceedings of the Thirteenth International Conference on Learning Representations*, pp. 28691–28735, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/46c10f6c8ea5aa6f267bcdabcb123f97-Abstract-Conference.html.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. OmniSQL: Synthesizing high-quality text-to-SQL data at scale. *Proceedings of the VLDB Endowment*, 18(11):4695–4709, 2025. doi: 10.14778/3749646.3749723. URL https://www.vldb.org/pvldb/vol18/p4695-li.pdf.

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 C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 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*, volume 36, pp. 42330–42357. Curran Associates, Inc., 2023. doi: 10.52202/075280-1835. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Yinxi Li, Yuntian Deng, and Pengyu Nie. TokDrift: When LLM speaks in subwords but code speaks in grammar. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47605–47629, San Diego, California, United States, July 2026a. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.2199. URL https://aclanthology.org/2026.acl-long.2199/.

Yongmin Li, Jia Li, Ge Li, and Zhi Jin. AdapTrack: Constrained decoding without distorting LLM's output intent. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, pp. 529–541. Association for Computing Machinery, 2026b. doi: 10.1145/3744916.3773144. URL https://doi.org/10.1145/3744916.3773144.

Qingyuan Liang, Zhao Zhang, Zeyu Sun, Zheng Lin, Qi Luo, Yueyi Xiao, Yizhou Chen, Yuqun Zhang, Haotian Zhang, Lu Zhang, Bin Chen, and Yingfei Xiong. Grammar-based code representation: Is it a worthy pursuit for LLMs? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15640–15653, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.807. URL https://aclanthology.org/2025.findings-acl.807/.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4870–4888, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.438. URL https://aclanthology.org/2020.findings-emnlp.438/.

Michał Moskal, Harsha Nori, Hudson Cooper, and Loc Huynh. LLGuidance: Making structured outputs go brrr. Guidance AI project technical article, June 2025. URL https://guidance-ai.github.io/llguidance/llg-go-brrr.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I. Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26106–26128. PMLR, 2023. URL https://proceedings.mlr.press/v202/ni23b.html.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24547–24568. Curran Associates, Inc., 2024. doi: 10.52202/079017-0774. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2bdc2267c3d7d01523e2e17ac0a754f3-Paper-Conference.pdf.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48262–48275. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/park25l.html.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KmtVD97J43e.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Lei Sheng and Shuai-Shuai Xu. SLM-SQL: An exploration of small language models for text-to-SQL. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1497–1512, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-ijcnlp.92. URL https://aclanthology.org/2025.findings-ijcnlp.92/.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. Natural language to code translation with execution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3533–3546, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.231. URL https://aclanthology.org/2022.emnlp-main.231/.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? A study on the impact of format restrictions on performance of large language models. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1218–1236, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.91. URL https://aclanthology.org/2024.emnlp-industry.91/.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. SynCode: LLM generation with grammar augmentation. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=HiUZtgAPoH.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pp. 2692–2700. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7567–7578, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.677. URL https://aclanthology.org/2020.acl-main.677/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023. URL https://arxiv.org/abs/2307.09702.

Pengcheng Yin and Graham Neubig. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In Eduardo Blanco and Wei Lu (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 7–12, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2002. URL https://aclanthology.org/D18-2002/.

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. 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*, pp. 3911–3921, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

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. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 396–411, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.29. URL https://aclanthology.org/2020.emnlp-main.29/.

A Coverage and scoring

Table 2 retains the original development and held-out comparisons. These differ from the common 1,863-question comparison in Table 1, which also includes menu text. Both tables use the recorded execution judgments of our harness. Their counts alone establish neither an official test-suite result nor statistical significance.

Table 2: **Development and the original held-out evaluation.** Each outcome row sums to its denominator. Menu text uses the later 1,863-question test set. The compiled checkpoint was selected from five seeds with narrowing channels off. The earlier matched-budget comparison in Section B uses different checkpoints. The plain-SQL baseline used 771 more second-stage training examples.

	Supported development, of 914			Held out, of 1,908		
Arm	Correct	Ran, wrong	Did not run	Correct	Ran, wrong	Did not run
Text baseline	551	198	165	1,116	455	337
Menu-index text control	618	247	49	1,863-row basis only, Table 1		
Compiled variant, channels off	587	320	7	1,150	752	6
Compiled variant, channels on	599	307	8	1,164	738	6

Of the 120 excluded development questions, 74 use set operations, 10 need unsupported join or scope features, 4 use other unsupported syntax, and 22 fail the canonical-AST render check. The remaining 10 were excluded by a stale token-range check and are encodable. We retain the fixed 914-question evaluation because no new paired evaluation was run on the corrected 924-question set. In the earlier matched-budget pair, the text baseline answers 29 excluded questions, giving 580 correct on all 1,034 questions against 551 for the compiled anchor when exclusions count as compiled failures. These checkpoints differ from the deployed models.

On test, text answers 36 of the 239 excluded questions. Counting all exclusions as compiled failures reduces the compiled model's accuracy lead from 48 to 12: 1,164 against 1,152 on all 2,147 questions. The full-set difference is not significant under exact McNemar (236 versus 224 discordant wins, $p = 0.608$), Monte Carlo query-cluster sign flips ($p = 0.665$), or Monte Carlo database clustering ($p = 0.755$). Exclusions represent missing grammar coverage, not failed decodes of supported queries. The later canonical-AST render check excludes another 45 questions from the supported set. Its string comparison does not normalize all harmless differences, such as column qualification. A mismatch therefore does not establish semantic inequivalence or that the models' answers were wrong.

The validity transformation also leaves more wrong answers that execute. A descriptive regular-expression analysis of the deployed compiled model's 1,903 scored test predictions finds 397 table sets that differ from the reference query's. Of these, 344 are wrong answers and 53 execute correctly. Thus 344 of the 739 scored misses involve a different table set. This heuristic is not a parser-based or causal error classification.

B Training and evaluation protocol

All three output representations use the same augmented MiniCPM5-1B warm base and the schema-literacy, Spider, and rehearsal curriculum of Section 4, with matched optimizer steps. The shared vocabulary has 130,883 rows: the original 130,560 plus 320 action rows and three control rows. Of the action rows, 224 use keyword or byte surface forms and 96 use the mean host embedding plus noise. The three control rows use descriptive surfaces. Padding has no added row. Vocabulary and initial weights are shared, while sequence length, supervised positions, loss contributions, and segmentation differ. The initialization helper inferred keyword surfaces from 6,106 training and 914 development reference streams, without filtering the split. A retrospective train-only reconstruction gives identical keyword assignments, so this procedural use of development labels does not demonstrate changed initialization vectors. Development was also used for checkpoint and recipe selection. No test targets occur in this initialization corpus.

The earlier plain-SQL checkpoint uses the full 6,106-example stage-two corpus. A later database-disjoint holdout withheld 771 examples from stage two for the compiled run, and menu text matched its remaining 5,335 examples. We did not rerun plain SQL on this smaller corpus. Its larger stage-two set does not make the comparison controlled: the examples and training histories differ. Compiled and menu text share their stage-two and stage-three examples, while menu text drops nine stage-one examples that fail its rendering check. Plain SQL switches training seeds between stages. The holdout is excluded from stage two only, not from the entire curriculum. Both text arms decode freely through the shared augmented vocabulary, although their targets use ordinary text tokens. Menu text's recorded mask label is boilerplate, not an active compiled decoder. Its 512-token cap was never reached (maximum 203). The primary held-out plain-SQL run also stayed below that cap (maximum 116). Early accuracy comparisons used a pooled seed-stratified exact McNemar test, while later comparisons used a cluster-robust estimator that accounts for related questions and repeated seeds. We assess each result under the test specified for that experiment.

Checkpoint selection. The compiled checkpoint is the development winner among five seeds with narrowing channels off: 587, 577, 566, 561, and 566 correct (mean 571.4, sample sd 10.5). Its selected score rises from 587 to 599 with question-number copy and join adjacency. These rules can exclude reference outputs: number copy excludes one development reference stream whose question says "at least 3" but whose SQL uses > 2. Join adjacency excludes two training reference streams and none on the development set when neighbors include foreign keys or case-insensitive shared column names. Number copy excludes LIMIT sites. Both channels retain the original menu when their narrowing is unavailable or empty. Channels-on evaluations are incomplete across seeds. The menu-text control later answered 618 questions, above the five-seed range and contrary to our recorded prediction that it would fall within it (Section 5.4). The plain-SQL anchor scores 551; two later rehearsal seeds score 543 and 544 (mean 546 across the three). The anchor was fixed before these two runs and was not reselected. The common held-out table uses the selected seed-0 compiled checkpoint, the matched seed-2 plain-SQL checkpoint, and the seed-0 stage-three menu-text checkpoint. The compiled recipe underwent more development selection, including eight rejected changes. We apply no multiplicity correction to these development comparisons. A preregistered fairness control applying the compiled rehearsal recipe to text raises its historical score from 549 to 556 ($p = 0.525$), within the registered noise range.

The historical matched-budget pair scores 551 each at its final checkpoints. Selecting both checkpoints by their holdout peaks instead gives 544 for the compiled variant and 565 for text. This is a sensitivity within that earlier pair, not a selection-only comparison for the deployed three-stage system. The historical final-checkpoint comparison also lacks its matched text seed-2 evaluation. Later retraining does not replace that missing replication. Removing index prefixes from menu text gives 600 of 914, compared with 618 for menu text. Its 49-question gap to plain SQL changes the whole rendering and training history, so it is not a pure clause-order effect. It is significant under query clustering but not under exact database clustering ($p = 0.0826$). No corresponding held-out or 8B index-prefix control was run.

Scoring and use of the test split. Our harness compares row multisets, preserving column order but ignoring row order even when the reference query specifies ORDER BY. Thus it measures result agreement under this convention, not full SQL equivalence. Read-only re-execution reproduces all held-out arm and verifier labels. However, 3 compiled, 3 plain-SQL, 5 menu-text, and 8 primary-verifier answers counted correct differ in ordered rows. The official scorers use different conventions (Table 4). The original held-out accuracy test required a positive difference with a two-sided paired sign-flip $p < 0.05$. It groups questions from one database with the same normalized reference SQL query, then compares the observed difference with differences obtained by exchanging the models' results for whole groups. The exact p-value is 0.0457, while the original sampling-based estimate was 0.0459. An exact McNemar test that treats each question separately gives $p = 0.0223$. Grouping instead by database gives exact $p = 0.158$. On the later 1,863-question set retained after the render check, the reference-query test gives $p = 0.057$. The supplied distilled test suites cover none of the 40 test databases, so held-out convention checks use official single-instance execution scoring.

The test split was used in seven completed contacts: the original paired greedy evaluation, text sampling, text evaluation on excluded questions, menu-text evaluation, the first verifier evaluation, the seed controls and second judge, and verification without the compiled candidate. A conditional

8B test plan was withdrawn. The original compiled 1B system was decoded in the first contact only. Later routing analyses reuse saved predictions. Systems and decision rules were fixed before their respective contacts, and no system was selected on test.

C METHOD DETAILS

The runtime loads the catalog, lowers the SQL grammar into a structural model, and binds catalog menus under the versioned layout. At a reference decision, the transition state determines both the payload's role and its admissible menu. To select a column, the model chooses a catalog table already in scope and then a column within that table. This keeps the column's index fixed when other tables enter the query. The trained model rejects catalogs with more than 256 tables or 256 columns per table. We also tested an alternative that puts every in-scope column in one numbered list. It permits multi-digit indices and up to 65,536 entries, but we verified this limit only with large artificial database schemas. No model was trained to use this alternative.

Proof of Proposition 1. We induct over accepted transitions, maintaining the scope stack reconstructed from the prefix. Selecting a table adds its catalog entry to the innermost frame. Selecting a column checks membership in that frame's permitted menu before recording the reference. A repeated table index within a frame is rejected, excluding self-joins. An uncorrelated subquery opens its own frame and cannot refer to the enclosing scope. Closing it restores the outer frame unchanged. Since the stream is FROM-first, a frame's tables are entered before its column decisions. Each accepted transition preserves the invariant, so every reference in a complete accepted stream is in its permitted menu. This argument concerns internal references. It imposes no condition on SQL rendering or completion. A native string span is capped at 64 host tokens and must reach its compiled terminator before the stream can be complete. Budget exhaustion or early stopping can leave no rendered query. The six held-out execution failures comprise five unrendered predictions and one database error. Their taxonomy does not establish a common cause. In the evaluated renderer, a column named `NULL` could render as a null literal and a keyword identifier such as `CASE` could produce a syntax error, and a NUL string can be rejected by SQLite. Later compiler repairs quote these identifiers and reject NUL literals, but the reported model predictions were not regenerated. Round-trip checks shared by parser and renderer do not detect all such errors. External-dialect tests are needed.

The pooled 74.96% census used the earlier 833-id codec, before factored column addressing and native literal spans, so it does not measure the trained 321-id representation. Table 3 groups the 20 model choices in the trained model's target stream for this query. With the factored layout, the compiler has 87 transitions: 67 are fixed and 20 are predicted. Before the model selects column `age`, the decoder supplies the table choice because `head` is the only table in scope. The query has no string literal, so the native-literal extension does not change its target stream. These counts describe compiler transitions, not measured model forward passes. Renaming identifiers without changing catalog structure preserves the menu indices, but changes the prompt and therefore need not preserve the model's choices.

Table 3: **Free decisions in the worked example.** Indices mark positions in the model's output after fixed actions are omitted. Consecutive structural decisions are grouped. Their exact token trace is reproducible from the compiler. The five content decisions are marked by an asterisk.

Positions	Decisions	Number
1	Table `head`*	1
2–8	Alias, from-clause, join, where-presence, expression kinds	7
9–10	Column `age`*, comparison `>`*	2
11	Integer expression kind	1
12	Integer value `56`*	1
13–18	Optional clauses, select-list start, distinct	6
19	Select-item kind `count(*)`*	1
20	End of select list	1

Separate editing probes concern tree-sitter codecs for subsets of JSON and C, not a trained SQL editing model. Of 20 partial or invalid fixtures, only one round-trips exactly and 15 yield no parse prefix. The byte fallback is untrained. These probes do not demonstrate editing, completion, or fill-in-the-middle support.

D MASKING, ROBUSTNESS, AND SCALE CONTROLS

Constrained-decoding baseline. The repaired grammar mask tracks catalog scope, but supports fewer constructs and uses an earlier text checkpoint than the deployed compiled model. Its clean-schema comparison uses 685 questions: accuracy is 62.19% with masking and 62.04% without it (52 versus 51 discordant wins, exact $p = 1.0$). Of the masked decodes, 654 complete, 18 reach the 256-token cap, and 13 reach a state with no admissible token. Greedy masking without lookahead or backtracking can reach such a dead end. These results do not characterize PICARD or Synchromesh. On the same 685 questions after schema names are changed, the repaired baseline answers 46 correctly and fails to execute on 255, including four timeouts. The compiled comparator answers 69 and plain text 19. This contradicts the prediction set before evaluation that masking would preserve validity under renaming. These are earlier checkpoints, and compiled accuracy also falls sharply, from 47.15% to 10.07%. Internal reference validity does not preserve semantic grounding. The results above use the repaired grammar. An earlier version mishandled the capitalization of keywords and aggregate names and the spaces before commas. The repair did not change which SQL constructs the grammar supports. An early renamed run and a separate 512-token-cap control used the older grammar, so those runs do not characterize the repaired version.

Compiled mask ablation. All conditions use the same 914 development questions and retain two observed failures when executing reference SQL queries. Only the middle two rows below isolate score filtering: both retain transition state and native literal handling, with both narrowing channels off.

Condition	Correct	Non-executing
Mask and narrowing channels on	599	8
Masked, narrowing channels off	587	7
Unmasked, state retained	578	53
Range only, native literal handling also removed	0	914

The unmasked condition stops 49 streams at their first inadmissible choice, and 867 predictions are byte-identical to its masked counterpart. The final row combines two removals and is not an isolated mask effect. Under teacher forcing on prefixes of the reference streams, the compiled mask changes the greedy choice at 55 of 26,830 compiled decisions (0.2%), including 25 of 4,022 column-menu decisions. The separate full-vocabulary inadmissible-argmax count is 64 of 27,436 scored positions, including 26 of 4,022 column-menu decisions. These one-step measurements do not describe the model's trajectory after an error. The free-running comparison above shows that the mask still prevents failures. These counts are exploratory reporting under a dated deviation. The originally registered report is incomplete. The original composed run lacks a decoding-mode record. A later repeat reproduces its results but cannot restore that record. Historical dtype, checkpoint-weight hashes, compiler-wheel attestation for the unmasked run, and complete trajectory telemetry are unavailable. The results establish neither learned validity without masking nor a tokenizer-specific accuracy benefit.

Robustness qualifications. The Dr.Spider schema/content results and the nine-set paraphrase result in Section 5.6 use the frozen pair. Compiled/plain correct counts are 900/713 of 2,342 for synonyms, 1,208/874 of 2,552 for abbreviations, and 56/135 of 351 for content equivalence. The paraphrase net is $+26$ of 6,698. The original schema-perturbation preregistration named earlier step-1500 checkpoints and McNemar testing. The frozen-pair analysis uses query-cluster sign flips. Conflicting schema-evaluation timestamps do not establish that its prediction preceded the result. An earlier recorded prediction said the models would differ by less than two percentage points on content equivalence. Instead, both checkpoint pairs favor plain SQL by more than that margin. A post-hoc retention analysis on 3,413 perturbed questions whose originals both arms answered correctly gives 69.53% for the compiled variant and 72.19% for text across all nine paraphrase sets

($p = 0.0594$). An earlier seven-set snapshot gave a larger deficit of 4.75 points ($p \approx 0.02$). The complete nine-set result is the fixed reporting basis. Neither establishes a robustness advantage. Earlier zero-shot synthetic-renaming predictions survive and can be recounted. Only the adapted comparison retains aggregates without its per-question predictions. These studies use different checkpoints, authored difficulty, and extensive development selection, limiting generalization to real catalogs. The exploratory linking-difficulty plan was hypothesis-generating and lacks a complete retained stratum/estimator receipt. We do not count it as a confirmed or refuted prediction. For Spider-Realistic, a post-hoc database-cluster sensitivity gives $p = 0.0166$ for the -24 difference and does not survive the four-test correction. The registered row-level result is therefore sensitive to clustering. The equal-fraction reference assumes loss plus recovery rates of 0.30–0.40 shared by both arms. It is an expectation bracket, not a confidence interval or the row-level test's zero null.

Scale sensitivity. Table 4 retains the frozen 8B comparison and its menu-text control. The 8B generator comparisons use one seed on the development set. A separate test evaluation was planned for a newly trained 8B pair, but that training was deferred and the plan expired before the paper freeze, so no 8B SQL generator was tested on the Spider test split. The 23-question harness deficit gives exact McNemar $p = 0.040$, but the registered sign-flip over 487 reference-query clusters gives exact $p = 0.088$. It therefore does not establish a significant reversal under the registered test. The 8B menu-text variant answers 697 of 914 questions, near the compiled variant's 696, but the training data differ. The menu-text variant reused the 1B control's database-disjoint holdout at stage two, leaving 5,335 examples from 20 fewer training databases than the 6,106 examples used by both other 8B variants. The scorer changes the 8B difference from 23 fewer correct answers for the compiled variant under our harness to eight fewer under Spider's single-instance evaluator and six more under its test-suite metric. The corresponding 1B differences favor the compiled variant by 48, 47, and 27 answers (Table 4). At 8B, plain-SQL execution failures fall from 165 to 48, while compiled failures fall from eight to six. Compiled output uses the reference query's table set more often (841 against 818). Among wrong queries that execute, the heuristic census assigns more compiled errors to column and join choices. Its accuracy deficit shrinks when the scorer accepts the right output columns in a different order. These diagnostics and the Dr.Spider checkpoint sensitivity do not explain the change with model size.

Table 4: **Scoring sensitivity on 914 development questions.** Correct answers under our harness, the official single-instance evaluator, and the official test-suite metric. Compiled "on" and "off" refer to the two narrowing channels. The 1B rows use the same checkpoints as Table 2. Each 8B row is one stage-three seed.

System	Harness	Single instance	Test suite
1B plain SQL	551	575	530
1B compiled, channels on	599	622	557
8B plain SQL	719	749	693
8B compiled, channels on	696	741	699
8B compiled, channels off	689	734	689
8B menu text	697	731	693

The 8B Dr.Spider abbreviation comparison uses 2,552 supported questions. Its frozen stage-three pair scores 1,801 against 1,690, a 4.35-point gap below the preregistered five-point threshold. The stage-two pair scores 1,772 against 1,641, a 5.13-point gap, but was registered only as a sensitivity. It cannot replace the stage-three decision, and neither comparison isolates tokenization from menu representation. The separate 8B nine-set paraphrase-retention prediction was not evaluated and is counted as neither confirmed nor refuted.

E VERIFICATION AND CONFIDENCE CONTROLS

Execution-based fallback. After looking at the original held-out results, we chose plain SQL with compiled output as a fallback: compiled output is used only when plain SQL fails to execute. We then scored this rule from answers already generated. On the original 1,908-question set it answers 1,245 questions, compared with 1,116 for text alone, 1,164 for the compiled variant alone, 1,167

for the mirror rule that prefers compiled output whenever it executes, and 1,140 in expectation for random routing. Vote@4 and vote@8 answer 1,164 and 1,189. The two arms' oracle union is 1,352. The rule's gain comes entirely from text execution failures. The text vote@K baselines use one sampling seed, temperature 1.0, and top-p 0.95. They vote over the results of queries that execute, break ties by the earliest sample supporting a tied result, and score a question wrong if none executes. On the development set, the fallback answers 607 of 914 questions, compared with 551 for greedy plain SQL, 564 for vote@4, and 584 for vote@8. A second independently trained development pair reproduces the fallback gain: 613 correct against 553 for text and 540 for the compiled variant, without a same-seed sampling baseline. No vote@2 baseline was measured. The development and held-out voting runs use different tokenization schemes, so their counts are not one series. On the 1,863 questions shared with menu text, the saved vote@4 and vote@8 results give 1,155 and 1,181 correct. Plain SQL with compiled output as a fallback gives 1,232 correct, while menu text with compiled output as a fallback gives 1,273. The plain-SQL rule and vote@8 use 2.03 and 1.96 times as many estimated model positions as one plain-SQL decode on the original 1,908-question set. These estimates are not measured latency or FLOPs.

A preregistered development control resamples text after execution failure and reaches 588 against fallback's 607 (14 versus 33 discordant wins, McNemar $p = 0.0079$, database sign-flip $p = 0.0273$). Its inferred position budget is $1.182\times$ a text decode with shared prefill or $1.335\times$ with fresh prefill. The saved-prediction fallback comparison charges two full decodes, about $2\times$. Neither latency nor FLOPs were measured. This single-seed comparison establishes no efficiency advantage. A five-configuration pilot on the development set scored each arm's candidates under the other's likelihood. One setup answers 620 questions, compared with 600 when we take compiled output whenever it runs and plain SQL otherwise, or 618 when we reverse that preference. The two-question gain over the stronger rule is not significant ($p = 0.923$). Two configurations make the same choices as their corresponding execution-based rules, while the other two gain 10 and lose 32 correct answers against those rules. These mixed results do not establish a general likelihood-selection advantage.

Verifier controls. The verifier executes candidates, merges equal results, and presents distinct executing candidates in a deterministic per-question shuffle. It sees question and schema text, SQL, and bounded result previews, without reference answers, arm identities, confidence scores, or sample counts. Qwen3-8B uses greedy decoding, reasoning disabled, and a 16-token output cap. The prompt was fixed before test use. On the common held-out set it was called on 679 questions, returned no unparseable reply, and never abstained. One question had no executing candidate and scored wrong without a judge call. Explicit abstentions also score wrong when they occur. The unused parse-error fallback prefers an executing compiled candidate if it represents a shown result class, then the first shown candidate.

Table 5: **Candidate-pool controls.** Correct answers after verification and the oracle ceiling, which counts questions with any correct candidate. Each cell is judge / ceiling. Development and test use 914 and 1,863 questions respectively. The held-out column supplies Figure 2.

Pool	Development	Held out
Three plain-SQL rehearsal seeds	596 / 604	1,227 / 1,242
Three menu-text rehearsal seeds	658 / 679	1,326 / 1,371
Plain SQL and menu text	662 / 677	1,333 / 1,366
Three representations	679 / 716	1,365 / 1,438

Within each text representation, the three seeds share the same trained stage-two checkpoint and vary only the stage-three rehearsal seed. They are not independent repetitions of the full curriculum. The primary verifier criterion required at least 40 additional correct answers over menu-first fallback with query-cluster $p < 0.05$. The comparison with menu text alone used a threshold of 50 with the same significance condition. The later seed controls required gains of at least 30 over plain-SQL seeds and 20 over menu-text seeds. The compiled-candidate ablation also required at least 20. These later tests use exact McNemar $p < 0.05$ as their deciding statistic, with both cluster tests reported alongside it. Their absolute count thresholds were carried unchanged from 914 development questions to 1,863 held-out questions, so their relative margins are smaller on the held-out set. On the common held-out set, menu text alone answers 14 more questions than the fallback that starts with plain SQL. The paired 95% confidence interval for this difference runs from 1.3 percentage points worse

to 2.8 points better, so the data establish neither a lead nor equal accuracy. The held-out gains of 138, 39, and 32 meet those criteria. Their reference-query/database sign-flip Monte Carlo estimates (10,000 draws, seed 0) are respectively $0.0001/0.0001$, $0.0130/0.0482$, and $0.0030/0.0134$, over 1,015 query clusters and 40 databases. On the development set, the 21-question gain over menu seeds meets the McNemar rule ($p = 0.035$), but database clustering gives $p = 0.13$. Adding compiled output gains 17, below the registered 20-question threshold. Adding a fourth, channels-off compiled candidate gives 676, below the three-arm total of 679. The plain-SQL-only verifier control gives the judge the greedy answer and eight sampled answers from the same model. On all 1,908 supported questions, it answers 1,244 correctly, against 1,189 for vote@8 over the sampled answers, a gain of 55 ($p = 6.9 \times 10^{-8}$). On the common 1,863-question set, it answers 1,232, while the best possible choice from those nine answers would answer 1,284. This comparison was assembled after the test run. No verifier over repeated menu-text samples was run.

A second judge, Qwen2.5-Coder-7B-Instruct, reaches 670 on the development set and 1,344 held out, exceeding fallback but trailing the primary judge by 9 and 21. It abstains on 103 held-out questions. The reported score counts these as wrong. Applying the disclosed abstention fallback would give 1,362, a sensitivity rather than the primary score. Both judges select the same SQL on 520 of 679 contested questions. More candidates do not necessarily help. On the development set, adding compiled alternatives raises the pool's oracle union from 716 to 767 but lowers the verifier from 679 to 631. This pool also contains an unregistered text-vote candidate. Restricting the pool to compiled alternatives gives 592 against the original compiled arm's 599. At 8B, the same judge reaches 728 over three representations and 734 over three text runs, against a best single score of 719. Neither gain is significant. These controls limit the result to the tested 1B candidate pools.

Confidence scoring and sensitivity. For the compiled-site score, we average chosen-token log-probabilities over compiled decision sites. Text uses the full sequence. The recorded probabilities are from the full vocabulary, not the distribution renormalized over admissible actions. Neither score is decision entropy. Native literal steps and their terminator contribute only 3.3% of steps and 5.1% of negative log-probability in a 911-question development-set diagnostic. This is not an error-calibration guarantee. For the three-arm held-out analysis, we use the common 1,863 questions, excluding five with no confidence score from the compiled model. Menu text is rescored under teacher forcing, while scores for the other arms were recorded during decoding. Each arm contributes one checkpoint, with its own generated answers and correctness labels.

Quantity	Compiled	Menu text	Plain text
Correct / 1,858	1,153	1,243	1,105
Whole-sequence AUROC	0.8391	0.7664	0.7146
Length-stratified AUROC	0.7950	0.7335	0.7070

Each arm has fixed whole-sequence length tertiles. Stratum AUROCs are weighted by sampled question counts, with paired resampling over 40 databases (20,000 bootstrap draws, seed zero). On the original 1,903-question basis, turning off the compiled narrowing channels gives site AUROC 0.8416 against text's 0.7115. Excluding literal terminators gives 0.8338 for the deployed system. These are diagnostics on a different question set.

The prespecified sensitivity keeps 1,744 questions whose re-encoded menu-text length is unchanged. Compiled minus menu text is $+0.0460$, $[-0.0002, 0.0932]$. Menu minus plain text is $+0.0446$, $[-0.0068, 0.1003]$. Removing 114 questions from ten databases changes their weights. Equal length does not establish equal token IDs, since the original menu-text IDs were not saved. The discrepancy with the primary result remains unresolved and does not support causal attribution to tokenization.