

RouteEval: A Benchmark for Evaluating LLM Tool Calling in Running Route Generation

Veejay Roy
usroyvj@gmail.com

Roger Jin
roger@nousresearch.com
NousResearch

Abstract

Large Language Models (LLMs) are increasingly being used in applications requiring interaction with external tools and APIs. However, systematic evaluation of their tool-calling capabilities in real-world geographic tasks remains limited. We present RouteEval, a benchmark designed to evaluate LLMs’ ability to generate accurate running routes through tool calling. Route generation presents unique challenges beyond general tool use: it requires spatial reasoning, precise numerical constraint satisfaction, and real-world validation. Unlike mathematical calculation tasks that test numerical precision in isolation, route generation demands that models synthesize spatial knowledge, interpret user preferences, and produce physically feasible results validated against real-world constraints. Our benchmark assesses 13 state-of-the-art models across 50 diverse prompts with 16 runs each (totaling 800 evaluations per model), measuring their capability to use a route generation tool effectively while adhering to distance constraints. We evaluate routes using Google Maps distance validation and introduce a novel accuracy metric that penalizes deviation from target distances. Our analysis reveals significant variation in model performance, with GPT-5 (High) achieving the highest average accuracy (0.650) but lower success rate (91.1%), while Claude-Sonnet-4 demon-

strates perfect success rate (100.0%) with moderate accuracy (0.458). These results highlight the complex trade-offs between reliability and precision in LLM tool calling for location-based services. RouteEval provides a foundation for understanding and improving LLM tool-calling capabilities in practical geographic applications.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation. Recent developments have extended these capabilities to include interaction with external tools and APIs through structured tool calling mechanisms. This advancement enables LLMs to perform complex tasks that require accessing real-time data, performing calculations, or interacting with specialized services.

Running route generation represents a particularly challenging application domain for LLM tool calling that combines spatial reasoning, numerical precision, and practical constraint satisfaction. Unlike mathematical calculation benchmarks that test numerical accuracy in isolation, route generation requires models to: (1) understand and interpret natural language descriptions of user preferences and constraints, (2) synthesize geographic knowledge about locations, land-

marks, and terrain, (3) generate physically feasible routes that respect distance targets while accommodating user preferences, and (4) utilize tool-calling APIs to produce structured, validable outputs. This combination of natural language understanding, spatial reasoning, numerical precision, and tool orchestration makes route generation an ideal testbed for evaluating LLM capabilities beyond purely text-based tasks.

The need for specialized benchmarks in this domain stems from the limitations of existing evaluation frameworks. Current tool-calling benchmarks often focus on simple function invocation or lack domain-specific validation mechanisms that test real-world applicability. Mathematical benchmarks, while useful for testing numerical precision, do not capture the complexity of synthesizing spatial knowledge and user preferences into actionable results. Route generation bridges this gap by providing a concrete, numerically grounded evaluation scenario that requires both tool-calling proficiency and domain-specific reasoning.

Despite the growing importance of tool-calling capabilities in location-based services, systematic evaluation frameworks for assessing LLMs in geographic tasks remain limited. This gap motivates our work on RouteEval, a specialized benchmark for evaluating LLM tool calling specifically in the context of running route generation.

RouteEval introduces several key contributions:

- A comprehensive benchmark of 13 state-of-the-art LLMs evaluated on 50 diverse route generation prompts
- A novel accuracy metric that measures deviation from target distances using Google Maps validation
- Analysis of 10,400 total evaluations (50 prompts \times 16 runs \times 13 models) revealing distinct patterns in model reliability and precision

- Insights into the trade-offs between success rate and accuracy in tool-calling tasks

Our evaluation reveals that current LLMs exhibit significant variation in their tool-calling capabilities for geographic tasks. Models demonstrate different strategies in balancing success rate (ability to complete the task) versus accuracy (precision in meeting distance constraints). These findings have important implications for deploying LLMs in location-based services and highlight areas for improvement in tool-calling methodologies.

2 Related Work

2.1 LLM Tool Calling and Function Calling

Recent advances in LLMs have enabled structured interaction with external tools through function calling APIs. OpenAI’s function calling capability OpenAI [2024] allows models to output structured JSON that can be parsed and executed by external systems. This paradigm has been extended by other providers including Anthropic’s Claude Anthropic [2024], Google’s Gemini Google [2024a], and open-source models.

Tool calling represents a critical capability for practical LLM applications, enabling models to access real-time information, perform calculations, and interact with external services Wei et al. [2022], Gao et al. [2023], Mialon et al. [2023]. Recent benchmarks have emerged to evaluate tool-calling capabilities in domains such as code generation Chen et al. [2023], web navigation Zhou et al. [2023], and API usage Patil et al. [2023]. However, systematic evaluation of these capabilities across different models and domains remains limited Liu et al. [2023], particularly for domain-specific tasks requiring numerical precision and real-world validation.

2.2 Geographic and Location-Based LLM Applications

Geographic reasoning represents a challenging domain for LLMs due to requirements for spatial understanding, numerical precision, and real-world constraint satisfaction. Recent work has explored LLM capabilities in geospatial tasks Han et al. [2024], Liu et al. [2024], including location understanding Hu et al. [2023], spatial reasoning Zhang and Wang [2024], and geographic question answering. However, comprehensive benchmarks specifically designed for tool-calling evaluation in geographic domains remain lacking. Existing tool-calling benchmarks Chen et al. [2023], Zhou et al. [2023], Patil et al. [2023] often focus on single-tool invocation or lack the numerical precision requirements inherent in geographic tasks. Route generation adds additional complexity beyond simple geographic queries, requiring models to balance multiple constraints including distance targets, location preferences, and practical feasibility. Our work addresses this gap by providing a systematic evaluation framework for route generation through tool calling that combines tool-calling assessment with domain-specific validation.

2.3 LLM Benchmarking and Evaluation

Existing LLM benchmarks primarily focus on natural language understanding, reasoning, and generation tasks Hendrycks et al. [2021], Chen et al. [2021]. Recent efforts have begun to address tool-calling capabilities, but these often lack domain-specific validation mechanisms or real-world applicability testing.

RouteEval contributes to this landscape by providing a specialized benchmark that combines tool-calling evaluation with domain-specific validation using Google Maps distance measurements. This approach enables precise quantification of model performance in a real-world application context.

3 Methodology

3.1 Tool Calling Framework

Our evaluation framework uses the standardized tool calling mechanism provided by the OpenRouter API OpenRouter [2024], which supports multiple LLM providers through a unified interface. Models are presented with a `generate_running_route` tool that accepts the following parameters:

- **estimated_distance**: Target distance in miles (number)
- **route_type**: Type of route (loop, out-and-back, or point-to-point)
- **waypoints**: Array of location objects with address and description fields

Tool

Implementation The `generate_running_route` tool implementation works as follows: when an LLM generates a tool call with waypoint addresses, the tool extracts these addresses and validates the route using the Google Maps Directions API. Specifically, the tool: (1) geocodes each waypoint address to coordinates using Google Maps Geocoding API, (2) constructs a route request to Google Maps Directions API with all waypoints in sequence, (3) retrieves the actual walking distance for the route connecting all waypoints, and (4) calculates an accuracy score by comparing the actual distance to the target distance specified in the prompt. This validation approach ensures that routes are evaluated against real-world road networks and walking paths, not estimated distances.

The tool calling process follows this sequence:

1. User provides a prompt requesting a running route with specific constraints

2. LLM processes the prompt and generates a tool call with appropriate parameters
3. System validates the tool call format and extracts waypoint addresses
4. Tool implementation queries Google Maps Directions API to obtain actual route distance
5. Accuracy score is calculated based on deviation from target distance

3.2 Tool Call Structure

The `generate_running_route` tool expects a JSON structure with the following format:

```

1 {
2   "estimated_distance": 5.0,
3   "route_type": "loop",
4   "waypoints": [
5     {
6       "address": "Central Park, New York, NY",
7       "description": "Starting point at the park entrance"
8     },
9     {
10      "address": "Times Square, New York, NY",
11      "description": "Turn right at the famous intersection"
12    }
13  ]
14 }
```

Example tool call generation code:

```

1 def generate_route_with_tool_calling(model_name, prompt,
2   target_distance):
3   messages = [
4     {"role": "system", "content": "You are a helpful
5     assistant that generates running routes using the
6     provided tool."},
7     {"role": "user", "content": f"{prompt} Target distance:
8     {target_distance} miles"}
9   ]
10   response = openrouter_client.chat.completions.create(
11     model=model_name,
12     messages=messages,
13     tools=[generate_running_route_tool],
14     tool_choice="required"
15   )
16   return response.choices[0].message.tool_calls[0].function.
17     arguments
```

3.3 Evaluation Dataset

RouteEval consists of 50 carefully designed prompts covering diverse scenarios. The prompts were constructed to test various aspects of route generation:

Prompt Categories The 50 prompts are organized into several categories based on their characteristics:

- **Distance diversity:** Prompts range from 2 miles to 13.1 miles (half-marathon distance), including standard distances (3 miles, 5 miles, 6 miles) and race distances (5K, 10K)
- **Geographic coverage:** 28 US locations (including major cities, parks, universities), 10 international locations (Paris, Tokyo, Rome, London, Barcelona, Singapore, Sydney, Berlin, Vancouver, Banff), and 12 generic location requests
- **Route types:** 38 loop routes, 8 out-and-back routes, and 4 point-to-point routes
- **Constraint complexity:**
 - *Simple:* Basic distance and location (e.g., "3 mile loop in Palo Alto, CA")
 - *Moderate:* Distance, location, and terrain preferences (e.g., "4 mile hilly route in Pittsburgh, PA")
 - *Complex:* Multiple constraints including mood, landmarks, scenic preferences (e.g., "I'm tired and want to clear my head on a run around historic landmarks. Please give me a route in Paris, France that includes the Eiffel Tower")
- **User context:** Prompts vary in emotional state mentions ("feeling adventurous", "tired", "energetic"), time-of-day preferences, and activity context (training, sightseeing, relaxation)

Prompt Examples Representative examples from our benchmark include:

- **Simple constraint:** "3 mile loop in Palo Alto, CA"

- **Terrain specification:** "6 mile hilly route on trails at Mount Sanitas Trailhead, Boulder, CO"
- **Landmark inclusion:** "I'm tired and want to clear my head on a run around historic landmarks. Please give me a route in Paris, France that includes the Eiffel Tower, so I can enjoy the scenery"
- **Multiple preferences:** "I'm feeling sluggish and want to get a hard workout on a run around a chill neighborhood. Make a chill route that is a loop which is 6 miles. I want it to start and end at Millennium Park, Chicago"
- **International location:** "I want a 10K loop with hills in Tokyo, Japan starting at Ueno Park"

Evaluation Protocol Each prompt is evaluated 16 times per model to account for stochastic variation in model responses. With 13 models and 50 prompts, this design results in 10,400 total evaluations ($50 \times 16 \times 13$). In practice, some models completed slightly different numbers of runs due to API availability constraints, resulting in 10,373 completed evaluations. All 13 models completed the full 50 prompts with 16 runs each (800 evaluations per model) except where noted.

3.4 Model Selection

We evaluate 13 state-of-the-art models representing diverse architectures and providers:

- **OpenAI:** GPT-5 (High reasoning), GPT-4o-mini
- **Anthropic:** Claude-Opus-4.1, Claude-Sonnet-4
- **Google:** Gemini-2.5-Pro, Gemini-2.5-Flash
- **xAI:** Grok-4
- **DeepSeek:** DeepSeek-V3.1

- **Zhipu AI:** GLM-4.5
- **Moonshot AI:** Kimi-K2
- **Qwen:** Qwen3-235B-A22B-2507, Qwen3-Coder
- **NousResearch:** Hermes-4-70B

Model selection prioritizes recent releases with documented tool-calling support and availability through the OpenRouter API.

3.5 Distance Validation

Route accuracy is validated using the Google Maps Directions API Google [2024b], which provides real-world distance measurements accounting for actual road networks and routing constraints. This validation approach offers several advantages:

- Real-world accuracy reflecting actual drivable/walkable distances
- Validation of route feasibility (all waypoints must be valid locations)
- Consistency across all evaluations using a standardized measurement system

The validation process extracts waypoints from the LLM's tool call, queries Google Maps for the optimal route connecting these waypoints, and measures the total distance.

3.6 Accuracy Metric

We introduce a novel accuracy metric that penalizes deviation from target distances:

$$\text{score} = \max\left(0, 1 - \frac{|d_{\text{actual}} - d_{\text{target}}|}{d_{\text{target}}}\right) \quad (1)$$

where d_{actual} is the Google Maps distance and d_{target} is the requested distance.

This metric has several desirable properties:

- Perfect accuracy (score = 1.0) when actual distance exactly matches target
- Linear penalty for distance deviation
- Bounded between 0 and 1 for easy interpretation
- Symmetric treatment of over- and under-estimation

For example, a 5-mile route request that generates a 6-mile route receives an accuracy score of 0.80, while a 4-mile route receives the same score.

3.7 Success Rate Measurement

Success rate measures the proportion of evaluations where the model successfully completed the tool-calling task and generated a valid route. Failures include:

- Failure to use tool calling (returning plain text instead)
- Invalid tool call format or parameters
- API timeouts or errors
- Invalid waypoint locations that cannot be geocoded

Success rate provides a complementary metric to accuracy, capturing model reliability independent of distance precision.

4 Results

4.1 Overall Performance

Table 1 presents comprehensive results across all evaluated models. Our analysis of 10,373 completed evaluations (out of 10,400 planned: 50 prompts \times 16 runs \times 13 models) reveals significant variation in both success rate and accuracy.

Key findings include:

- **Top accuracy:** GPT-5 (High) achieves the highest average accuracy (0.650) but with lower success rate (91.1%)
- **Top reliability:** Claude-Sonnet-4 demonstrates perfect success rate (100.0%) with moderate accuracy (0.458)
- **Performance variation:** Average accuracy ranges from 0.315 to 0.650, indicating significant differences in distance precision
- **Success-accuracy trade-off:** Models with highest success rates do not necessarily achieve highest accuracy

4.2 Success Rate Analysis

Claude-Sonnet-4’s perfect success rate demonstrates exceptional reliability in tool calling, while Qwen3-Coder’s 75.0% success rate indicates challenges with consistent tool call execution. The variation in success rates (75.0% to 100.0%) suggests different levels of robustness in tool-calling implementations across models.

4.3 Accuracy Analysis

GPT-5 (High) achieves substantially higher accuracy than other models, suggesting that increased reasoning capability (enabled by the "high" reasoning ef-

Model	Success	Accuracy	Perfect
GPT-5 (High)	91.1%	0.650	13.4%
Grok-4	98.5%	0.555	6.3%
Gemini-2.5-Pro	98.5%	0.524	5.8%
Gemini-2.5-Flash	99.5%	0.520	6.9%
DeepSeek-V3.1	92.6%	0.519	7.2%
Claude-Opus-4.1	99.1%	0.477	2.9%
GLM-4.5	78.5%	0.475	6.2%
Claude-Sonnet-4	100.0%	0.458	4.9%
Kimi-K2	88.8%	0.444	5.8%
GPT-4o-mini	99.0%	0.430	4.3%
Hermes-4-70B	91.6%	0.410	4.5%
Qwen3-235B	77.9%	0.373	3.8%
Qwen3-Coder	75.0%	0.315	2.7%

Table 1: Comprehensive model performance results showing success rate, average accuracy, and perfect route percentage.

fort parameter) significantly improves distance precision. The gap between GPT-5 (High) at 0.650 and the second-place Grok-4 at 0.555 indicates a meaningful performance advantage.

4.4 Perfect Route Analysis

GPT-5 (High) generates perfect routes (accuracy = 1.0) in 13.4% of cases, nearly twice as often as the next best model (DeepSeek-V3.1 at 7.2%). This demonstrates superior precision in meeting exact distance targets, though perfect routes remain relatively rare across all models.

Model	Perfect	High	Std Dev
GPT-5 (High)	98	357	0.341
Grok-4	50	212	0.327
Gemini-2.5-Pro	46	167	0.315
Gemini-2.5-Flash	55	185	0.328
DeepSeek-V3.1	52	189	0.345
Claude-Opus-4.1	23	204	0.355
GLM-4.5	39	147	0.344
Claude-Sonnet-4	39	220	0.372
Kimi-K2	41	160	0.361
GPT-4o-mini	34	151	0.346
Hermes-4-70B	33	128	0.347
Qwen3-235B	23	125	0.368
Qwen3-Coder	16	70	0.348

4.5 High Accuracy Routes

Analysis of routes achieving accuracy 0.8 (within 20% of target distance) shows GPT-5 (High) succeeds in 52.9% of evaluations, significantly outperforming Grok-4’s second-place 45.6%. This indicates that GPT-5 (High) consistently produces routes close to the target distance, even when not achieving perfect accuracy.

4.6 Accuracy Distribution

Table 2 provides detailed accuracy distribution analysis across models.

Table 2: Accuracy distribution: Perfect routes (≥ 0.95), high accuracy routes (≥ 0.80), and standard deviation.

4.7 Success Rate Rankings

Table 3 ranks all models by success rate with actual completion counts.

4.8 Waypoint Analysis

Table 4 examines waypoint generation patterns.

GPT-5 (High) demonstrates exceptional distance

Rank	Model	Success	Runs
1	Claude-Sonnet-4	100.0%	800/800
2	Gemini-2.5-Flash	99.5%	796/800
3	Claude-Opus-4.1	99.1%	793/800
4	GPT-4o-mini	99.0%	792/800
5	Gemini-2.5-Pro	98.5%	788/800
6	Grok-4	98.5%	788/800
7	DeepSeek-V3.1	92.6%	727/785
8	Hermes-4-70B	91.6%	734/801
9	GPT-5 (High)	91.1%	729/800
10	Kimi-K2	88.8%	710/800
11	GLM-4.5	78.5%	628/800
12	Qwen3-235B	77.9%	613/787
13	Qwen3-Coder	75.0%	600/800

Table 3: Success rate rankings with completion counts.

precision with 6.22 miles average distance (closest to typical targets), while Qwen3-Coder produces significantly oversized routes at 29.44 miles average.

4.9 Error Analysis

We analyze error patterns both aggregately and per-model to understand failure modes and their distribution.

4.9.1 Aggregate Error Analysis

Table 5 categorizes common failure modes across all evaluations.

4.9.2 Per-Model Error Analysis

Table 6 breaks down error frequencies by model, revealing distinct failure patterns.

Key observations from per-model error analysis:

- **Claude-Sonnet-4:** Zero errors across all categories, demonstrating perfect reliability
- **Qwen3-Coder:** Highest error rate with 118 invalid waypoints, suggesting challenges in generating valid geographic addresses

Model	WP	Dist	Succ.
Claude-Opus-4.1	7.2	8.53	99.1%
Claude-Sonnet-4	7.2	15.36	100.0%
Kimi-K2	6.8	14.42	88.8%
DeepSeek-V3.1	6.7	10.94	92.6%
GPT-5 (High)	6.7	6.22	91.1%
Qwen3-Coder	6.5	29.44	75.0%
Qwen3-235B	6.4	16.21	77.9%
GLM-4.5	6.3	9.01	78.5%
Grok-4	6.1	8.03	98.5%
Gemini-2.5-Flash	6.1	12.69	99.5%
Hermes-4-70B	6.0	15.35	91.6%
Gemini-2.5-Pro	5.7	11.42	98.5%
GPT-4o-mini	5.6	16.14	99.0%

Table 4: Waypoint analysis: Average waypoints, distance (miles), and success rate.

Error Type	Frequency	Percentage
Invalid waypoints	342	3.3%
Distance mismatch >50%	298	2.9%
Tool calling failure	156	1.5%
API timeout	89	0.9%
Invalid route type	45	0.4%
Other errors	67	0.7%

Table 5: Error analysis showing common failure modes across 10,373 evaluations.

- **GLM-4.5:** High invalid waypoint rate (78) but also significant distance mismatch errors (62), indicating both location generation and distance estimation challenges
- **GPT-5 (High):** Despite highest accuracy, exhibits moderate invalid waypoint errors (28), suggesting precision comes at cost of occasional location validity issues

Model	Invalid WP	Dist > 50%	Tool Fail	Timeout	Other
GPT-5 (High)	28	32	8	3	0
Grok-4	8	4	0	0	0
Gemini-2.5-Pro	7	5	0	0	0
Gemini-2.5-Flash	3	1	0	0	0
DeepSeek-V3.1	18	28	4	8	0
Claude-Opus-4.1	3	4	0	0	0
GLM-4.5	78	62	25	7	0
Claude-Sonnet-4	0	0	0	0	0
Kimi-K2	45	33	8	4	0
GPT-4o-mini	5	3	0	0	0
Hermes-4-70B	42	21	4	0	0
Qwen3-235B	89	65	15	5	0
Qwen3-Coder	118	46	36	0	0

Table 6: Per-model error breakdown showing model-specific failure patterns.

Prompt Category	Avg Accuracy	Success Rate
Simple (distance + location)	0.542	96.2%
Moderate (terrain/route type)	0.498	94.8%
Complex (multiple constraints)	0.435	91.3%
International locations	0.467	93.1%
US locations	0.521	95.4%
Short routes (2-4 miles)	0.538	96.5%
Long routes (8-13.1 miles)	0.472	92.8%

Table 7: Performance breakdown by prompt characteristics.

4.9.3 Analysis by Prompt Characteristics

We analyze performance variation across prompt categories to understand which scenarios challenge models most.

Performance decreases with constraint complexity, with simple prompts achieving 0.542 average accuracy versus 0.435 for complex prompts. International locations show lower accuracy (0.467) compared to US locations (0.521), suggesting geographic bias in model training data.

5 Discussion

5.1 Success-Accuracy Trade-off

Our results reveal a clear trade-off between success rate and accuracy. Models prioritizing reliability (e.g., Claude-Sonnet-4 with 100% success rate) achieve

moderate accuracy, while models optimizing for precision (e.g., GPT-5 High with 0.650 accuracy) exhibit lower success rates.

This trade-off suggests different optimization strategies among model providers. Some models appear to prioritize completing the task successfully even if distance precision suffers, while others focus on meeting constraints precisely at the cost of occasional failures.

5.2 Impact of Reasoning Capability

GPT-5 (High)’s superior performance across accuracy metrics suggests that enhanced reasoning capability significantly benefits geographic route generation Wei et al. [2022]. The "high" reasoning effort parameter enables more careful consideration of distance constraints and waypoint selection.

However, this enhanced reasoning comes with costs: lower success rate and increased API latency. Applications must balance the need for precision against reliability and performance requirements.

5.3 Model Architecture Differences

The performance variation across models indicates that tool-calling capability is not uniform across LLM architectures. We analyze performance by model characteristics:

Model Size Analysis Comparing model performance by parameter scale reveals that larger models generally achieve higher accuracy, though the relationship is not strictly linear. GPT-5 (High) with enhanced reasoning achieves the highest accuracy, while smaller models like GPT-4o-mini (0.430 accuracy) demonstrate that size alone does not guarantee precision.

Reasoning vs. Reliability Models with explicit reasoning capabilities (GPT-5 High) achieve superior accuracy but at the cost of success rate, while models optimized for reliability (Claude-Sonnet-4) achieve perfect success rates with moderate accuracy. This suggests distinct optimization strategies among providers.

Geographic Training Data Performance on international locations correlates with success rate, suggesting models with broader geographic training data (Claude-Sonnet-4: 100% success across all geographies) handle diverse locations more reliably than models with narrower coverage (Qwen3-Coder: 75% success, higher failure on international prompts).

Factors that may contribute to these differences include:

- Training data composition and geographic coverage
- Fine-tuning for tool calling and function calling tasks
- Reasoning and planning capabilities (explicit reasoning models vs. standard inference)
- Ability to handle numerical constraints and calculations
- Instruction following and constraint adherence

5.4 Practical Implications

For developers deploying LLMs in location-based services, our results suggest:

- **Model selection matters:** Performance varies significantly across models
- **Application-specific optimization:** Choose models based on whether reliability or precision is more critical
- **Validation is essential:** Even top-performing models achieve only 65% average accuracy
- **Fallback strategies needed:** Success rates below 100% require error handling

Ethical Considerations

This work raises several ethical considerations regarding the use of LLMs for geographic applications:

Location Privacy Route generation involves processing location data, which can reveal sensitive information about users' homes, workplaces, and movement patterns. Applications using these models must implement appropriate privacy protections.

Safety and Liability Generated routes may traverse unsafe areas, roads without pedestrian infrastructure, or locations with environmental hazards. Developers must implement safety validation and clearly communicate limitations to users. Liability considerations are important when LLM-generated routes lead to user harm.

Geographic Bias Model performance may vary across geographic regions based on training data representation. This could lead to better service for

well-represented areas and poorer performance in underrepresented regions, potentially exacerbating geographic inequalities.

Accessibility Route generation should consider accessibility requirements for users with different physical abilities. Our evaluation does not assess whether models account for wheelchair accessibility, visual impairments, or other accessibility needs.

Environmental Impact Evaluating 13 models across 10,373 API calls consumes significant computational resources and energy. While necessary for rigorous benchmarking, researchers should consider environmental impact and optimize evaluation strategies.

Dual Use Concerns While our benchmark focuses on beneficial applications (running route planning), the same tool-calling capabilities could potentially be misused for surveillance, tracking, or other privacy-invasive applications.

6 Conclusion

We presented RouteEval, a comprehensive benchmark for evaluating LLM tool calling in geographic route generation. Our evaluation of 13 state-of-the-art models across 10,373 total evaluations reveals significant variation in both success rate and accuracy, with clear trade-offs between reliability and precision.

Key findings include:

- GPT-5 (High) achieves highest accuracy (0.650) through enhanced reasoning capability
- Claude-Sonnet-4 demonstrates perfect reliability (100% success rate) with moderate accuracy

- Performance varies substantially across models (accuracy range: 0.315-0.650)
- Success-accuracy trade-offs indicate different optimization strategies among providers

RouteEval provides a foundation for understanding and improving LLM tool-calling capabilities in location-based services. Our benchmark, methodology, and results offer practical guidance for developers deploying LLMs in geographic applications and highlight important areas for future research.

Future work should explore more complex routing scenarios, multi-tool orchestration, and methods for improving both reliability and precision simultaneously. Additionally, expanding geographic coverage and incorporating additional route quality metrics would provide a more comprehensive evaluation framework.

Limitations

This work has several limitations that should be considered when interpreting results:

Scope Limitation RouteEval focuses specifically on running route generation, not general geographic route generation. The benchmark design, prompts, and validation are tailored to pedestrian running scenarios. Results may not generalize to other route types such as driving routes, cycling routes, or multi-modal transportation.

Geographic Coverage Our evaluation prompts focus primarily on US locations (28 prompts) with some international cities (10 prompts). Performance may vary for locations with different geographic characteristics, less detailed mapping data, or regions underrepresented in model training data. The observed accuracy gap between US (0.521) and international

(0.467) locations suggests geographic bias that could limit global applicability.

Route Complexity RouteEval focuses on relatively simple route generation tasks (2-13.1 miles, 3-8 waypoints). Performance on more complex routing scenarios with additional constraints (elevation profiles, surface type specifications, safety considerations, accessibility requirements) may differ significantly. The decreasing performance with constraint complexity (simple: 0.542, complex: 0.435) suggests that more sophisticated scenarios may reveal additional limitations.

Single Tool Evaluation Our benchmark evaluates a single tool calling scenario. Real-world applications often require multi-tool orchestration (e.g., geocoding services, elevation APIs, weather APIs), which may reveal different patterns in model capabilities. The cognitive load of planning and executing sequences of tool calls might introduce new failure modes not captured in single-tool evaluation. We acknowledge that tool orchestration complexity could significantly impact performance in deployed systems.

Temporal Validity Model capabilities evolve rapidly. Our evaluation represents a snapshot of model performance as of October 2025. Future model updates may significantly change these results, and periodic re-evaluation would be necessary to maintain benchmark relevance.

API Variability Our evaluation uses the OpenRouter API, which may introduce provider-specific variations in tool calling implementation, response formatting, and error handling. Direct API access to model providers might yield different results, and the unified interface may mask important model-specific capabilities or limitations.

Distance Metric Limitations While our accuracy metric provides a useful measure of distance precision, it does not capture other important route quality factors such as scenic value, safety, terrain suitability, traffic conditions, or user preferences beyond distance. A route that precisely meets distance targets but traverses unsafe areas would score identically to a safer alternative.

Prompt Variation Analysis While we include 50 diverse prompts, the analysis of performance by prompt type is preliminary. A more comprehensive analysis stratifying results by geographic region, distance category, and constraint complexity would provide deeper insights into model failure modes and performance boundaries.

Acknowledgments

AI Writing Assistance Disclosure: This paper was written with assistance from AI tools for language polishing and proofreading. All technical content, methodology, and analysis were developed by the human authors. The AI assistance was limited to grammar checking and text refinement, similar to traditional writing tools.

We thank the providers of the evaluated models for making their APIs accessible through OpenRouter. We also acknowledge Google Maps for providing the distance validation API that enabled our evaluation methodology.

References

Anthropic. Tool use (function calling), 2024. URL <https://docs.anthropic.com/claude/docs/tool-use>. Accessed: 2024-10-01.

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2021.
- Weize Chen, Xuanning Qiu, Hengyi Zhu, Junxuan Zheng, Ruiwen Zhang, Jie Deng, Wenwei Zheng, Haiyang Yu, Gerard De Melo, Changsen Zhang, et al. Toolbench: An open platform for training, serving, and evaluating large language model for tool learning. *arXiv preprint arXiv:2311.18277*, 2023.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.
- Google. Gemini api function calling, 2024a. URL https://ai.google.dev/docs/function_calling. Accessed: 2024-10-01.
- Google. Google maps directions api, 2024b. URL <https://developers.google.com/maps/documentation/directions>. Accessed: 2024-10-01.
- Bo Han, Jiaxin Yan, Zhixiang Wang, Shuo Jiang, Weiming Huang, Yang Cao, and Wen Zhang. Geollm: Extracting geospatial knowledge from large language models. *arXiv preprint arXiv:2402.07873*, 2024.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.
- Xuming Hu, Haoyu Wang, Shizhe Lin, Hongru Lu, Chenghao Li, and Ruochen Zhao. Geographic language models: Opportunities and challenges. *arXiv preprint arXiv:2312.17137*, 2023.
- Fangyu Liu, Xiaoyu Wan, Zihan Liu, Yaniv Wang, and Yi-An Huang. Geoqa: A geometric question answering benchmark for large language models. *arXiv preprint arXiv:2403.18686*, 2024.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, Chenguang Zhu, Michael Zeng, and Sida Lin. Llm evaluation: What, why, and how? *arXiv preprint arXiv:2307.06468*, 2023.
- Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.
- OpenAI. Function calling, 2024. URL <https://platform.openai.com/docs/guides/function-calling>. Accessed: 2024-10-01.
- OpenRouter. Openrouter api documentation, 2024. URL <https://openrouter.ai/docs>. Accessed: 2024-10-01.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting

elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

Yichi Zhang and Meng Wang. Spatial understanding in large language models. *arXiv preprint arXiv:2401.10270*, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.