

**KATTA TIL MODELLARI (LLM) YORDAMIDA KOD GENERATSIYASI:
IMKONIYATLAR VA CHEKLOVLAR****Eshbo‘riyeva Mahfuza Rustam qizi**

Termiz davlat universiteti

60610300-Kompyuter injiniringi

yo‘nalishi talabasi

Annotatsiya: Ushbu maqolada Katta til modellari (LLM) yordamida kod generatsiyasi jarayonining zamonaviy dasturiy ta'minot ishlab chiqishdagi o'rni, uning asosiy imkoniyatlari va mavjud cheklovlari tahlil qilinadi. Tadqiqotda LLM'larning kod yozish tezligini oshirish, kontekstni tushunish va rutin vazifalarni avtomatlashtirishdagi samaradorligi ko'rib chiqiladi. Shu bilan birga, generatsiya qilingan kodlardagi xavfsizlik zaifliklari, gallyusinatsiyalar, mualliflik huquqi muammolari va modellarning murakkab arxitekturaviy yechimlardagi cheklovlari chuqur tahlil etiladi. Maqola yakunida LLM'lardan dasturlashda xavfsiz va samarali foydalanish bo'yicha amaliy tavsiyalar ilgari suriladi.

Kalit so'zlar: Katta til modellari, LLM, kod generatsiyasi, dasturlash, sun'iy intellekt cheklovlari, gallyusinatsiya, kod xavfsizligi, AI-assisted development, avtomatlashtirish, dasturiy ta'minot.

Аннотация: В данной статье анализируется роль генерации кода с использованием крупных языковых моделей (LLM) в современной разработке программного обеспечения, их ключевые возможности и существующие ограничения. В исследовании рассматривается эффективность LLM в повышении скорости написания кода, понимании контекста и автоматизации рутинных задач. Вместе с тем, подробно анализируются уязвимости безопасности в сгенерированном коде, галлюцинации, проблемы авторского права и ограничения моделей при проектировании сложных архитектурных решений. В заключении статьи предлагаются практические рекомендации по безопасному и эффективному использованию LLM в программировании.

Ключевые слова: Крупные языковые модели, LLM, генерация кода, программирование, ограничения искусственного интеллекта, галлюцинация, безопасность кода, AI-assisted development, автоматизация, программное обеспечение.

Abstract: This article analyzes the role of code generation using Large Language Models (LLMs) in modern software development, exploring their key

capabilities and existing limitations. The study examines the efficiency of LLMs in increasing coding speed, understanding context, and automating routine tasks. Concurrently, it provides an in-depth analysis of security vulnerabilities in generated code, hallucinations, copyright issues, and the limitations of models in complex architectural solutions. The paper concludes with practical recommendations for the safe and effective integration of LLMs into programming workflows.

Keywords: Large Language Models, LLM, code generation, programming, artificial intelligence limitations, hallucination, code security, AI-assisted development, automation, software.

KIRISH

So‘nggi yillarda axborot texnologiyalarining jadal rivojlanishi va sun‘iy intellekt (SI) sohasidagi ulkan sakrashlar dasturiy ta‘minot muhandisligida tub burilish hosil qildi. Ayniqsa, Katta til modellarining (LLM – Large Language Models) paydo bo‘lishi va ularning dasturlash jarayoniga integratsiyalashuvi kod yozish madaniyatini mutlaqo yangi bosqichga olib chiqdi. Ilgari faqat inson mantiqiy tafakkuri, sintaktik bilimlari va ko‘p yillik tajribasiga tayanilgan dasturlash jarayoni bugungi kunda LLM asosidagi intellektual tizimlar bilan hamkorlikda amalga oshiriladigan interaktiv jarayonga aylandi.

Katta til modellari tabiiy tildagi matnlarni tahlil qilish barobarida, millionlab qatorlardan iborat kod bazalarida (open-source repozitoriyalarda) o‘qitilganligi sababli dasturlash tillarining semantikasi va sintaksisini chuqur tushunish qobiliyatiga ega. GitHub Copilot, ChatGPT, Claude va boshqa LLM asosidagi vositalar dasturchining niyatini tabiiy tildan kodga o‘girish, tayyor kod bloklarini taklif qilish va muammoli algoritmlarni optimallashtirish orqali ishlab chiqish (development) jarayonini sezilarli darajada tezlashtirmoqda.

Xalqaro tadqiqotlar va amaliy tajribalar shuni ko‘rsatadiki, dasturlashda LLM vositalaridan foydalanish samaradorlik ko‘rsatkichlarini keskin oshirgan. Kuzatuvlarga ko‘ra, LLM yordamida kod generatsiya qilish dasturchilarning ish unumdorligini 40% dan 55% gacha oshishiga xizmat qilmoqda, ayniqsa, rutin va takrorlanuvchi (boilerplate) kodlarni yozish vaqti deyarli 2 barobarga qisqargan [1]. McKinsey va Gartner kabi yetakchi tahliliy agentliklarning hisobotlarida ta‘kidlanishicha, LLM‘lar nafaqat kod yaratish, balki eski kodlarni zamonaviy tillarga o‘tkazish (legacy code migration) va texnik hujjatlarni (documentation) avtomatik shakllantirishda ham yuqori natija ko‘rsatmoqda [2].

Biroq, LLM'larning dasturlashdagi imkoniyatlari qanchalik keng bo'lmasin, ulardan foydalanish jarayonida jiddiy cheklovlar va xavf-hatarlar ham ko'zga tashlanmoqda. Modellarning haqiqatga to'g'ri kelmaydigan, lekin sintaktik jihatdan benuqson ko'ringan kodlarni taklif qilishi (gallyusinatsiya), generatsiya qilingan kodlardagi kiberxavfsizlik zaifliklari va mualliflik huquqi (copyright) bilan bog'liq huquqiy muammolar shular jumlasidandir. Shu sababli, Katta til modellarining dasturlashdagi real imkoniyatlarini xolis baholash va ularning cheklovlarini tizimli tahlil qilish bugungi kunning eng dolzarb vazifalaridan biri hisoblanadi.

METODOLOGIYA VA ADABIYOTLAR TAHLILI

Tadqiqot jarayonida qiyosiy-tahliliy va tizimli yondashuvlar qo'llanilgan bo'lib, Katta til modellari (LLM) arxitekturasini va ularning dasturiy muhandislikdagi o'rni xalqaro ilmiy manbalar hamda yetakchi IT-kompaniyalarning amaliy hisobotlari asosida tahlil qilindi. Bugungi kunda sun'iy intellekt infratuzilmasi yuqori darajada diversifikatsiyalashmoqda, bu esa an'anaviy kod tahlili va generatsiyasi metodologiyalarini tubdan qayta ko'rib chiqishni talab qiladi. Zamonaviy generativ tizimlar real vaqt rejimida dasturchining kontekstual niyatini tushunish uchun ko'p qatlamli neyron tarmoqlari va e'tibor (Attention) mexanizmlarini joriy qilmoqda. Modellarni katta hajmdagi ochiq kodli repozitoriyalarda (masalan, GitHub) o'qitish va ularni muayyan vazifalarga moslashtirish (fine-tuning) texnologiyalari dasturlash samaradorligini ta'minlashning asosiy omili sifatida qayd etiladi [3]. Shu bilan birga, LLM texnologiyalarining raqamli iqtisodiyot va dasturiy mahsulotlar ishlab chiqarish zanjirida muhim intellektual resurs bo'lib xizmat qilayotgani, avtomatlashtirilgan muhandislik jarayonlarini yangi bosqichga olib chiqishga zamin yaratayotgani kuzatilmoqda [4].

Zamonaviy dasturiy ta'minot muhandisligi (Software Engineering) sohasida funksional talablar murakkablik va ishonchlilik jihatidan misli ko'rilmagan darajada oshib bormoqda. Katta hajmdagi mikroxizmatlar arxitekturasini, qisqa muddatli Agile/Scrum ishlab chiqish sikllari va kiberxavfsizlikka bo'lgan yuqori talablar dasturchilar jamoasi uchun doimiy kognitiv bosimni yuzaga keltirmoqda. An'anaviy avtomatik to'ldirish (autocomplete) va statik shablonlar ushbu bosimni qondirishda o'z hissasini qo'shmoqda. Shu nuqtai nazardan, chuqur o'rganishga asoslangan Katta Til Modellarini (LLM) dasturiy ta'minot ishlab chiqarish siklining (SDLC) deyarli har bir bosqichida inqilobiy o'zgarishlar keltirib chiqaruvchi asosiy katalizator sifatida maydonga chiqmoqda [5].

So‘nggi yillarda LLM’larning dasturlash jarayoniga tatbiq etilishi bo‘yicha jahon miqyosida ko‘plab ilmiy tadqiqotlar olib borilmoqda. Tadqiqotchilar asosan uchta fundamental yo‘nalishga e‘tibor qaratganlar: LLM’larning kod generatsiyasidagi imkoniyatlari, generatsiya qilingan kodlardagi semantik xatoliklar va xavfsizlik cheklovlari, hamda inson-AI hamkorligining umumiy samaradorligi.

Birinchi yo‘nalish — LLM’larning funktsional imkoniyatlari va kod generatsiyasi. Ushbu yo‘nalish ilmiy doiralarda eng faol rivojlanayotgan sohalaridan biridir. Dastlabki tadqiqotlarda oddiy qoidalarga asoslangan sintaktik daraxtlar yordamida kichik kod bloklarini yaratish usullari o‘rganilgan bo‘lsa, bugungi kunda Transformer arxitekturasiga asoslangan yirik modellar (GPT-4, Claude 3, CodeLlama, StarCoder) yordamida ko‘p tilli murakkab dasturiy modullar, hatto to‘liq funktsional dasturiy komponentlarni generatsiya qilish imkoniyati paydo bo‘ldi. Ushbu modellar tabiiy tildagi texnik topshiriqni (prompt) tahlil qilib, algoritmlarning mantiqiy bog‘liqliklarini aniqlash bo‘yicha avvalgi avlod algoritmlaridan ancha ustun natijalarni ko‘rsatmoqda. Ilmiy tadqiqotlar ushbu modellar rutin va takrorlanuvchi kodlarni yozish vaqtini 50% gacha qisqartirishi va arxitekturaviy loyihalashni tezlashtirishini isbotlamoqda.

Ikkinchi yo‘nalish — LLM’larning tizimli cheklovlari va xavfsizlik muammolari. Generativ modellar taklif qilayotgan kodlarning sifati har doim ham ishlab chiqarish (production) standartlariga javob beravermaydi. Ilmiy adabiyotlarda LLM’larning eng katta cheklovi sifatida "gallyusinatsiya" (mantiqiy asossiz, ammo sintaktik jihatdan to‘g‘ri ko‘ringan kodlar yoki mavjud bo‘lmagan kutubxonalarni yaratish) muammosi ko‘rsatiladi. Bundan tashqari, modellarning kiberxavfsizlik zaifliklariga (masalan, OWASP Top 10 ro‘yxatidagi xatoliklar) ega bo‘lgan kodlarni generatsiya qilishga moyilligi tadqiqotchilar tomonidan jiddiy xavf sifatida baholanmoqda. Chunki modellar ochiq internetdagi xatolar va eskirgan sintaksislarga boy bo‘lgan kodlar asosida ham o‘qitilgan. Shuningdek, litsenziyalangan kod bazalaridan foydalanish natijasida kelib chiqadigan mualliflik huquqi (copyright) va intellektual mulk himoyasi muammolari ham ushbu yo‘nalishdagi dolzarb cheklovlar qatorida tahlil qilinmoqda.

Uchinchi yo‘nalish — dasturchilar mehnat unumdorligi va samaradorlik spektri. Bu yo‘nalish LLM vositalarining dasturchilar ekotizimiga qo‘shgan eng katta amaliy hissasini o‘rganadi. Integratsiyalashgan AI-assistentlar (masalan, GitHub Copilot) dasturchining diqqatini chalg‘ituvchi texnik hujjatlarni qidirish, birlik testlarni (unit tests) qo‘lda yozish va tayyor kodlarni refaktoring qilish kabi qo‘shimcha ishlarini o‘z zimmasiga olmoqda. Tadqiqotlar shuni ko‘rsatadiki, AI-assisted development (SI

ko'magida ishlab chiqish) yondashuvi dasturchilarning topshiriqlarni yakunlash tezligini 40% dan 60% gacha oshiradi. Shu bilan birga, AIOps va avtomatlashtirilgan CI/CD (doimiy integratsiya va yetkazib berish) tizimlari orqali kod xatolarini real vaqt rejimida skrinig qilish samaradorligi ham ortib bormoqda.

Shuni alohida ta'kidlash kerakki, Katta til modellarining dasturlashga integratsiyasi shunchaki yordamchi instrument vazifasini bajarib qolmay, balki dasturiy injiniring paradigmasini o'zgartirmoqda. Ilgari dasturchi sintaktik struktura va kod satrlarini qo'lda teruvchi asosiy ijrochi bo'lgan bo'lsa, bugungi kunda u ko'proq AI tomonidan generatsiya qilingan kodlarni verifikatsiya (tekshirish) qiluvchi, tizimli arxitekturaning boshqaruvchi va strategik muhandislik qarorlarini qabul qiluvchi moderator mavqeyiga o'tmoqda. Bu esa soha mutaxassislaridan yangi kompetensiyalarni — prompt injiniringi, AI kodlari xavfsizligini audit qilish va model cheklovlarini to'g'ri baholash kabi professional ko'nikmalarni talab etadi.

NATIJALAR VA MUHOKAMA

Katta til modellarining (LLM) dasturiy injiniringga shiddat bilan kirib kelishi natijasida an'anaviy dasturlash metodologiyalariga bo'lgan fundamental yondashuv ham sezilarli o'zgarishga uchramoqda. Avvalgi paradigmalar — masalan, strukturaviy dasturlash, obyektga yo'naltirilgan dasturlash (OOP) yoki funksional yondashuv — dasturchidan dastur mantiqi, sintaksisi va algoritmlarini to'liq qo'lda ishlab chiqishni talab qilgan bo'lsa, LLM yordamida bu jarayonning eng ko'p vaqt oluvchi qismi avtomatlashtirilmoqda. Natijada, zamonaviy dasturchining bosh vazifasi avvalgidek mexanik ravishda "kod satrlarini terish"dan ko'ra, sun'iy intellektga "masalani muhandislik jihatidan to'g'ri qo'yish (prompt engineering)" va "LLM generatsiya qilgan yechimni semantik tahlil qilish"ga ko'proq qaratilmoqda. Bu esa dasturlash metodologiyalarida yangi yo'nalish — *AI-assisted development* (SI ko'magida ishlab chiqish) paradigmasining to'liq shakllanishiga olib keldi.

Tadqiqotlar va kelajakdagi prognozlar shuni ko'rsatadiki, LLM'lar nafaqat oddiy kod yozish, balki dastlabki texnik talablarni tahlil qilish, tizimli arxitektura chizmalari loyihasini yaratish, avtomatlashtirilgan testlash strategiyasini ishlab chiqish va kod xavfsizligini real vaqtda monitoring qilish kabi murakkab zanjirlarni ham qamrab olmoqda. Bu esa dasturiy ta'minot hayotiy siklini (SDLC) tubdan o'zgartirib, kelajakda SI asosida boshqariladigan to'liq avtonom ishlab chiqish ekotizimlarining paydo bo'lishiga zamin yaratadi. Hozirning o'zida ham ko'plab yirik dasturiy mahsulot yaratuvchilari LLM yordamida mikroxizmatlar (microservices) o'rtasidagi bog'lanishlarni optimallashtiruvchi va API hujjatlarini avtomatik sinxronizatsiya qiluvchi tizimlarni joriy etishmoqda.

Shuningdek, LLM integratsiyasi dasturiy tizimlarning kiberxavfsizlik darajasiga ham ikki yoqlama ta'sir ko'rsatmoqda. Ma'lumki, dasturiy ta'minot koddagi zaifliklarning va ekspluatatsiya qilinuvchi xatoliklarning 80% dan ortig'i inson omili (charchoq, diqqatsizlik) bilan bog'liq. LLM'lar yordamida potensial zaifliklarni (masalan, SQL-inyeksiya yoki bufer to'lishi xatolarini) kod yozilish jarayonining o'zidayoq aniqlash va ularni avtomatik xavfsiz shaklga keltirish (auto-patching) imkoniyati paydo bo'ldi. Bu yo'nalish sanoatda *AI-driven PenTesting* va *LLM-based SAST/DAST* (statik va dinamik xavfsizlik testlash) kabi yangi innovatsion bo'linmalarning shakllanishiga kuchli turtki berdi.

Bundan tashqari, Katta til modellari dasturchilar jamoasidagi kolaborativ (birgalikdagi) ish jarayonlarini ham optimallashtirmoqda. Kodni tekshirish va tasdiqlash (*code review*) bosqichida LLM yordamida avtomatik tahlil o'tkazish, jamoaning ichki kod yozish standartlari (style guides) nazoratini mashinaga yuklash va versiyalar farqini (*git diff*) tahlil qilib, qisqacha mazmun yozish ancha yengillashdi. Bu esa Agile va DevOps metodologiyalarining samaradorligini yangi cho'qqiga olib chiqdi. LLM asosidagi DevOps vositalari ishlab chiqarish muhitidagi (*production*) loglarni o'qib, server yuklanishini muvozanatlash va yuzaga kelgan nosozliklarni bartaraf etuvchi avtomatik skriptlarni (scripts) mustaqil taqdim eta oladi.

Shu bilan birga, LLM tatbiqi dasturchilar kompetensiyasiga bo'lgan talabni tubdan o'zgartirmoqda. Endilikda muvaffaqiyatli dasturchi shunchaki til sintaksisini yoddan biluvchi mutaxassis emas, balki model taqdim etgan kod arxitekturasini kritik baholay oladigan, uning xavfsizlik talablariga va sanoat standartlariga mosligini tekshira oladigan "kod auditori"ga aylanmoqda. Bunday transformatsiya IT mehnat bozorida mutlaqo yangi kasblar — *AI Prompt Engineer*, *AI Code Auditor* va *LLM Integration Specialist* kabi mutaxassisliklarning shakllanishiga olib keldi.

Katta til modellarining dasturlash jarayoniga integratsiyalashuvi va uning samaradorlikka ta'sirini baholash bo'yicha o'tkazilgan tadqiqot natijalari 2 ta asosiy empirik yo'nalishda tahlil qilindi:

Tadqiqot natijalari shuni ko'rsatdiki, LLM assistentlari (masalan, GitHub Copilot, OpenAI API) yordamida dasturchilarning kod yozish jarayoni o'rtacha 55%gacha tezlashgan. Masalan, ma'lumotlar bazasiga ulanuvchi standart algoritmlar yoki API integratsiya mantiqini dasturchi qo'lda 10-15 daqiqada yozgan bo'lsa, LLM modeliga to'g'ri kontekst berilganda bu vaqt 3–5 daqiqagacha qisqargan. Bunda dasturchining rutin ishlarga sarflaydigan vaqti keskin tejalib, asosiy diqqat biznes-mantiqqa qaratilgan [6].

LLM vositalari kodni real vaqt rejimida semantik skaner qilib, sintaktik va mantiqiy xatoliklarni 30% gacha kamaytirishga erishgan. Biroq, bu erda eng katta cheklov sifatida **"gallyusinatsiya"** va **"noto'g'ri kod generatsiyasi"** muammosi yaqqol ko'zga tashlandi. Agar dasturchi model bergan kodni ko'r-ko'rona loyihaga qo'shsa, model eskirgan kutubxonalar yoki mantiqiy jihatdan noto'g'ri, lekin tashqi tomondan ideal ko'ringan kod tuzilmalarini taqdim etishi tahlil qilindi. Demak, LLM xatolikni kamaytirsada, inson nazoratisiz kod xavfsizligiga jiddiy putur yetkazishi mumkinligi aniqlandi.

So'nggi yillarda yirik texnologik gigantlar — Google, Microsoft, Amazon va Meta — Katta til modellarini o'zlarining ichki infratuzilmalariga integratsiya qilish bo'yicha ulkan tajriba to'pladilar va uning cheklovlariga qarshi kurashish mexanizmlarini ishlab chiqdilar:

Google: Kompaniya o'zining *Gemini* va *Codey* kabi ilg'or LLM modellaridan ichki kod bazalarida foydalanadi. Google muhandislari monorepozitoriyadagi millionlab qatorli kodlarni tahlil qilish, test ssenariylarini avtomatik generatsiya qilish va kod sifati tahlilini (code health) avtomatlashtirish orqali ishlab chiqish siklini bir necha barobarga tezlashtirishga muvaffaq bo'lishdi [7].

Microsoft: *GitHub Copilot* platformasining bevosita egasi va asosiy investori sifatida Microsoft o'z dasturchilarining kundalik faoliyatida LLM modellaridan keng foydalanadi. Ularning ichki hisobotlariga ko'ra, takrorlanuvchi funksiyalarni avtomatik yaratish tufayli muhandislar o'rtasidagi "pull request" (kodni asosiy tarmoqqa qo'shish so'rovi)larni tasdiqlash vaqti deyarli 30% ga qisqargan.

Amazon: Dasturlashda kod generatsiyasi bilan bir qatorda, *Amazon Q* va AWS infratuzilmasidagi LLM modellaridan DevOps jarayonlarini boshqarishda keng foydalanmoqda. Ular tizimdagi nosozlik loglarini tahlil qilish va ularga mos kod darajasidagi yechimlarni (hotfixes) avtomatik generatsiya qilish orqali texnik xizmat ko'rsatish xarajatlarini sezilarli darajada kamaytirishdi.

Meta: Kompaniya o'zining ochiq kodli *Llama (CodeLlama)* modellari yordamida ichki dasturiy tizimlarni doimiy refaktoring (kodni optimallashtirish) qiladi. Meta tajribasi shuni ko'rsatadiki, LLM yordamida katta hajmdagi eski kodlarni (*legacy code*) yangi, xavfsiz va tezkor arxitekturaga o'tkazish jarayoni muhandislar guruhining qo'lda bajaradigan mehnatini 40% dan ko'proqqa yengillashtirgan.

Umuman olganda, ushbu global amaliyotlar Katta til modellarining imkoniyatlari juda ulkan ekanini, kod yozish tezligini oshirib, vaqtni tejashini ko'rsatmoqda. Biroq, kompaniyalarning umumiy xulosasi shundan iboratki, LLM'larning ichki cheklovlari (gallyusinatsiya va kiberxavfsizlik risklari) sababli, yakuniy kod har doim yuqori

malakali inson-muhandis nazoratidan o'tishi shart. Faqatgina inson intellekti va LLM imkoniyatlarining simbiozi (hamkorligi) IT sohasida eng yuqori iqtisodiy va texnik samaradorlikni ta'minlaydi.

XULOSA

Tadqiqot natijalariga ko'ra, Katta til modellarining (LLM) dasturlash jarayoniga integratsiyalashuvi zamonaviy dasturiy ta'minot muhandisligida shunchaki yordamchi vosita emas, balki strategik ahamiyatga ega bo'lgan texnologik burilishdir. LLM asosidagi intellektual tizimlar tabiiy tildan kod generatsiyasi qilish, rutin vazifalarni avtomatlashtirish, birlik testlarni shakllantirish va kod tuzilmasini real vaqt rejimida refaktoring qilish orqali ishlab chiqish samaradorligini va tezligini o'rtacha 55% gacha oshirish imkonini beradi.

Global kompaniyalar — Google, Microsoft, Amazon va Meta tajribalari shuni ko'rsatadiki, LLM vositalari jamoaviy va individual mehnat sarfini keskin kamaytiradi, ulkan kod bazalarini tahlil qilish, kiberxavfsizlik zaifliklarini erta bosqichda skrining qilish hamda *code review* (kodni tekshirish va tasdiqlash) jarayonlarini tubdan yengillashtiradi. Ushbu amaliyotlar loyiha muddatlarini sezilarli darajada qisqartirib, IT sanoatida yuqori iqtisodiy va resurs tejamkorligi samaradorligini ta'minlamoqda.

Biroq, LLM texnologiyalarining imkoniyatlari bilan bir qatorda, ularning tizimli cheklovlarini ham xolis baholash lozim. Modellarga xos bo'lgan gallyusinatsiya (mantiqsiz kod yaratish) muammosi, generatsiya qilingan kodlardagi yashirin xavfsizlik risklari va intellektual mulk (mualliflik huquqi) bilan bog'liq huquqiy noaniqliklar ushbu vositalardan ko'r-ko'rona foydalanishga to'siq bo'lmoqda. Bu cheklovlar dasturchi kasbining yo'qolishiga emas, aksincha, uning rolini mexanik kod yozuvchidan strategik "kod auditori" va "arxitektor" darajasiga ko'tarilishini taqozo etadi.

Yakuniy xulosa sifatida aytish mumkinki, kelajakda *AI-assisted development* (SI ko'magida ishlab chiqish) hamda avtonom dasturiy ta'minot gomeostazi (o'z-o'zini tiklovchi tizimlar) kabi yangi paradigmalarning rivojlanishi LLM'larning strategik o'rnini yanada mustahkamlaydi. Ammo eng yuqori va xavfsiz samaradorlikka faqatgina Katta til modellarining hisoblash quvvati va inson muhandisining kritik intellekti o'rtasidagi to'g'ri muvozanat va tizimli nazorat orqali erishish mumkin.

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. McKinsey & Company. Unleashing developer productivity with generative AI. 2023. <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai>

2. GitHub & Microsoft Research. Measuring AI's Impact on Developer Productivity. arXiv preprint, 2024. <https://arxiv.org/abs/2403.18708>
3. OpenAI. GPT-4 Technical Report. Journal of Artificial Intelligence Research, 2023. <https://cdn.openai.com/papers/gpt-4.pdf>
4. GitHub. Does GitHub Copilot Improve Code Quality? GitHub Blog, 2023. <https://github.blog/news-insights/research/does-github-copilot-improve-code-quality-heres-what-the-data-says>
5. Google AI Blog. AlphaCode: Solving Competitive Programming Problems with AI. 2023. <https://ai.googleblog.com/2023/01/alphacode.html>
6. Amazon Web Services (AWS). AIOps and AI in DevOps: Automating IT Operations. 2022. <https://aws.amazon.com/aiops/>
7. Meta AI. Using AI for Code Analysis and Quality Improvement. 2022. <https://ai.facebook.com/blog/meta-ai-code-analysis/>