

Schema-Aware Retrieval-Augmented Tool-Use Agents for API Selection, Planning, and Calling: An Empirical Study on API-Bank

Hao Ran^{1*}, Ryan Cooper², Megan Bennett³, Jun Wang⁴

¹Department of Computer Science, University of Massachusetts Amherst, Amherst, MA, USA

²Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA

³Department of Computer Science & Engineering, Washington University in St. Louis, St. Louis, MO, USA

⁴Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

hao.ran.cs@gmail.com

Abstract

Tool-use agents must identify the correct application programming interface, construct a valid call, and preserve execution state across multi-step dialogue. This study evaluates a retrieval-augmented architecture that separates those responsibilities into global retrieval, trajectory-local candidate formation, schema-aware planning, typed argument binding, and execution validation. The experiments use 1,012 API-Bank records drawn from 261 dialogue files: 534 API-call targets and 478 response targets spanning 50 tools, 47 parameter names, and three difficulty levels. All selection models are evaluated with five-fold StratifiedGroupKFold splits by dialogue file, which prevents calls from the same trajectory from appearing in both training and test folds. Global BM25 and latent-semantic embedding retrieval achieve 41.57% and 41.01% Top-1 accuracy. Restricting retrieval to tools exposed by the active trajectory raises Top-1 accuracy to 64.98%-68.16%. The proposed SchemaPlan-RAC pairwise planner reaches 95.32% Top-1, 99.44% Top-3, and 0.973 mean reciprocal rank. Coupled with a deterministic schema-aware argument agent, it obtains 92.26% parameter-name F1, 74.97% argument-value accuracy, 64.42% strict call exact match, and 88.20% schema-executable accuracy. The planner improvement over the strongest manifest embedding baseline is significant under an exact McNemar test ($p < 10^{-38}$). Ablations show that trajectory candidates, schema constraints, and transition state contribute sequential gains. Error analysis identifies free-text values and temporal normalization as the dominant remaining failures.

Keywords—Tool-Use Agents, Function Calling, API Retrieval, Agent Planning, Schema Validation, API-Bank, Retrieval-Augmented Generation.

A. INTRODUCTION

Large language models increasingly operate as controllers rather than closed-form generators: they select an external capability, supply arguments, observe the result, and decide whether another action is needed. API-Bank formalized this setting around tool retrieval, API invocation, and dialogue completion [1]. The structural challenge is that a plausible response can still violate tool availability, schema requirements, argument completeness, or trajectory state. Related work on HDFS anomaly diagnosis shows how retrieved evidence and selective refusal can constrain failure narratives [2], while evidence-grounded financial disclosure QA demonstrates the value of tying generated answers to auditable source material [3].

Tool-learning systems reach external action through complementary designs. Toolformer inserts self-supervised API calls [4]; ReAct interleaves reasoning and action [5]; Gorilla retrieves documentation to reduce API hallucination [6]; and ToolLLM scales training to large real-world registries [7]. RestGPT separates planning, selection, and

execution for REST services [8], whereas HuggingGPT routes requests to expert models [9]. ART retrieves multi-step tool-use patterns [10], and MRKL systems frame language models as dispatchers among symbolic, neural, and knowledge modules [11]. These designs motivate modular evaluation of both semantic choice and executable structure.

Retrieval is therefore central. Retrieval-augmented generation combines generation with external memory [12], REALM couples retrieval and pretraining [13], and dense passage retrieval replaces lexical matching with learned semantic search [14]. Retrieval-summary integration for code intelligence similarly separates finding relevant code from explaining it [15], while long-document contractual analysis emphasizes segmentation and evidence preservation [16]. ColBERT supplies token-level late interaction [17]. BM25 remains an interpretable probabilistic baseline [18]; latent semantic analysis compresses term-document structure [19]; and classical term weighting supports sparse ranking [20]. Sentence-BERT [21], BERT [22], and T5 [23] provide stronger neural alternatives, but this study uses local BM25

and LSA to isolate planning gains without external model downloads.

Planning contributes information that retrieval alone cannot supply. Human-uncertainty distillation shows how modeled disagreement can support calibrated decisions [24], and uncertainty-aware late fusion learns how confidence should shape the combination of multiple signals [25]. Chain-of-thought decomposition [26], self-consistency [27], program-aided execution [28], and affordance-constrained action selection [29] further motivate explicit intermediate decisions. Spectral rank aggregation provides theory for combining ordered preferences [30], while reciprocal rank fusion supplies a simple heterogeneous retriever ensemble [31]. RETRO and nearest-neighbor language models demonstrate inference-time use of external memory [32], [33], and synchronization estimators highlight uncertainty around structured estimates [34]. Together, these ideas motivate a tool agent that narrows candidates, scores schema and state compatibility, and binds arguments only after selection.

The practical failure modes remain tightly coupled. A global retriever confuses similarly worded operations such as AddReminder, ModifyReminder, and QueryReminder. A candidate retriever can still choose an API that conflicts with the current dialogue stage. A correct API selection can fail at execution because a normalized date, authentication token, identifier, list, or Boolean argument has the wrong representation. A system-level evaluation therefore requires more than Top-1 tool accuracy. It needs argument-name accuracy, value accuracy, exact-call accuracy, and an execution-oriented schema check.

This paper presents SchemaPlan-RAC, a retrieval-augmented calling architecture that treats API selection as pairwise planning over the active tool registry. The method combines sparse, latent-semantic, and character retrieval scores with trajectory availability, ToolSearcher discoveries, prior-call state, schema evidence, action compatibility, authentication readiness, and learned transition likelihood. A typed argument agent then normalizes dates, times, credentials, identifiers, lists, content fields, and tool-specific values before a validator checks required target arguments and declared types. The complete pipeline is shown in Fig. 1.

The study contributes a complete evaluation of 1,012 API-Bank records, including 534 call targets and a structural audit of 478 response targets; compares five retrieval and planning families under dialogue-grouped cross-validation; evaluates six end-to-end calling systems; and reports ablations, confidence intervals, latency, parameter-family, complexity, and trace-level error analyses. Implementation uses scikit-learn [35], and paired correctness is assessed with McNemar's test [36].

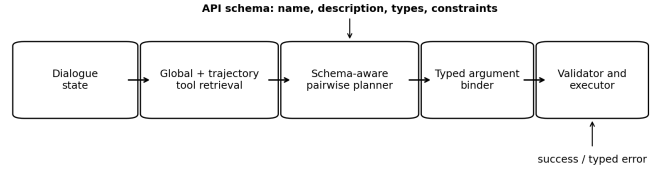


Fig. 1. SchemaPlan-RAC separates retrieval, trajectory-aware planning, typed binding, and execution validation.

B. METHOD

A. Experimental Corpus and Task Definition

The archive contains four JSON arrays: two API-request files and two assistant-response files. Each record provides a source dialogue filename, identifier, API descriptions, context, and expected output. Level 1 covers short interactions; Level 2 adds longer state; and Level 3 introduces ToolSearcher discovery. The primary benchmark has 534 calls (174, 225, and 135 by level), and the response trace has 478 records (170, 205, and 103), spanning 261 dialogue files. API descriptions are parsed into a canonical registry of names, descriptions, parameters, types, and field descriptions. A quote- and bracket-aware splitter parses each target call. The corpus contains 50 target tools and 47 parameter names. The most frequent fields are password (130), token (126), username (120), time (91), date (62), content (50), location (45), and keywords (45); the most frequent tools are GetUserToken (108), ToolSearcher (45), and RecordHealthData (22).

For call record i , the gold output is $y_i = (t_i, a_i)$, where t_i is the target tool and a_i is a dictionary of arguments. Given a dialogue state q_i and registry T , a system predicts a ranked tool list, chooses $\hat{t}_i$, and produces arguments $\hat{a}_i$. The evaluation separates retrieval from calling so that a method can rank the correct tool highly without satisfying the final schema. Response-generation records are not used to train the selector. They are audited to verify that the observed difficulty levels correspond to increasing multi-call structure.

TABLE I. Experimental Corpus By Task And Level

Difficulty	API-call records	Response records	All records
Level 1	174	170	344
Level 2	225	205	430
Level 3	135	103	238
Total	534	478	1,012

TABLE II. Tool And Parameter Inventory

Item	Measured value	Role in evaluation
Dialogue files	261	Group key used for cross-validation
Target APIs	50	Canonical tool registry

Item	Measured value	Role in evaluation
Distinct parameters	47	Names observed in target calls
Most frequent API	GetUserToken (108)	20.22% of call records
Second API	ToolSearcher (45)	Dominant Level-3 discovery action
Most frequent fields	password / token / username	130 / 126 / 120 occurrences

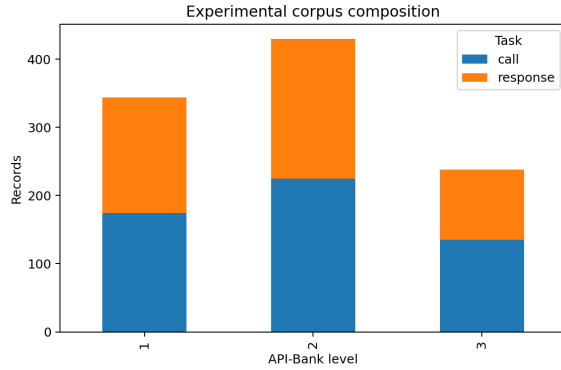


Fig. 2. Composition of the 1,012-record corpus by task and API-Bank level.

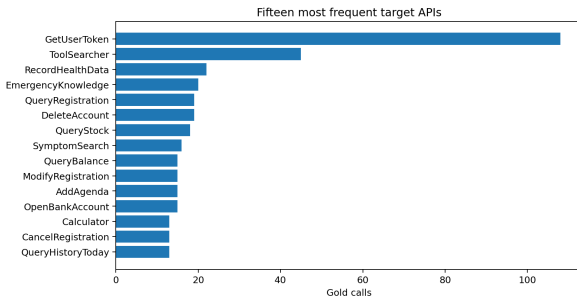


Fig. 3. Frequency of the fifteen most common target APIs among 534 calls.

B. Text and Schema Normalization

The query retains the final twelve nonempty context lines after generation markers are removed, preserving the latest request, clarifications, prior calls, search results, and execution observations. Tool names are split at CamelCase boundaries; each registry document concatenates normalized name, description, parameter names, types, and descriptions. Tokenization lowercases alphanumeric and underscore tokens.

Three corpus-specific rules are applied consistently: implicit month-and-day expressions use 2023; TimedSwitch and CancelTimedSwitch targets treat device_id as a string alias; and Translate requires src and tgt_lang without forcing an omitted source-language field. These rules align parsing, binding, and validation with the observed targets without altering target values.

Trajectory-local candidate sets contain three sources: API descriptions exposed by the current instruction, tool names returned by prior ToolSearcher results, and tool names already observed in prior calls. The first two sources define the retrievable manifest. Prior calls remain separate state features because repetition can either signal an iterative operation or indicate that a different next action is required.

C. Retrieval Baselines

BM25-Global ranks all 50 tool documents using $k1=1.5$ and $b=0.75$. Let $f(w,t)$ denote the frequency of token w in tool document t , $|t|$ its length, and avgdl the mean tool-document length. The score is the sum of inverse-document-frequency-weighted saturation terms [18].

$$\text{BM25}(q,t) = \sum_w \text{IDF}(w) * f(w,t)(k1+1) / [f(w,t)+k1(1-b+b|t|/\text{avgdl})].$$

LSA-Embedding fits word TF-IDF features with unigram and bigram terms on the tool documents and dialogue queries, projects them into a 32-dimensional latent space with truncated singular value decomposition, L2-normalizes the vectors, and ranks tools by cosine similarity. This baseline represents compact embedding retrieval rather than a generative model. A separate character TF-IDF representation with 3- to 5-character word-boundary n-grams supplies a spelling-robust similarity feature to the planner.

BM25-Manifest and LSA-Manifest prioritize tools in the trajectory-local manifest while retaining the global score as a deterministic tie breaker. Hybrid-RRF combines the two manifest rankings with reciprocal rank fusion [31]. With rank $r_m(t)$ from retriever m and constant $k=60$, the fused score is $\sum_m 1/(k+r_m(t))$. These retrieval methods test whether active-tool restriction alone explains performance before schema planning is introduced.

$$\text{RRF}(t) = 1/(60 + \text{rank_BM25}(t)) + 1/(60 + \text{rank_LSA}(t)).$$

D. SchemaPlan-RAC Pairwise Planner

SchemaPlan-RAC frames selection as binary classification over every dialogue-tool pair. Its 25 features combine normalized BM25, LSA, and character scores; manifest, ToolSearcher, and prior-call membership; name and document overlap; action compatibility; parameter-evidence coverage; credential readiness; discovery and authentication gates; tool identity; dialogue level; and a fold-local transition probability from the latest called tool.

Action compatibility maps tool prefixes to dialogue cues. Parameter coverage checks observable support: temporal fields require temporal expressions, email fields require an email pattern, identifiers require labeled or long numeric strings, and token fields require a token or prior authentication call. Gating favors GetUserToken when credentials exist without a token and favors token-requiring tools after acquisition.

Transition probability is estimated only on each training fold with additive smoothing: $(\text{count}(p,t)+0.2)/(\text{count}(p)+0.2|T|)$, where p is the most recent tool. Balanced logistic regression ($C=2.0$, liblinear) scores all 50 candidates. The compact model tests whether explicit registry and state evidence resolves ambiguity left by retrieval.

Table Iii. End-To-End System Configurations

System	Selector	Argument binder	Planning information
BM25-Global + Surface	Global BM25	Literal extraction	No trajectory constraint
LSA-Embedding + Surface	Global 32-D LSA	Literal extraction	No trajectory constraint
BM25-Manifest + Typed	Manifest BM25	Schema-aware normalization	Active tool set
LSA-Manifest + Typed	Manifest LSA	Schema-aware normalization	Active tool set
Hybrid-RRF + Typed	BM25/LSA RRF	Schema-aware normalization	Active tool set
SchemaPlan-RAC + Agent	Pairwise planner	Typed binding + state repair	Schema and transition features

E. Typed Argument Agent and Execution Validator

The surface binder accepts only values whose final normalized string appears literally in the dialogue. The typed binder deterministically parses the corpus's dates, times, ranges, credentials, identifiers, emails, URLs, symbols, formulas, health fields, locations, lists, reminders, alarms, meetings, translation fields, and ToolSearcher keywords.

The agent binder adds state repair: when a required value is absent from the latest turn, it searches prior calls for the same parameter or a declared alias. This supports token reuse and modification flows involving reminders, agendas, alarms, and meetings. Values are emitted only when supported by a parser or prior call.

The validator requires the correct tool, every target key, and a type- or format-valid value. Dates use YYYY-MM-DD except QueryHistoryToday (MM-DD); datetimes use YYYY-MM-DD HH:MM:SS; and email, URL, integer, Boolean, and list fields must satisfy their declared representations. Strict exact match also requires identical key sets and normalized values, so an executable free-text call may still fail exact reproduction.

Binding begins only after selection. Recency and operation type resolve multiple dates, times, credentials, or identifiers; modification operations keep source and destination slots distinct. Lists and Booleans require explicit evidence, state repair copies only same-name or aliased fields, and unsupported required values remain absent for the validator.

F. Evaluation Protocol

Selection metrics are Top-1, Top-3, Top-5, MRR, and mean rank. End-to-end metrics are selection accuracy, micro parameter-name F1, unconditional and correct-tool-conditional value accuracy, strict call exact match, and schema executability. Wrong routing receives zero value credit; parameter-name F1 penalizes both routing and schema errors.

Each call yields one positive and 49 negative dialogue-tool pairs, but results are aggregated over 534 call-level ranked lists rather than 26,700 pairs. Manifest, discovery, and prior-call indicators remain features rather than hard exclusions, allowing recovery from incomplete manifests.

Argument comparison normalizes case and whitespace and stably serializes structured values. Strict match requires the correct API, identical keys, and equal normalized values. Executability requires the correct API, every target key, and type-valid values; additional keys do not invalidate an otherwise executable target. Conditional value accuracy isolates binding after correct routing.

All trainable planner results use five-fold StratifiedGroupKFold grouped by dialogue filename, with 106 or 107 test calls and 51-54 test trajectories per fold. No trajectory crosses train and test. Seed 2026 fixes the decomposition and classifier. Unsupervised retrieval representations use registry and query text only; transition counts and pairwise labels remain fold-local.

Confidence intervals use 5,000 call-level bootstrap resamples. The planner is compared with the strongest manifest embedding baseline by an exact two-sided McNemar test [36]. Post-fit latency is averaged over repeated local execution. Experiments use Python, NumPy, pandas, SciPy, matplotlib, and scikit-learn [35].

C. RESULTS AND DISCUSSION

A. API Retrieval and Selection

Table IV and Fig. 4 show that global retrieval is insufficient for API selection. BM25-Global obtains 41.57% Top-1 and 0.547 MRR, while LSA-Embedding obtains 41.01% Top-1 and 0.569 MRR. The embedding method improves Top-3 from 59.36% to 65.73% and reduces mean rank from 6.40 to 5.15, but it does not improve the first choice. The result reflects name and description overlap among operations that differ primarily by dialogue state, such as add, modify, delete, and query variants.

Trajectory-local manifests produce the first major gain. BM25-Manifest reaches 64.98% Top-1, LSA-Manifest reaches 67.79%, and Hybrid-RRF reaches 68.16%. Their Top-3 values are all above 95.88%, which shows that the active trajectory usually exposes a small set containing the correct tool. However, approximately one third of first choices remain wrong because local candidates still include

multiple compatible actions. Retrieval narrows the action space; it does not fully identify the next state transition.

SchemaPlan-RAC resolves most of that ambiguity. It reaches 95.32% Top-1, 99.44% Top-3, 99.63% Top-5, and 0.973 MRR. The mean rank is 1.08. Relative to LSA-Manifest, the Top-1 gain is 27.53 percentage points. The planner makes only 25 tool-selection errors across 534 calls. The exact paired test reports 152 cases in which the planner is correct and LSA-Manifest is wrong, versus five cases in the opposite direction, yielding $p=8.43 \times 10^{-39}$. The bootstrap 95% interval for planner Top-1 is 93.45%-97.00%.

The progression from retrieval to gated action has close operational analogues. DriftGuard links multi-signal drift detection to controlled retraining and rollback [37]; residual fusion separates VM degradation from noisy-neighbor effects [38]; AIOps work connects learned signals with fault analysis [39]; retrieved normal prototypes ground OpenStack log explanations [40]; and CI diagnosis maps execution logs to repair recommendations [41]. Across these domains, evidence becomes useful only after contextual scoring and an explicit action gate.

Table IV. Api Retrieval And Selection Results

Method	Top-1	Top-3	Top-5	MRR	Mean rank
BM25-Global	41.57%	59.36%	72.28%	0.547	6.40
LSA-Embedding	41.01%	65.73%	78.28%	0.569	5.15
BM25-Manifest	64.98%	96.44%	98.50%	0.806	1.57
LSA-Manifest	67.79%	95.88%	98.31%	0.819	1.60
Hybrid-RRF	68.16%	96.07%	98.69%	0.821	1.54
SchemaPlan-RAC	95.32%	99.44%	99.63%	0.973	1.08

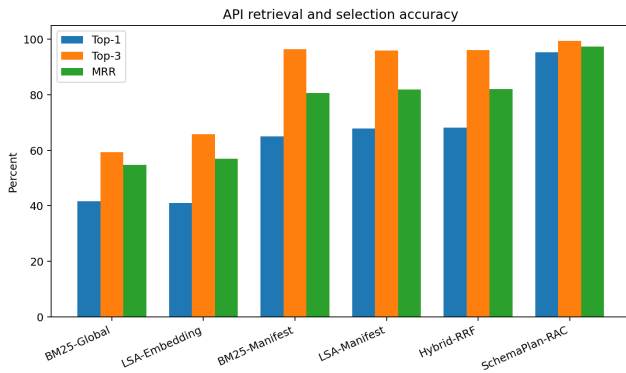


Fig. 4. Retrieval and planner ranking metrics across six API selectors.

B. Difficulty-Level Analysis

Table V and Fig. 5 separate performance by benchmark level. Global BM25 is strongest on Level 1 at 48.28%, falls to 35.56% on Level 2, and recovers slightly to 42.96% on Level 3. LSA-Embedding reaches 47.41% on Level 3 because ToolSearcher language and returned tool descriptions add semantic evidence. Manifest methods perform well on Level

1 and Level 3 but remain weaker on Level 2, where several known tools compete within longer trajectories. Hybrid-RRF obtains 80.46%, 56.44%, and 71.85% on Levels 1-3.

SchemaPlan-RAC maintains 98.85% on Level 1, 95.11% on Level 2, and 91.11% on Level 3. The declining trend is expected because later levels include more state transitions and dynamic tool discovery. The absolute Level-3 result remains 19.26 points above the best manifest baseline. ToolSearcher gating is decisive: before discovery, ToolSearcher is favored when it is the only exposed tool or the dialogue explicitly requests library search; after a result reveals a tool name, the discovered API receives positive evidence and ToolSearcher loses that gate.

Table V. Top-1 Selection Accuracy By Difficulty Level

Method	Level 1	Level 2	Level 3
BM25-Global	48.28%	35.56%	42.96%
LSA-Manifest	79.89%	54.67%	74.07%
Hybrid-RRF	80.46%	56.44%	71.85%
SchemaPlan-RAC	98.85%	95.11%	91.11%

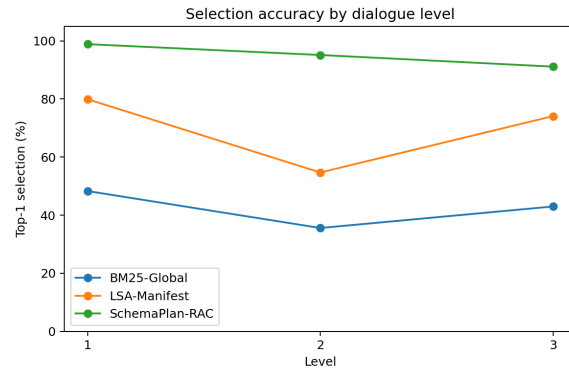


Fig. 5. Top-1 tool selection across the three dialogue levels.

C. End-to-End Calling

Selection gains translate into executable calls only when argument binding is also strengthened. Table VI and Fig. 6 compare all six systems. BM25-Global with surface extraction obtains 18.16% strict exact match and 23.41% schema-executable accuracy. LSA-Embedding with the same binder reaches 21.16% and 25.47%. These baselines fail both because the API is frequently wrong and because natural-language values do not appear in the exact normalized format required by the target call.

Typed binding substantially improves every manifest selector. BM25-Manifest reaches 42.70% strict exact and 60.30% executable accuracy. LSA-Manifest reaches 44.57% and 63.48%, and Hybrid-RRF reaches 44.19% and 63.48%. Conditional value accuracy rises above 51.8% for all three, confirming that schema-guided normalization extracts useful arguments once the selector identifies the correct API.

The complete SchemaPlan-RAC agent achieves 95.32% selection accuracy, 92.26% parameter-name F1, 74.97% unconditional value accuracy, 77.92% conditional value accuracy, 64.42% strict exact match, and 88.20% schema-executable accuracy. The 23.78-point gap between strict exact and executability is informative rather than contradictory. Execution validation accepts correctly typed nonempty content when exact wording differs, whereas strict match requires byte-equivalent normalized values and an identical key set. Free-text fields therefore affect strict accuracy more strongly than execution readiness.

The level-wise full-system results in Table VII show a second distinction. Selection decreases monotonically with difficulty, but strict exact peaks at Level 2 (73.78%) because many Level-2 calls reuse explicit identifiers, tokens, and structured values from earlier turns. Level 3 has the lowest strict exact at 51.11% because ToolSearcher keywords and dynamically discovered tool arguments are open-ended. Executability remains 92.53%, 88.00%, and 82.96% across the three levels.

The end-to-end sequence is also clear in absolute counts. The planner selects the correct API for 509 of 534 calls. The binder and validator produce 471 schema-executable calls, and 344 calls reproduce the complete target exactly. Thus, 38 correctly routed calls still lack a required or type-valid value, while 127 executable calls differ from the target in normalized content or key-set equality. This separation matters because a routing-only evaluation would treat all 509 calls as successful, whereas strict exact match would conceal 127 calls that are structurally ready for execution. Reporting both measures exposes where the remaining work lies: field grounding and value normalization rather than broad API discovery.

Because a wrong call can carry asymmetric cost, executability is closer to a risk-constrained decision metric than to generic text similarity. Comparable concerns appear in C5.0 bank-credit assessment [42], budget-constrained auto-bidding [43], AI-driven financial risk analysis [44], and counterfactual learning-to-rank for advertising [45]. Each setting separates prediction quality from the policy that determines whether and how a model output is acted upon.

Table Vi. End-To-End Api Calling Performance

System	Selection	Param-name F1	Value accuracy	Strict exact	Executable
BM25-Global + Surface	41.57%	33.25%	25.45%	18.16%	23.41%
LSA-Embedding + Surface	41.01%	34.05%	27.61%	21.16%	25.47%
BM25-Manifest + Typed	64.98%	63.44%	49.20%	42.70%	60.30%
LSA-Manifest + Typed	67.79%	66.42%	51.53%	44.57%	63.48%

System	Selection	Param-name F1	Value accuracy	Strict exact	Executable
Hybrid-RRF + Typed	68.16%	66.80%	51.81%	44.19%	63.48%
SchemaPlan-RAC + Agent	95.32%	92.26%	74.97%	64.42%	88.20%

Table VII. Schemaplan-Rac Performance By Level

Level	N	Selection	Strict exact	Executable
Level 1	174	98.85%	62.64%	92.53%
Level 2	225	95.11%	73.78%	88.00%
Level 3	135	91.11%	51.11%	82.96%

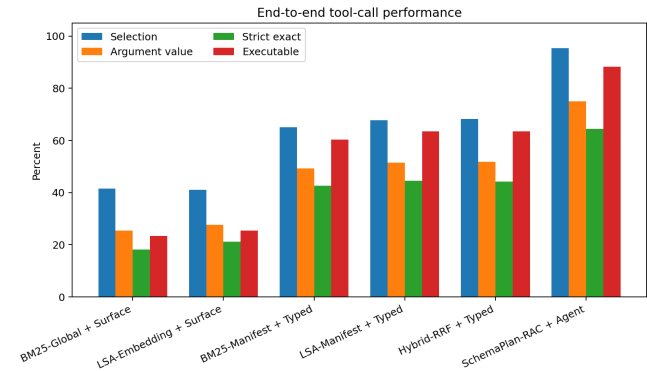


Fig. 6. End-to-end selection, argument, exact-call, and executability metrics.

D. Argument Families and Call Complexity

Table VIII and Fig. 7 divide gold arguments into five families. Authentication values reach 82.70% exact accuracy across 422 instances, and entity or identifier fields reach 82.49% across 177 instances. These fields benefit from explicit labels, stable patterns, and prior-call reuse. Numeric or structured fields reach 74.42%, and temporal fields reach 73.20%. Temporal errors remain frequent because the corpus contains date-only values, date ranges, standalone times, day-boundary conventions, and tool-specific formats such as MM-DD for QueryHistoryToday.

Free-text and content values are the main weakness at 55.04% exact accuracy across 238 instances. This family includes ToolSearcher keywords, questions, symptoms, meeting topics, locations, message content, translation text, and music names. The binder deliberately avoids semantic rewriting; it extracts spans or prior values. As a result, a concise target phrase and a longer dialogue-derived phrase can both be executable strings while failing exact equality.

Call complexity creates a clear gradient. Table IX and Fig. 8 show 99.45% selection, 74.59% strict exact, and 97.79% executability for calls with zero or one argument. For two or three arguments, the values are 93.60%, 65.66%, and 86.20%. Calls with four to six arguments retain 91.07% selection but fall to 25.00% strict exact and 67.86% executability. Each

additional field creates another opportunity for an omitted value, formatting error, or nonidentical free-text span. The planner remains robust because tool selection relies on the entire schema rather than complete argument extraction.

The same evidence-to-action distinction appears in enterprise marketing decisions [46], constrained budgeting that combines demand and response models [47], uncertainty-aware VIX forecasting [48], and personalized e-commerce recommendation [49]. These applications differ from API calling, yet all require heterogeneous signals to be converted into a bounded decision rather than treated as an unconstrained score.

Modular fusion is likewise central to autonomous-driving perception [50], attention-enhanced image detection [51], spatiotemporal digital-twin dispatching [52], and traffic-sign explanation cards that connect localization, classification, and retrieved evidence [53]. The common design lesson is to preserve separate perception, evidence, planning, and presentation stages so that an error can be localized before an action is issued.

Representation and interface choices further determine whether model output is usable. Progressive PDF rendering prioritizes informative regions [54], edge VMAF distillation produces compact quality signals [55], and financial risk dashboards organize explanations through visual hierarchy [56]. Clinical prediction studies on multiple-myeloma residual disease [57] and DenseNet cancer imaging [58] similarly show that an operational prediction depends on explicit input representation and decision context.

Table Viii. Argument Value Accuracy By Parameter Family

Family	Gold values	Correct-tool rate	Exact value accuracy
Authentication	422	90.52%	82.70%
Temporal	250	95.20%	73.20%
Free text/content	238	92.44%	55.04%
Entity/identifier	177	100.00%	82.49%
Numeric/structured	86	95.35%	74.42%

Table Ix. Performance By Gold Argument Count

Arguments	Calls	Selection	Strict exact	Executable
0-1	181	99.45%	74.59%	97.79%
2-3	297	93.60%	65.66%	86.20%
4-6	56	91.07%	25.00%	67.86%

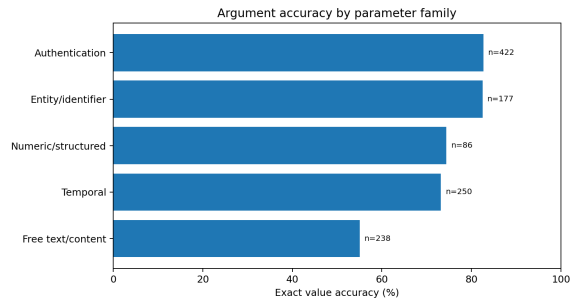


Fig. 7. Exact argument-value accuracy by semantic parameter family.

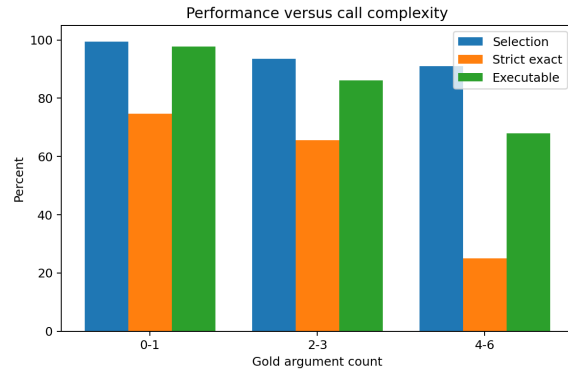


Fig. 8. Tool selection and calling accuracy as argument count increases.

E. Ablation, Efficiency, and Statistical Reliability

The ablation in Table X isolates the planner's information sources. Retrieval scores alone reach 49.06% Top-1. Adding trajectory candidate indicators raises accuracy to 81.65%, an increase of 32.58 points. Adding schema constraints raises it to 89.33%. The fold-specific transition feature produces the final 95.32%. The progression demonstrates that semantic relevance, active-tool availability, parameter feasibility, and dialogue transition each solve a different part of the selection problem.

Table XI reports local inference latency after model fitting. BM25 scores all 50 tools in 0.333 ms per query. Word TF-IDF transformation requires 0.192 ms, character TF-IDF 0.449 ms, and the pairwise logistic planner scores 50 candidate rows in 0.032 ms once features are available. These measurements exclude one-time vectorizer and classifier fitting. The complete nonneural pipeline therefore remains suitable for local experimentation and high-throughput routing.

Table XII summarizes uncertainty and significance. Planner Top-1 is 95.32% with a bootstrap interval of 93.45%-97.00%. Full-system executability is 88.20% with an interval of 85.39%-90.82%. The planner's paired improvement over LSA-Manifest is statistically decisive. The narrow confidence intervals also show that the main conclusions do not depend on a small number of trajectories.

Table X. Schemaplan-Rac Feature Ablation

Feature set	Top-1	Top-3	MRR	Mean rank
Retrieval scores only	49.06%	73.03%	0.640	3.87
+ trajectory candidates	81.65%	98.50%	0.902	1.26
+ schema constraints	89.33%	99.25%	0.943	1.14
Full + transition	95.32%	99.44%	0.973	1.08

Table Xi. Measured Local Inference Latency

Operation	Milliseconds per query
BM25 scoring	0.333
Word TF-IDF transform	0.192
Character TF-IDF transform	0.449
Schema planner (50 candidates)	0.032

Table Xii. Bootstrap Intervals And Paired Test

Metric	Estimate	95% interval	Comparator	p-value
Planner Top-1	95.32%	[93.45%, 97.00%]	LSA-Manifest	8.43e-39
Full-system executability	88.20%	[85.39%, 90.82%]	-	-

F. Error Analysis and Response-Trace Consistency

The observed error counts in Table XIII and Fig. 9 are nonexclusive because one call can contain multiple argument failures. Free-text or content mismatches occur in 81 calls, temporal mismatches in 53, and missing arguments in 38. Wrong-API errors occur in only 25 calls, consistent with 95.32% selection. Entity or identifier mismatches occur in 22 calls, authentication mismatches in 11, and numeric or structured mismatches in seven. The distribution confirms that future gains depend more on argument grounding than on broader retrieval. Deep-autoencoder IoT traffic classification similarly depends on separating benign variation from actionable anomalies [59].

Free-text errors concentrate in ToolSearcher keywords and content-bearing communication or scheduling calls. A target often uses a compressed noun phrase, while deterministic extraction retains a complete request clause. Temporal errors arise from repeated dates in a trajectory, end-of-day expansion, and choosing the new rather than old value during modification. Missing arguments appear most often in four-to six-field calls, especially meetings and hotels. Wrong-API errors cluster in operations sharing the same entity and action family, including query versus modify variants. Predictive DDoS mitigation adds the further requirement that a detection score be converted into a controlled intervention [60].

The response records provide an independent check on trajectory structure. Table XIV shows that every response record contains at least one prior API call. The mean number of prior calls increases from 1.28 at Level 1 to 1.98 at Level 2 and remains 1.87 at Level 3. Multi-call traces account for 26.47%, 61.95%, and 66.02% of the three levels.

ToolSearcher appears in 96.12% of Level-3 response traces and in none of Levels 1 or 2. This pattern matches the design assumption used by the planner: Level 3 emphasizes dynamic discovery, while Level 2 emphasizes known-tool state transitions.

The combined results answer the central research question. Retrieval determines which tools are semantically plausible, but state and schema determine which action is executable now. Candidate restriction supplies most of the first gain, schema evidence removes tools whose required fields lack support, and transition likelihood resolves operations that remain semantically and structurally similar. Typed binding then converts the selected plan into a valid call. The architecture therefore benefits from a disciplined separation of retrieval, planning, binding, and validation rather than a single unconstrained generation step.

The response-trace audit confirms the planner's state assumptions. Level 1 is mostly single-call, whereas nearly two thirds of Level-2 and Level-3 records contain multiple prior calls. ToolSearcher appears in 99 of 103 Level-3 traces and in neither earlier level. Longer traces are not uniformly harder: stable tokens and identifiers improve Level-2 binding, while open-ended discovery and content fields reduce Level-3 exact match.

Table Xiii. Full-System Error Categories

Error category	Observed cases
Free text/content value	81
Temporal value	53
Missing argument	38
Wrong API	25
Entity/identifier value	22
Authentication value	11
Numeric/structured value	7

Table Xiv. Structural Audit Of Response-Generation Traces

Level	N	Mean prior calls	Multi-call	ToolSearcher	Success marker
Level 1	170	1.28	26.47%	0.00%	29.41%
Level 2	205	1.98	61.95%	0.00%	57.56%
Level 3	103	1.87	66.02%	96.12%	19.42%

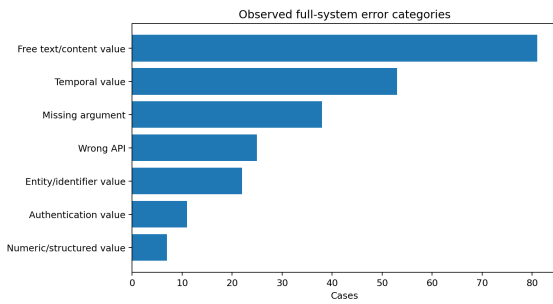


Fig. 9. Nonexclusive error categories observed in the complete SchemaPlan-RAC calling system.

D. LIMITATIONS

The evaluated archive is a compact API-Bank subset with 50 tools and 1,012 records, so conclusions apply to this corpus rather than the full benchmark [1]. Selection gains also depend on registry descriptions and trajectory-local exposure; deployments with incomplete, stale, or adversarial manifests require explicit out-of-registry rejection.

The local LSA baseline does not represent the accuracy or cost of a pretrained sentence encoder [21]. It isolates planning and avoids external downloads. A stronger neural retriever could improve global ranking, but the ablation still shows that retrieval-only features trail trajectory and schema planning.

Argument binding is deterministic and English-specific. It handles the date, time, credential, scheduling, health, search, communication, hotel, and device patterns observed in the corpus, but it does not perform general semantic paraphrase resolution. This constraint explains the lower exact accuracy for free-text fields. The execution metric validates target-required keys and types; it does not contact live services, measure downstream side effects, or verify that an external database accepts a semantically unusual string.

The evaluation also omits human-subject outcomes. Research on bilingual computer-science education shows that language and presentation can affect engagement and learning [61], so future tool-agent studies should test whether operators understand retrieved evidence, uncertainty, and refusal states.

Supervised labels, transition counts, and classifiers are fold-local, while unsupervised representations use all registry and dialogue text. This prevents trajectory leakage but does not test unseen schemas. A leave-tools-out design would instead measure transfer to APIs with no positive training pairs.

Generalization to new APIs remains open. Domain-bridging methods based on approximately shared features offer one route for transfer across partially aligned schemas [62]. Deployment is likewise untested across heterogeneous infrastructure; hybrid-cloud LLM architectures foreground the cost, privacy, and placement constraints that a production router would need to model [63].

Some targets contain benchmark-specific aliases or optional fields. The evaluator recognizes `device_id` for timed-switch calls and does not require an omitted Translate source-

language field. These explicit normalizations preserve target consistency but limit direct comparison with stricter schema evaluators.

V. Conclusion

This study demonstrates that reliable API calling requires more than semantic retrieval. On 534 API-call targets from a 1,012-record API-Bank corpus, global BM25 and latent-semantic retrieval achieve about 41% Top-1 accuracy. Trajectory-local manifests raise selection to 65%-68%, and the SchemaPlan-RAC pairwise planner raises it to 95.32%. The complete retrieval, planning, binding, and validation pipeline obtains 64.42% strict call exact match and 88.20% schema-executable accuracy.

The ablation identifies a consistent sequence of improvements: semantic scores establish relevance, active trajectory candidates constrain availability, schema features test feasibility, and transition state selects the next action. Parameter analysis shows that authentication and identifiers are handled reliably, while free-text and temporal values account for most remaining errors. Complexity analysis shows that tool selection remains strong even when exact argument matching declines on calls with four to six fields.

The resulting architecture is compact, interpretable, and reproducible. It runs with sparse and latent-semantic retrieval, a balanced logistic planner, deterministic argument normalization, and typed validation. The empirical evidence supports a modular design principle for tool-use agents: retrieve broadly enough to preserve recall, plan against the active schema and dialogue state, bind only evidence-backed values, and validate before execution.

E. REFERENCES

- [1] M. Li et al., "API-Bank: A comprehensive benchmark for tool-augmented LLMs," in Proc. EMNLP, 2023, pp. 3102–3116.
- [2] J. Nie and D. Zheng, "Ambiguity-aware HDFS log anomaly detection with retrieval-augmented failure narratives and selective refusal," J. Adv. Comput. Syst., vol. 3, no. 1, pp. 66–80, Jan. 2023, doi: 10.69987/JACS.2023.30105.
- [3] S. Zhou, Z. Li, and E. Wang, "Evidence-grounded RAG for tokenized trade receivable disclosure QA under U.S. capital market standards," J. Adv. Comput. Syst., vol. 3, no. 7, pp. 41–57, Jul. 2023, doi: 10.69987/JACS.2023.30704.
- [4] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in Adv. Neural Inf. Process. Syst., vol. 36, 2023.
- [5] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in Proc. ICLR, 2023.
- [6] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive APIs," arXiv:2305.15334, 2023.

- [7] Y. Qin et al., "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," arXiv:2307.16789, 2023.
- [8] Y. Song et al., "RestGPT: Connecting large language models with real-world RESTful APIs," arXiv:2306.06624, 2023.
- [9] Y. Shen et al., "HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face," in *Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 38154–38180.
- [10] B. Paranjape et al., "ART: Automatic multi-step reasoning and tool-use for large language models," arXiv:2303.09014, 2023.
- [11] Karpas et al., "MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," arXiv:2205.00445, 2022.
- [12] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9459–9474.
- [13] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Retrieval augmented language model pre-training," in *Proc. ICML*, vol. 119, 2020, pp. 3929–3938.
- [14] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020, pp. 6769–6781.
- [15] Y. Li, "Findable then explainable: Retrieval-summary integration for code intelligence on a lightweight CodeSearchNet subset," *J. Adv. Comput. Syst.*, vol. 4, no. 7, pp. 65–82, Jul. 2024, doi: 10.69987/JACS.2024.40706.
- [16] S. Zhou, Z. Li, and E. Wang, "Long-document RAG for contractual and insurance clause analysis in receivables RWA structures," *J. Adv. Comput. Syst.*, vol. 4, no. 8, pp. 88–104, Aug. 2024, doi: 10.69987/JACS.2024.40810.
- [17] O. Khatib and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," in *Proc. ACM SIGIR*, 2020, pp. 39–48.
- [18] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009.
- [19] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
- [20] Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
- [21] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.
- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
- [23] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
- [24] Z. S. Zhong, R. Ma, and H. Zhao, "Human-uncertainty distillation for calibrated vision models on CIFAR-10H," *J. Adv. Comput. Syst.*, vol. 3, no. 2, pp. 77–89, Feb. 2023, doi: 10.69987/JACS.2023.30206.
- [25] Q. Xin, "Uncertainty-aware late fusion for 3D perception: Confidence calibration and fusion rule learning," *J. Technol. Informatics Eng.*, vol. 4, no. 1, pp. 215–238, Feb. 2025, doi: 10.51903/jtie.v4i1.485.
- [26] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24824–24837.
- [27] X. Wang et al., "Self-consistency improves chain of thought reasoning in language models," in *Proc. ICLR*, 2023.
- [28] L. Gao et al., "PAL: Program-aided language models," in *Proc. ICML*, vol. 202, 2023, pp. 10764–10799.
- [29] M. Ahn et al., "Do as I can, not as I say: Grounding language in robotic affordances," in *Proc. CoRL*, vol. 205, 2023, pp. 287–318.
- [30] Z. S. Zhong and S. Ling, "Improved theoretical guarantee for rank aggregation via spectral method," *Inf. Inference*, vol. 13, no. 3, Art. no. iaae020, Sep. 2024, doi: 10.1093/imaai/iaae020.
- [31] V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proc. ACM SIGIR*, 2009, pp. 758–759.
- [32] S. Borgeaud et al., "Improving language models by retrieving from trillions of tokens," in *Proc. ICML*, vol. 162, 2022, pp. 2206–2240.
- [33] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," in *Proc. ICLR*, 2020.
- [34] Z. S. Zhong and S. Ling, "Uncertainty quantification of spectral estimator and MLE for orthogonal group synchronization," arXiv:2408.05944, Aug. 2024.
- [35] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [36] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.
- [37] Zhang, "DriftGuard: Multi-signal drift early warning and safe re-training/rollback for CTR/CVR models," *J. Adv. Comput. Syst.*, vol. 3, no. 7, pp. 24–40, 2023, doi: 10.69987/JACS.2023.30703.
- [38] Nie and D. Zheng, "Noisy-neighbor-aware VM degradation risk modeling with unsupervised residual

- fusion,” *J. Adv. Comput. Syst.*, vol. 4, no. 4, pp. 112–123, Apr. 2024, doi: 10.69987/JACS.2024.40409.
- [39] Y. He, Y. Pan, Y. Wang, S. Du, and Q. Xin, “Intelligent fault analysis with AIOps technology,” *J. Theory Pract. Eng. Sci.*, vol. 4, no. 1, pp. 94–100, Feb. 2024, doi: 10.53469/jtpes.2024.04(01).13.
- [40] Q. Xin, “Explaining OpenStack failure-injection log anomalies with retrieved normal prototypes,” *Emerg. Inf. Sci. Technol.*, vol. 6, no. 2, pp. 125–146, Nov. 2025, doi: 10.18196/eist.v6i2.31232.
- [41] H. Zhang, “LLM-driven CI failure diagnosis and automated repair: From GitHub Actions logs to patch recommendation,” *J. Technol. Informatics Eng.*, vol. 4, no. 1, pp. 190–214, Feb. 2025.
- [42] Q. Xin, R. Song, Z. Wang, Z. Xu, and F. Zhao, “Enhancing bank credit risk management using the C5.0 decision tree algorithm,” *J. Comput. Technol. Appl. Math.*, vol. 1, no. 4, pp. 100–107, Nov. 2024, doi: 10.5281/zenodo.14032041.
- [43] H. Zhang, “Risk-aware budget-constrained auto-bidding under first-price RTB: A distributional constrained deep reinforcement learning framework,” *J. Adv. Comput. Syst.*, vol. 4, no. 6, pp. 30–47, Jun. 2024, doi: 10.69987/JACS.2024.40603.
- [44] T. Yang, Q. Xin, X. Zhan, S. Zhuang, and H. Li, “Enhancing financial services through big data and AI-driven customer insights and risk analysis,” *J. Knowl. Learn. Sci. Technol.*, vol. 3, no. 3, pp. 53–62, Jul. 2024, doi: 10.60087/jklst.vol3.n3.p53-62.
- [45] H. Zhang, “Counterfactual learning-to-rank for ads: Off-policy evaluation on the Open Bandit Dataset,” *J. Adv. Comput. Syst.*, vol. 5, no. 12, pp. 1–11, 2025.
- [46] Wang, Q. Xin, Y. Liu, J. Wang, and T. Yang, “Predicting enterprise marketing decision making with intelligent data-driven approaches,” *J. Ind. Eng. Appl. Sci.*, vol. 2, no. 3, pp. 12–19, Jun. 2024, doi: 10.5281/zenodo.11357252.
- [47] Y. Lu, H. Zhou, and Y. Zhang, “A constrained, data-driven budgeting framework integrating macro demand forecasting and marketing response modeling,” *J. Technol. Informatics Eng.*, vol. 4, no. 3, Dec. 2025, doi: 10.51903/jtie.v4i3.466.
- [48] H. Zhou and K. Zhang, “News-based uncertainty and macro-market fusion for VIX direction forecasting: Evidence from 2015–2024 FRED panel,” *J. Technol. Informatics Eng.*, vol. 4, no. 2, pp. 487–501, Aug. 2025, doi: 10.51903/jtie.v4i2.540.
- [49] Xu, H. Zhou, H. Zheng, M. Zhu, and Q. Xin, “Intelligent classification and personalized recommendation of e-commerce products based on machine learning,” *Appl. Comput. Eng.*, vol. 64, no. 1, pp. 143–149, May 2024, doi: 10.54254/2755-2721/64/20241365.
- [50] Y. Wang, S. Du, Q. Xin, Y. He, and W. Qian, “Autonomous driving system driven by artificial intelligence perception fusion,” *Acad. J. Sci. Technol.*, vol. 9, no. 2, pp. 193–198, Feb. 2024, doi: 10.54097/e0b9ak47.
- [51] Z. Ling, Q. Xin, Y. Lin, G. Su, and Z. Shui, “Optimization of autonomous driving image detection based on RFACnv and triplet attention,” *Appl. Comput. Eng.*, vol. 77, no. 1, pp. 210–217, 2024, doi: 10.54254/2755-2721/77/2024MA0067.
- [52] Zhang, R. Ma, and P. Greg, “Digital-twin dispatching for urban mobility via spatio-temporal transformers and offline reinforcement learning,” *J. Technol. Informatics Eng.*, vol. 4, no. 2, pp. 337–363, Aug. 2025, doi: 10.51903/jtie.v4i2.501.
- [53] R. Ma, L. Zhang, and T. Song, “Computer-vision-informed visual explanation cards for autonomous-driving traffic-sign alerts: Localization, classification, and retrieved evidence on GTSDb,” *Int. J. Graph. Des.*, vol. 3, no. 2, pp. 455–473, Oct. 2025, doi: 10.51903/ijgd.v3i2.3991.
- [54] H. Wang, Y. Ren, and X. Chang, “Layout-aware progressive PDF rendering: AI prioritization of PDF slices to reduce time-to-functional-first-frame on FUNSD,” *J. Technol. Informatics Eng.*, vol. 4, no. 2, pp. 425–446, Aug. 2025, doi: 10.51903/jtie.v4i2.523.
- [55] B. Zhou, H. Wang, and X. Chang, “Distilling VMAF into an edge-deployable quality predictor: A pilot shot-level proxy with LLM-ready quality tokens,” *J. Technol. Informatics Eng.*, vol. 4, no. 2, pp. 447–463, Aug. 2025, doi: 10.51903/jtie.v4i2.522.
- [56] Z. Li, S. Zhou, and Z. Zhou, “Financial risk dashboard design for institutional RWA investors: Visual hierarchy, chart comprehension, and explainability in FinChart-Bench,” *Int. J. Graph. Des.*, vol. 3, no. 1, pp. 196–210, May 2025, doi: 10.51903/ijgd.v3i1.3715.
- [57] J. Chen, J. Xiong, Y. Wang, Q. Xin, and H. Zhou, “Implementation of an AI-based MRD evaluation and prediction model for multiple myeloma,” *Front. Comput. Intell. Syst.*, vol. 6, no. 3, pp. 127–131, Jan. 2024, doi: 10.54097/zJ4MnbWW.
- [58] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, “Cancer image classification based on DenseNet model,” *J. Phys.: Conf. Ser.*, vol. 1651, no. 1, Art. no. 012143, Nov. 2020, doi: 10.1088/1742-6596/1651/1/012143.
- [59] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, “IoT traffic classification and anomaly detection method based on deep autoencoders,” *Appl. Comput. Eng.*, vol. 69, no. 1, pp. 64–70, Jul. 2024, doi: 10.54254/2755-2721/69/20241511.
- [60] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, “Predictive optimization of DDoS attack mitigation in distributed systems using machine learning,” *Appl. Comput. Eng.*, vol. 64, no. 1, pp. 89–94, May 2024, doi: 10.54254/2755-2721/64/20241350.
- [61] G. S. Raj et al., “Impact of bilingual CS education on student learning and engagement in a data structures course,” in *Proc. 19th Koli Calling Int. Conf. Comput.*

- Educ. Res., 2019, pp. 1–10, doi: 10.1145/3364510.3364518.
- [62] Z. S. Zhong, X. Pan, and Q. Lei, “Bridging domains with approximately shared features,” in Proc. AISTATS, PMLR, vol. 258, 2025, pp. 559–567.
- [63] Q. Xin, “Hybrid cloud architecture for efficient and cost-effective large language model deployment,” J. Inf. Syst. Informatics, vol. 7, no. 3, pp. 2182–2195, Sep. 2025, doi: 10.51519/journalisi.v7i3.1170.