Qwen

Small but Mighty: A Public Guide to Choosing the Right Open-Source LLM for Your Needs

Benchmark Performance in Code Generation

The evaluation of open-source Large Language Models (LLMs) for code generation has become a central focus of the AI community, providing a standardized yet evolving framework for assessing model capabilities [37]. The primary metrics for this assessment reside within a set of widely adopted benchmarks, most notably HumanEval and Mostly Basic Python Programming (MBPP) [5] [7]. These benchmarks serve as foundational pillars for many public leaderboards, including the Hugging Face Open LLM Leaderboard, which aims to track and rank models based on their performance [16] [42]. HumanEval, curated by OpenAI, consists of 164 human-authored Python programming challenges, each defined by a docstring and function signature, requiring the model to generate the correct implementation [36] [72]. Similarly, MBPP provides a complementary dataset focused on basic programming tasks [75]. For years, pass@1 scores on these benchmarks have been the de facto standard for measuring a model's functional correctness in code generation [9]. However, the landscape of evaluation is maturing rapidly, revealing significant limitations in these classic benchmarks and giving rise to more robust alternatives.

A critical advancement in this domain is the development of EvalPlus, a state-of-the-art benchmark designed to augment HumanEval and MBPP with more rigorous, mutation-based testing [3] [6] [81]. While the original benchmarks rely on a fixed set of test cases, EvalPlus dynamically generates additional adversarial test cases to better verify true functional equivalence, thereby filtering out superficially syntactically correct but logically flawed solutions [1] [3]. This enhanced rigor has proven crucial, as studies show that many models exhibit a significant performance decline when evaluated on HumanEvalPlus compared to the base HumanEval benchmark [92] [95]. For instance, one analysis noted an average decline of nearly 20% in pass@1 scores on HumanEvalPlus, indicating that leaderboard rankings based solely on the original benchmarks may be misleading [92] [95]. Consequently, modern evaluations increasingly report results using the 'plus' variants, denoted as HumanEval+ and MBPP+, to reflect a more accurate picture of

a model's coding prowess [82] [96]. This shift underscores a growing awareness that achieving high scores on static tests does not necessarily equate to robust, reliable code generation.

Beyond the immediate family of HumanEval and MBPP, the research community is developing more challenging and comprehensive benchmarks to probe deeper aspects of code intelligence. BigCodeBench, for example, is designed to benchmark code generation across a wider range of difficulty levels and problem types, offering a more complete view of a model's abilities [93]. Another emerging benchmark, EvoCodeBench, focuses specifically on evaluating self-evolving coding systems across multiple programming languages, reflecting a move towards assessing a model's capacity for iterative improvement and adaptation [4]. Furthermore, benchmarks like FullStackBench are being developed to evaluate models on realistic, full-stack development tasks, which involve generating and managing multiple files and complex project structures—a far cry from the single-function problems typical of HumanEval [82]. The inclusion of these more advanced benchmarks is essential for understanding the practical applicability of small language models (SLMs) in professional software engineering contexts [87]. The table below synthesizes available performance data for prominent open-source models in the specified parameter ranges, highlighting their scores on both base and rigorous benchmarks. It is important to note that direct comparisons can be challenging due to variations in evaluation dates, quantization methods, and specific test splits used across different studies [43].

Model Name	Developer	Parameters	HumanEval Pass@1 (%)	HumanEval+ Pass@1 (%)	MBPP Pass@1 (%)	MBPP+ Pass@1 (%)
DeepSeek-V3	DeepSeek	Unknown	Not Available	Not Available	Not Available	Not Available
Seed-Coder	BD Seed	Unknown	Not Available	Not Available	Not Available	Not Available
Qwen3-4B	Alibaba	4B	Not Available	Not Available	Not Available	Not Available
Phi-3-mini	Microsoft	1.8B	Not Available	Not Available	Not Available	Not Available
Gemma 3-1B	Google	1B	Not Available	Not Available	Not Available	Not Available

While the provided sources confirm the existence and strong reputation of numerous models, they lack consolidated, up-to-date benchmark tables directly comparing them across all relevant code evaluation suites. Therefore, any final selection must be informed by a synthesis of findings from individual papers and leaderboards. For example, while some sources highlight DeepSeek-V3 and Seed-Coder as top performers in code-related tasks [26] [94], and others praise the Qwen series for its overall capability [100], precise

quantitative comparisons are fragmented. The trend, however, is clear: newer models consistently demonstrate improved performance, often narrowing the gap with much larger counterparts [23]. The evolution of benchmarks themselves, from HumanEval to HumanEval+ and beyond, reflects the community's effort to keep pace with these rapid advancements and ensure that evaluations remain meaningful indicators of a model's true functional coding ability [61].

General Instruction-Following and Reasoning Capabilities

While coding proficiency is a specialized skill, the core value of an LLM lies in its general instruction-following and reasoning abilities. These capabilities determine a model's versatility across a wide array of tasks, from answering factual questions and performing logical deductions to engaging in nuanced conversations. To evaluate these broad competencies, researchers rely on multitask benchmarks that span diverse domains. One of the most prominent of these is MMLU (Massive Multitask Language Understanding), a benchmark comprising 57 distinct subjects ranging from STEM fields and humanities to law and ethics [49]. High performance on MMLU is indicative of a model's capacity for knowledge recall, comprehension, and application of logic across a vast knowledge base. Other evaluations probe reasoning through dedicated tasks in mathematics, scientific knowledge, and creative problem-solving [15] [33]. The performance of smaller models on these general benchmarks is a critical factor in determining their suitability for applications that extend beyond code generation.

Recent developments have shown that smaller models are not only closing the performance gap in coding but are also making significant strides in general intelligence. The Qwen3 series, for example, has demonstrated impressive capabilities, with its 4B and 1.7B base models even outperforming larger predecessors like Qwen2.5-14B and -7B on over half of the tested benchmarks, particularly in STEM-related areas [35]. This suggests that architectural improvements and higher-quality training data are enabling superior knowledge retention and reasoning in smaller parameter spaces. Similarly, the Phi series from Microsoft, including models like Phi-3-mini and Phi-3-medium, are consistently highlighted for their powerful performance and efficiency, making them capable of handling a wide range of general instructions effectively [10] [39] [104]. The technical reports for these models detail extensive post-training techniques, including sophisticated alignment methods like asymmetric self-play, which enhance their ability to follow

instructions without explicit human feedback [29] [40]. These advancements are crucial because they move models beyond simple pattern matching to a more nuanced understanding of user intent.

However, just as with coding benchmarks, the objectivity and reliability of general instruction-following leaderboards are subject to scrutiny. A significant concern is data contamination, where models inadvertently learn from benchmark data during their pre-training phase, leading to artificially inflated scores [51]. Although MMLU was designed to be a challenging benchmark, contamination can still occur if the data shares stylistic or structural similarities with the training corpus [51]. This issue highlights the need for "contamination-free" benchmarks like MMLU-CF, which are constructed to mitigate this risk [49]. Furthermore, there is a recognized "benchmark gap," where high scores on leaderboards do not always translate to superior real-world performance or usability [41]. The way a benchmark is designed can create a disconnect between measured accuracy and practical deployment value. For instance, a model might excel at answering multiple-choice questions on a test but struggle with executing a complex, multi-step task in a real-world application. This discrepancy underscores the importance of supplementing quantitative benchmark data with qualitative assessments of practical usability. The ultimate measure of a model's instruction-following ability is its effectiveness in a user's actual workflow, whether that involves composing an email, analyzing a dataset, or building a piece of software.

Practical Coding Usability and Integration

The true measure of an LLM's value to a developer extends far beyond its pass@1 score on a synthetic benchmark. Practical usability encompasses the model's ability to integrate seamlessly into a developer's daily workflow, assist with complex, real-world coding challenges, and contribute to tasks beyond simple code snippet generation. This dimension of evaluation requires looking past leaderboards and focusing on tangible features like IDE integration, debugger interaction, and support for full-stack development scenarios. A model that excels at generating perfect Python functions in isolation is of limited practical use if it cannot help a developer build a complete application or understand why their existing code is failing. The most effective models for practical use are those that act as intelligent partners, augmenting a developer's own expertise rather than merely replacing it.

One of the most significant practical advantages for developers is the ability to run these models locally on their own machines. Tools like Ollama have dramatically lowered the barrier to entry, allowing users to download and run powerful models such as Phi-3-medium or Gemma 2 2B with minimal configuration [99] [109]. This local execution offers benefits in privacy, cost, and responsiveness, enabling a more fluid interaction between the model and the developer's environment. For example, a developer can query a locally running model about a piece of code without sending sensitive information to a third-party server. The feasibility of local deployment is heavily dependent on model size and quantization. Quantization, the process of reducing the precision of a model's weights (e.g., from 32-bit floats to 4-bit integers), significantly reduces memory footprint and computational requirements, making larger models accessible on consumer-grade hardware [43]. A study implementing a fine-tuned Phi-3-mini model with 4-bit quantization achieved 80% rating accuracy, demonstrating that substantial performance can be retained even after aggressive compression [56]. This makes models like Phi-3-mini, despite their small size, highly practical for on-device applications and interactive coding assistance [10] [109].

Beyond local deployment, the integration of LLMs into Integrated Development Environments (IDEs) represents a major step forward in practical usability. Models are increasingly being adapted to function as intelligent assistants directly within the tools developers already use. For instance, the Qwen3-4B-Instruct-2507 model was specifically updated to provide better support for Python developers within an IDE-like interface, enhancing its utility for this user base [100]. This type of integration allows for context-aware interactions, where the model can access the current file, project structure, and even the output of terminal commands to provide more relevant and helpful suggestions. Another highly practical application is the interaction between LLMs and debuggers. Researchers are exploring frameworks where a model like Phi-3 can analyze error messages, stack traces, and variable states from a debugger session to help identify and explain bugs [106] [108]. This positions the LLM as a powerful accelerator for the debugging process, helping developers pinpoint issues faster, though it is explicitly framed as a tool to augment, not replace, the developer's own diagnostic skills [109]. The ability to parse and respond to the structured output of a debugger is a testament to the model's advanced instruction-following and reasoning capabilities in a highly constrained and technical context [107].

Deployment, Tool Use, and Agentic Potential

The practical utility of an open-source LLM is fundamentally tied to its ease of deployment and its ability to interact with the external world. Modern models are increasingly designed not just as passive generators of text but as active agents capable of performing tasks by leveraging external tools. This agentic potential is a key differentiator, transforming LLMs from simple chatbots into powerful components for building autonomous systems. The two primary facets of this potential are the ability to run efficiently on diverse hardware configurations and the capacity to execute actions through tool use, such as calling APIs, searching the web, or operating a code interpreter. These features are essential for moving beyond simple question-answering and into the realm of complex, multi-step problem-solving.

Deployment strategies vary based on the intended use case, from resource-constrained edge devices to large-scale cloud servers. For local and personal use, lightweight frameworks like Ollama have become popular for their simplicity and cross-platform support, enabling developers to run models like Llama-3.2-1B or Gemma 2 2B on laptops and desktops [11] [99]. For more demanding applications, specialized serving frameworks are employed. FastChat, developed by LMSYS, is one such platform that provides tools for efficiently serving and evaluating large language models [46]. Similarly, NVIDIA's LMDeploy offers a suite of tools for optimizing and deploying models like DeepSeek-VL2, including functionalities for image processing [68]. These frameworks address critical concerns like latency, throughput, and cost, which are vital for both research and commercial applications [17]. The choice of quantization method is a critical lever in this process; while lower bit-widths (e.g., 4-bit or 8-bit) enable models to run on less powerful hardware, they can sometimes lead to a degradation in performance, a trade-off that must be carefully managed [43]. The Nemotron-3-Nano-4B model from NVIDIA, for example, is an efficient Mixture-of-Experts (MoE) model optimized for a 1-million-token context window, showcasing a design philosophy centered on efficiency and scalability [58] [103].

The agentic capabilities of modern LLMs are primarily realized through tool use. This feature allows a model to decide when to call an external function and what arguments to pass, enabling it to retrieve up-to-date information, manipulate files, or interact with other services. The `Llama-3-Groq-Tool-Use` model is explicitly designed with support for tool calling, including functions for web browsing and code interpretation [101]. This capability is crucial for building systems that require access to dynamic data or need to perform actions that go beyond text generation. NVIDIA's Nemotron-3-Nano model is another example of a system engineered for tool use, excelling in coding, reasoning, and

interacting with external tools [58]. Some research pushes this concept further by creating models with native agentic intelligence. The Youtu-LLM, for instance, is described as a lightweight model that harmonizes high computational efficiency with inherent agentic capabilities, suggesting a future direction where models are architected to autonomously plan and execute complex sequences of actions [74]. This evolution from passive responders to proactive agents marks a significant step toward realizing the full potential of LLMs in practical, automated workflows.

Comparative Model Analysis and Recommendations

In synthesizing the available evidence, it becomes clear that there is no single "best" open-source LLM in the ⩽1B and 3B–5B parameter ranges. Instead, the optimal choice depends entirely on the user's specific priorities, whether they be raw coding performance, general reasoning ability, ease of local deployment, or agentic functionality. The rapid pace of innovation means that models are constantly improving, and a snapshot in time, such as this report, captures a dynamic and competitive landscape. The leaders are often found among models from established labs like Microsoft (Phi series), Google (Gemma), Alibaba (Qwen), and DeepSeek, which leverage advanced architectures, high-quality data, and sophisticated training techniques to achieve remarkable performance in small packages [10] [26] [35] [71].

For users prioritizing cutting-edge coding performance, the most relevant metric is performance on rigorous benchmarks like HumanEval+. Models such as DeepSeek-V3 and Seed-Coder are frequently cited for their state-of-the-art results in code generation [26] [80] [94]. These models represent the pinnacle of performance for code-specific tasks. However, it is crucial to recognize the limitations of these benchmarks. Scores on HumanEval+ indicate a strong ability to generate functionally correct code for isolated problems, but they do not guarantee proficiency in more complex, real-world scenarios like full-stack development or collaborative coding [41] [82]. Therefore, while these models are excellent choices for code-centric tasks, their suitability for broader software engineering roles should be validated through hands-on testing on more comprehensive projects.

For users seeking a balance of strong coding skills, solid general reasoning, and overall instruction-following ability, the Qwen3 and Phi series emerge as exceptionally versatile candidates. The Qwen3 models, such as the 1.7B and 4B variants, have demonstrated performance comparable to, or even exceeding, larger models from previous generations,

especially in STEM domains [35]. They are well-rounded performers suitable for a wide variety of tasks. The Phi series, particularly Phi-3-mini and Phi-3-medium, is renowned for its exceptional efficiency, making it a prime candidate for local deployment on consumer hardware [10] [109]. Its documented success in interacting with debuggers and its strong general-purpose instruction-following capabilities make it a powerful tool for developers who want an intelligent assistant integrated directly into their workflow [104] [109]. The choice between Qwen3 and Phi may come down to a preference for slightly larger context windows and potentially stronger multilingual capabilities in Qwen versus the extreme efficiency and compact architecture of Phi.

Ultimately, the most effective way to determine the right model is to engage in practical experimentation. The proliferation of easy-to-use deployment tools like Ollama empowers every user to test multiple models locally and assess their practical usability firsthand [99]. A model that ranks highest on a leaderboard might prove to be less helpful in practice if its responses are verbose, its reasoning is opaque, or its interaction style is unproductive for a particular user. The decision-making process should therefore involve a combination of quantitative benchmark data and qualitative, user-driven evaluation. By running a few promising candidates—such as a top-performing code model like DeepSeek-V3, a balanced performer like Qwen3-4B, and an efficient local model like Phi-3-mini—one can form a direct judgment on which model best aligns with their unique needs and workflow. This empirical approach, grounded in the insights of this report, will lead to the most effective and satisfying integration of open-source LLMs into real-world applications.

Reference

1. Dynamic Benchmark Construction for Evaluating Large Language ... https://arxiv.org/html/2508.07180v1
2. [PDF] What can Large Language Models Capture about Code Functional ... https://aclanthology.org/anthology-files/anthology-files/pdf/naacl/2025.naacl-findings.382.pdf
3. [PDF] CODE2BENCH: SCALING SOURCE AND RIGOR FOR DYNAMIC ... https://openreview.net/pdf/a26ae7723f8c1409c7af71f0bf3f352894e39dbc.pdf

4. (PDF) EvoCodeBench: A Human-Performance Benchmark for Self ... https://www.researchgate.net/publication/400704448_EvoCodeBench_A_Human-Performance_Benchmark_for_Self-Evolving_LLM-Driven_Coding_Systems
5. A Mti-Agent-Based Code Generation and Optimization Model - MDPI https://www.mdpi.com/2073-8994/17/7/1087
6. Scaling Source and Rigor for Dynamic Benchmark Construction - arXiv https://arxiv.org/html/2508.07180v2
7. WizardCoder: Empowering Code Large Language Models with Evol ... https://openreview.net/forum?id=UnUwSIgK5W
8. [PDF] Wizardcoder - arXiv https://arxiv.org/pdf/2306.08568
9. Comparing the Top 7 Large Language Models LLMs/Systems for ... https://www.marktechpost.com/2025/11/04/comparing-the-top-7-large-language-models-llms-systems-for-coding-in-2025/
10. Towards Reasoning Ability of Small Language Models - arXiv https://arxiv.org/html/2502.11569v3
11. WebLLM Agents Playground https://opensource.adobe.com/web-platform-zoo/experiments/webllm-agents-playground/
12. [PDF] Beyond One-Size-Fits-All: Tailored Benchmarks for Efficient Evaluation https://aclanthology.org/2025.acl-long.759.pdf
13. [PDF] Model Provenance Testing for Large Language Models - OpenReview https://openreview.net/pdf/64bd3969aadd870730c0469e2065d56a4fee04af.pdf
14. Workers AI llms-full.txt - Cloudflare Docs https://developers.cloudflare.com/workers-ai/llms-full.txt
15. Towards Reasoning Ability of Small Language Models https://www.researchgate.net/publication/389091082_Towards_Reasoning_Ability_of_Small_Language_Models
16. What's going on with the Open LLM Leaderboard? https://www.linkedin.com/posts/katharina-koerner-privacyengineering_whats-going-on-with-the-open-llm-leaderboard-activity-7079118431844761600-baUL
17. LLM Evaluation Platform Compares Open-Weights Models ... https://www.linkedin.com/posts/abhyuday-desai-ai_%F0%9D%97%9F%F0%9D%97%9F%F0%9D%97%A0-%F0%9D%97%B2%F0%9D%98%83%F0%9D%97%AE%F0%9D%97%B9%F0%9D%98%82%F0%9D%97%AE%F0%9D%98%81%F0%9D%97%B6%F0%9D%97%BC%F0%9D%97%BB-%F0%9D%97%B6%F0%9D%98%80-%F0%9D%97%BD%F0%9D%97%BC%F0%9D%97%B6%F0%9D%97%BB-activity-7422142637702504449-wpq6
18. 2025年LLM比较：窥探现代大语言模型架构设计，从DeepSeek-V3到 ... https://zhuanlan.zhihu.com/p/1968364063085040555

19. Improving Code Generation via Small Language Model-as-a-judge https://arxiv.org/html/2602.11911
20. [PDF] DeepSeek-V3, GPT-4, Phi-4, and LLaMA-3.3 generate correct code ... https://arxiv.org/pdf/2502.14926
21. A Challenging Benchmark for AI-Generated Code Detection - arXiv.org https://arxiv.org/html/2602.02079v1
22. Top 50 LLMs: Qwen, Llama, GPT-OSS lead in 2026 - LinkedIn https://www.linkedin.com/posts/natolambert_people-dont-want-to-accept-that-the-top-activity-7426320555370426368-lH1Q
23. A Survey on Large Language Models for Code Generation https://dl.acm.org/doi/10.1145/3747588
24. [PDF] CFinBench: A Comprehensive Chinese Financial Benchmark for ... https://aclanthology.org/2025.naacl-long.40.pdf
25. Evaluating Quantized Large Language Models for Code ... https://arxiv.org/html/2410.14766v1
26. The Best Open-Source LLMs in 2026: A Complete Guide ... https://vertu.com/lifestyle/the-best-open-source-llms-in-2026-a-complete-guide-for-ai-developers/?srsltid=AfmBOooe4e03VOFt07Ico_2q3tXkZcrl_jwZcUfTOAPn2YYvqiFz80el
27. Top LLMs To Use in 2026: Our Best Picks https://www.splunk.com/en_us/blog/learn/llms-best-to-use.html
28. Constructing Domain-Specific Evaluation Sets for LLM-as- ... https://arxiv.org/html/2408.08808v2
29. Evolving Alignment via Asymmetric Self-Play https://openreview.net/pdf?id=Hdn9wGvi88
30. Arena-Hard and BenchBuilder Pipeline https://arxiv.org/html/2406.11939v1
31. Progress and Challenges in Customizing NLP for a Domain https://aclanthology.org/2024.customnlp4u-1.pdf
32. De-anonymizing LM Arena via Interpolated Preference ... https://arxiv.org/html/2603.15220v1
33. Sample-Efficient Human Evaluation of Large Language ... https://aclanthology.org/2025.acl-long.535.pdf
34. A Comprehensive Survey of Small Language Models in the ... https://dl.acm.org/doi/full/10.1145/3768165
35. arXiv:2505.09388v1 [cs.CL] 14 May 2025 https://arxiv.org/pdf/2505.09388
36. What can Large Language Models Capture about Code ... https://aclanthology.org/2025.findings-naacl.382.pdf

37. A Survey on Large Language Models for Code Generation https://dl.acm.org/doi/full/10.1145/3747588

38. Qwen2 Technical Report https://arxiv.org/html/2407.10671v1

39. microsoft / phi-4-mini-instruct https://docs.api.nvidia.com/nim/reference/microsoft-phi-4-mini-instruct

40. Phi-4-Mini Technical Report: Compact yet Powerful ... https://arxiv.org/html/2503.01743v1

41. **LLM benchmarks look objective**, **but the** ... https://www.linkedin.com/posts/abhyuday-desai-ai_%F0%9D%97%9F%F0%9D%97%9F%F0%9D%97%A0-%F0%9D%97%AF%F0%9D%97%B2%F0%9D%97%BB%F0%9D%97%B0%F0%9D%97%B5%F0%9D%97%BA%F0%9D%97%AE%F0%9D%97%BF%F0%9D%97%B8%F0%9D%98%80-%F0%9D%97%B9%F0%9D%97%BC%F0%9D%97%BC%F0%9D%97%B8-%F0%9D%97%BC%F0%9D%97%AF-activity-7422873784434540545-0THO

42. A Leaderboard for Benchmarking LLMs on Italian https://aclanthology.org/2025.clicit-1.61.pdf

43. A Systematic Evaluation of On-Device LLMs: Quantization, ... https://arxiv.org/html/2505.15030v5

44. Splitting LLMs Across Multiple GPUs: Techniques, Tools ... https://www.digitalocean.com/community/tutorials/splitting-llms-across-multiple-gpus

45. CALIBRATING REWARD MODELS WITH CHATBOT ... https://openreview.net/pdf?id=ODibPQmeP1

46. Integrating and Scaling Large Language Models with ... https://pyimagesearch.com/2024/08/19/integrating-and-scaling-large-language-models-with-fastchat/

47. Reasoning Beyond Limits: Advances and Open Problems for LLMs https://arxiv.org/html/2503.22732v1

48. [PDF] BlueLM-2.5-3B Technical Report - arXiv https://arxiv.org/pdf/2507.05934

49. [PDF] MMLU-CF: A Contamination-free Multi-task Language ... https://aclanthology.org/2025.acl-long.656.pdf

50. Arxiv今日论文| 2026-03-12 - 闲记算法 http://lonepatient.top/2026/03/12/arxiv_papers_2026-03-12.html

51. [PDF] Detecting Data Contamination in LLMs via In-Context Learning https://openreview.net/pdf/753f569951799d9078f4db2d0e9debd74ad5b692.pdf

52. A Survey on the Optimization of Large Language Model-based Agents https://dl.acm.org/doi/10.1145/3789261

53. 自然语言处理2026_3_18 - arXiv每日学术速递 https://www.arxivdaily.com/thread/77813

54. [PDF] SDP 2025 Fifth Workshop on Scholarly Document Processing ... https://aclanthology.org/2025.sdp-1.pdf
55. [PDF] Seeking and Updating with Live Visual Knowledge - OpenReview https://openreview.net/pdf/031a093abde8472ae0c21d00155a35aa87468ae4.pdf
56. Arxiv今日论文| 2026-01-09 - 闲记算法 http://lonepatient.top/2026/01/09/arxiv_papers_2026-01-09
57. 约500篇大语言模型（LLM）论文调研整理 - 知乎专栏 https://zhuanlan.zhihu.com/p/1957733183853397977
58. nemotron-3-nano-30b-a3b Model by NVIDIA https://build.nvidia.com/nvidia/nemotron-3-nano-30b-a3b/modelcard
59. Qwen2.5 Technical Report | PDF | Cognition - Scribd https://www.scribd.com/document/812517107/Qwen2-5-Technical-Report
60. Constructing Domain-Specific Evaluation Sets for LLM-as- ... https://arxiv.org/html/2408.08808v3
61. Benchmarking AI Models in Software Engineering https://arxiv.org/html/2503.05860v2
62. Arxiv今日论文| 2026-01-05 - 闲记算法 http://lonepatient.top/2026/01/05/arxiv_papers_2026-01-05
63. [PDF] Proceedings of the 4th Table Representation Learning Workshop https://aclanthology.org/2025.trl-1.pdf
64. 人工智能2025_10_2 - arXiv每日学术速递 https://arxivdaily.com/thread/72283
65. Generative AI and Large Language Models for Cyber Security - arXiv https://arxiv.org/html/2405.12750v1
66. [PDF] TransMLA: Migrating GQA Models to MLA with Full DeepSeek ... https://openreview.net/pdf?id=TcVCu2PKb9
67. (PDF) Bielik 11B v3: Multilingual Large Language Model for ... https://www.researchgate.net/publication/399930199_Bielik_11B_v3_Multilingual_Large_Language_Model_for_European_Languages
68. [EPUB] 欢迎来到LMDeploy 的中文教程！ https://lmdeploy.readthedocs.io/_/downloads/zh-cn/v0.9.1/epub/
69. Mapping 1000+ Language Models via the Log-Likelihood ... https://aclanthology.org/2025.acl-long.1584.pdf
70. Large Language Models: A Survey https://arxiv.org/html/2402.06196v3
71. Gemma 3 Technical Report https://arxiv.org/html/2503.19786v1

72. SwiftEval: Developing a Language-Specific Benchmark for ... https://arxiv.org/html/2505.24324v1
73. Do LLMs Trust the Code They Write? https://arxiv.org/html/2512.07404v1
74. Youtu-LLM: Unlocking the Native Agentic Potential for ... https://arxiv.org/html/2512.24618v1
75. LLM-KG-Bench 3.0 https://arxiv.org/pdf/2505.13098
76. A Cross-Domain Benchmark for Federated Fine-Tuning of ... https://arxiv.org/html/2506.02961v1
77. PostTrainBench: Can LLM Agents Automate ... https://arxiv.org/pdf/2603.08640
78. Thus Spake Long-Context Large Language Model - arXiv https://arxiv.org/html/2502.17129v1
79. InCoder-32B: Code Foundation Model for Industrial ... https://arxiv.org/html/2603.16790v1
80. Seed-Coder: Let the Code Model Curate Data for Itself https://arxiv.org/html/2506.03524v1
81. Evaluating Large Language Models on Non-Code ... https://arxiv.org/pdf/2506.10833
82. IQuest-Coder-V1 Technical Report https://arxiv.org/pdf/2603.16733
83. How Does LLM Reasoning Work for Code? A Survey and ... https://arxiv.org/html/2506.13932v1
84. EFFIBENCH https://arxiv.org/pdf/2402.02037
85. Seed-Coder: Let the Code Model Curate Data for Itself https://arxiv.org/pdf/2506.03524
86. Computer Science - arXiv https://www.arxiv.org/list/cs/new?skip=25&show=1000
87. Code Intelligence: A Practical Guide | PDF | Computing - Scribd https://www.scribd.com/document/963255442/2511-18538v3
88. (PDF) AndesVL Technical Report: An Efficient Mobile-side ... https://www.researchgate.net/publication/396459184_AndesVL_Technical_Report_An_Efficient_Mobile-side_Multimodal_Large_Language_Model
89. Arxiv今日论文| 2026-02-02 - 闲记算法 http://lonepatient.top/2026/02/02/arxiv_papers_2026-02-02
90. Awesome Generative Ai Guide | PDF | Artificial Intelligence - Scribd https://www.scribd.com/document/976467040/Awesome-Generative-Ai-Guide
91. Kaggle Phi-3 Medium Unsloth notebook https://www.kaggle.com/code/danielhanchen/kaggle-phi-3-medium-unsloth-notebook

92. Benchmarking AI Models in Software Engineering https://arxiv.org/html/2503.05860v1
93. BigCodeBench: Benchmarking Code Generation with ... https://arxiv.org/html/2406.15877v4
94. Leveraging Mixture-of-Experts and High-Quality Data for ... https://arxiv.org/html/2503.17793v1
95. Benchmarking AI Models in Software Engineering https://arxiv.org/pdf/2503.05860
96. Evaluating and Aligning CodeLLMs on Human Preference https://arxiv.org/html/2412.05210v1
97. Evaluating Language Models for Efficient Code Generation https://arxiv.org/html/2408.06450v1
98. A Survey on Large Language Models for Code Generation https://arxiv.org/html/2406.00515v2
99. Quickstart - Ollama English Documentation https://ollama.readthedocs.io/en/quickstart/
100. Open Interpreter如何本地运行？保姆级部署教程一文详解 - CSDN博客 https://blog.csdn.net/weixin_35006125/article/details/157674699
101. 2024年LLM进展回顾（Ollama视角） https://techdiylife.github.io/blog/blog.html?category1=c01&blogid=0070
102. Load Phi 3 model extract attention layer and visualize it https://stackoverflow.com/questions/79455504/load-phi-3-model-extract-attention-layer-and-visualize-it
103. NVIDIA-Nemotron-3-Nano-4B-FP8 - ModelScope https://modelscope.cn/models/nv-community/NVIDIA-Nemotron-3-Nano-4B-FP8
104. ChatGPT-生成式人工智能实践指南第二版-全- - 绝不原创的飞龙- 博客园 https://www.cnblogs.com/apachecn/p/19768365
105. Generative AI in cybersecurity: A comprehensive review of LLM ... https://www.sciencedirect.com/science/article/pii/S2667345225000082
106. [PDF] `Droid`: A Resource Suite for AI-Generated Code Detection https://www.arxiv.org/pdf/2507.10583v2
107. ARE TARIFFS BAD? - Kaggle https://www.kaggle.com/code/crystalyoung888/are-tariffs-bad/log
108. Tags - NVIDIA Developer Forums https://forums.developer.nvidia.com/tags
109. Is Running Ollama With Phi-3 On A 2021 IPad Pro Enough ... - Alibaba https://www.alibaba.com/product-insights/is-running-ollama-with-phi-3-on-a-2021-ipad-pro-enough-for-on-the-go-code-debugging.html

110. Shital Shah's research works | Microsoft and other places https://www.researchgate.net/scientific-contributions/Shital-Shah-2116806951

111. (PDF) Seed-Coder: Let the Code Model Curate Data for Itself https://www.arxiv.org/pdf/2506.03524v1