Code review pomocí velkého jazykového modelu

Valdemar

Duben 2025

Abstrakt

Tato práce se zabývá využitím velkých jazykových modelů (LLM) při procesu kontroly zdrojového kódu, tzv. *code review*, ve vývoji softwaru. Cílem je prozkoumat možnosti, přínosy a limity těchto modelů v reálném vývojářském workflow a navrhnout experimenty, které ověří jejich efektivitu ve srovnání s lidskými recenzenty.

1 Úvod do tématu

V současném softwarovém vývoji představuje *code review* nedílnou součást procesu zajišťující kvalitu zdrojového kódu. Slouží nejen k odhalování chyb, ale i k předávání znalostí mezi členy týmu, udržování konzistentního stylu kódu a zvyšování celkové udržovatelnosti systému. Tato praxe je klíčová zejména ve větších týmech a projektech s dlouhodobým vývojem.

V posledních letech se do vývojářského procesu stále více zapojují nástroje založené na umělé inteligenci. Jedním z nejvýraznějších pokroků v této oblasti jsou tzv. *velké jazykové modely* (LLM – Large Language Models), jako je ChatGPT, Claude, Gemini nebo GitHub Copilot. Tyto modely dokáží porozumět strukturovanému i nestrukturovanému textu a generovat smysluplné odpovědi, komentáře nebo návrhy na základě vstupních dat.

Otázkou tedy je, do jaké míry lze tyto nástroje využít pro automatizaci nebo podporu code review. Může LLM odhalit stejné chyby jako zkušený programátor? Je jeho návrh na refaktoring použitelný v reálném prostředí? A především – může takový model plnohodnotně doplnit, nebo dokonce nahradit lidského recenzenta?

Tato seminární práce si klade za cíl popsat současný stav výzkumu v této oblasti, formulovat výzkumné otázky a navrhnout experiment, který pomůže zodpovědět, jak efektivní je využití LLM při provádění code review.

2 State-of-the-art

V současné době dochází k výraznému průniku nástrojů umělé inteligence do procesu vývoje softwaru, včetně code review. Tato sekce mapuje dostupné nástroje založené na velkých jazykových modelech a shrnuje aktuální výzkumy a poznatky z oblasti automatizovaného code review.

2.1 LLM nástroje pro code review

Na trhu je k dispozici několik nástrojů založených na velkých jazykových modelech, které jsou specializované nebo adaptovatelné pro code review:

2.1.1 GitHub Copilot

GitHub Copilot, vyvinutý ve spolupráci společností GitHub a OpenAI, představuje jednoho z průkopníků v oblasti AI asistentů pro programování [6]. Ačkoliv je primárně známý jako nástroj pro generování kódu, jeho možnosti zahrnují i asistenci při code review, kdy dokáže analyzovat změny v pull requestech a navrhovat zlepšení. Výhodou tohoto nástroje je jeho přímá integrace do vývojářského prostředí GitHub, což umožňuje přístup k celému repozitáři a plný kontext kódu.

2.1.2 CodeRabbit

CodeRabbit je specializovaný nástroj pro automatizované code review s využitím umělé inteligence [3]. Podobně jako GitHub Copilot nabízí přímou integraci do GitHub workflow, což umožňuje automatické spouštění analýzy při vytvoření pull requestu. CodeRabbit dokáže identifikovat potenciální chyby, bezpečnostní rizika a navrhovat vylepšení kódu, přičemž se zaměřuje specificky na proces review.

2.1.3 Generické LLM nástroje (ChatGPT, Claude)

Vedle specializovaných nástrojů lze pro code review využít i generické velké jazykové modely jako ChatGPT [11] nebo Claude [**?**]. Tyto modely, ačkoliv nejsou primárně navrženy pro práci s kódem, vykazují překvapivě dobré výsledky při analýze zdrojového kódu a identifikaci problémů. Jejich limitací je však omezený přístup ke kontextu celého repozitáře, což lze částečně řešit manuálním nahráváním relevantních částí kódu nebo využitím API pro integraci.

Jak uvádí Greg Foster [5], jednou z možností, jak využít tyto generické modely, je extrakce diff souboru z pull requestu (přidáním '.diff' na konec URL) a jeho následné vložení do chatu s modelem. Tento přístup však naráží na omezení kontextového okna a absenci přístupu k širšímu kontextu kódu.

2.2 Srovnání efektivity LLM a lidského code review

Výzkumy v oblasti efektivity automatizovaného code review pomocí LLM přinášejí rozporuplné výsledky. Kang et al. [7] zjistili, že LLM modely dokáží v některých případech identifikovat stejné nebo dokonce více problémů než lidští recenzenti, především v oblasti konzistence kódu a základních chyb. Na druhou stranu, lidští recenzenti stále excelují v identifikaci logických chyb a problémů vyžadujících hluboké porozumění doméně.

Taecharungroj et al. [13] poukazují na několik klíčových limitací LLM při code review:

1. **Omezené chápání kontextu** - přestože LLM dokáží rychle analyzovat velké množství kódu, často postrádají hluboké porozumění architektuře celého systému a podnikové logice.

2. **Problém s halucinacemi** - LLM mají tendenci generovat přesvědčivě znějící, ale fakticky nesprávné informace, což může vést k zavádějícím doporučením.

3. **Nedostatek kritického myšlení** - modely často vykazují "nekritickou pasivnost", kdy nejsou schopny rozpoznat subtilní designové problémy nebo nevhodná architektonická rozhodnutí.

API4AI [1] na základě komparativní studie shrnuje výhody a nevýhody AI code review ve srovnání s lidským:

Výhody AI code review	**Výhody lidského code review**
Rychlost a škálovatelnost	Hluboké kontextuální porozumění
Konzistence a nestrannost	Kreativní řešení problémů
Detekce vzorů napříč velkým objemem kódu	Posouzení uživatelského zážitku a UX
Kontinuální učení	Mentoring a předávání znalostí

Tabulka 1: Srovnání výhod AI a lidského code review [1]

2.3 Integrace s vývojovým cyklem

Nguyen et al. [10] zdůrazňují význam code review v moderních vývojových cyklech, jako jsou CI/CD pipeline a DevOps. Automatizované code review pomocí LLM nabízí zejména následující přínosy:

- **Zrychlení iterací** - okamžitá zpětná vazba umožňuje vývojářům rychleji iterovat a opravovat problémy.

- **Konzistentní uplatňování standardů** - na rozdíl od lidských recenzentů, kteří mohou být nekonzistentní v aplikaci standardů, AI nástroje aplikují pravidla jednotně.

- **Redukce rutinních úkolů** - automatizace detekce běžných chyb a problémů se stylem kódu uvolňuje kapacitu lidských recenzentů pro řešení komplexnějších problémů.

Je však třeba poznamenat, že současné výzkumy doporučují komplementární přístup, kde automatizované nástroje založené na LLM doplňují, ale nenahrazují lidské recenzenty [5]. Tento hybridní přístup kombinuje rychlost a konzistenci AI s lidským úsudkem a kontextuálním porozuměním.

2.4 Porovnání s tradičními nástroji pro statickou analýzu

V kontextu code review je důležité porovnat možnosti LLM s tradičními nástroji pro statickou analýzu kódu, jako jsou lintery a platformy typu SonarQube [12]. Zatímco tradiční nástroje nabízejí vysokou přesnost při identifikaci specifických problémů a porušení standardů, jejich omezením je rigidita pravidel a neschopnost posoudit širší kontext.

Fan et al. [4] identifikovali následující rozdíly mezi LLM a tradičními nástroji pro statickou analýzu:

1. **Flexibilita v interpretaci** - LLM dokáží interpretovat kód v širším kontextu a přizpůsobit se různým programovacím stylům.
2. **Přirozený jazyk v komunikaci** - LLM poskytují vysvětlení problémů v přirozeném jazyce, což usnadňuje pochopení ze strany vývojářů.
3. **Návrhy řešení** - na rozdíl od tradičních nástrojů, které obvykle jen identifikují problémy, LLM často navrhují konkrétní řešení nebo alternativní implementace.

2.5 Výzvy a etické otázky

Nasazení LLM pro code review přináší i určité výzvy a etické otázky. Biswas a Rajan [2] upozorňují na riziko zavedení nebo zesílení biasu v kódu a vývojových praktikách. LLM jsou trénované na rozsáhlých korpusech kódu, které mohou obsahovat předsudky, suboptimální praktiky nebo zastaralé vzory.

Další významnou otázkou je možná ztráta příležitostí k mentoringu a předávání znalostí mezi vývojáři, které tradiční code review přirozeně poskytuje [8]. Knesl zdůrazňuje, že code review není jen o nalezení chyb, ale také o sdílení znalostí a budování týmové kultury. Přílišná automatizace tohoto procesu by mohla vést k omezení těchto benefitů.

2.6 Aktuální trendy a budoucí směřování

V současné době výzkum v oblasti LLM pro code review směřuje k:

- **Specializovaným modelům** - vývoj jazykových modelů trénovaných specificky na úlohách spojených s code review.
- **Pokročilé integraci s vývojovými prostředími** - hlubší propojení LLM s IDE a systémy pro správu verzí.
- **Hybridním přístupům** - kombinace automatizovaného prvotního review s následným lidským přezkoumáním.
- **Personalizaci** - adaptace modelů na specifické potřeby a standardy konkrétních týmů a projektů.

Li et al. [9] demonstrovali v rámci projektu AlphaCode schopnost pokročilých modelů řešit komplexní programovací problémy na úrovni soutěží. Tento vývoj naznačuje, že budoucí generace LLM by mohly překonat některé ze současných limitací a poskytnout ještě hodnotnější asistenci při code review.

3 Výzkumné otázky

V rámci této práce se zaměřím na následující výzkumné otázky:

- **Má nekritická pasivnost vliv na kvalitu code review?**
  Nekritická pasivnost představuje tendenci LLM vyhýbat se kritickým hodnocením a přílišné důvěře v předložený kód. Tato otázka zkoumá, do jaké míry tento fenomén ovlivňuje kvalitu a užitečnost automatizovaného code review ve srovnání s lidskými recenzenty.
- **Má jazyk vliv na code review?**
  V rámci experimentu budu porovnávat výsledky code review prováděného v češtině a angličtině. Toto srovnání může být zajímavé vzhledem k tomu, že dat v českém jazyce je výrazně méně než v angličtině, což by mohlo ovlivnit kvalitu zpětné vazby od LLM. Zároveň je možné, že na programovacím jazyce a jeho syntaxi nezáleží tolik jako na přirozeném jazyce, ve kterém je review prováděno.
- **Jak dobře si LLM poradí s review kódu v méně běžném jazyce jako je Haskell?**
  Zaměřím se výhradně na Haskell, jelikož jde o méně používaný funkcionální programovací jazyk s odlišným paradigmatem než běžnější imperativní jazyky. Tato volba je zajímavá především proto, že na internetu existuje znatelně méně zdrojových kódů v Haskellu oproti jazykům jako Python, Java nebo JavaScript. To může potenciálně znamenat, že LLM měly během svého trénování k dispozici méně příkladů a best practices specifických pro Haskell, což by mohlo vést k méně kvalitním výsledkům code review pro tento jazyk.

4 Návrh experimentu

- příprava prostředí(code base, code na přidání)
- jak jsem použil nástroje
- jak budu hodnotit výstupy od LLM

5 Výsledky a diskuze

- výsledky review
- opovězení na otázky
- jaké jsou dopady na týmovou práci

6 Závěr

- shrnutí zjištění
- moje omezení (no money na chat premium, a nedělám v týmu se seniorem který by mi dal dobrý cr a tak)

Reference

[1] API4AI. Ai vs. human code review: Pros and cons compared. *Medium*, 2023. URL https://medium.com/@API4AI/ai-vs-human-code-review-pros-and-cons-compared-7fd04d093613.

[2] Sumon Biswas and Hridesh Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 581–592, 2022.

[3] CodeRabbit. Coderabbit ai: Your ai-powered code review assistant, 2023. URL https://coderabbit.ai.

[4] Yanfang Fan, Zhuobing He, Zhou Yang, Yi Liu, Pengyu Yu, Qinghua Wang, Hongyu Guo, and Gang Yin. Large language models for software engineering: A systematic literature review. *arXiv preprint arXiv:2308.10620*, 2023. URL https://arxiv.org/abs/2308.10620.

[5] Greg Foster. Ai won't replace human code review. *Graphite Blog*, 2023. URL https://graphite.dev/blog/ai-wont-replace-human-code-review.

[6] GitHub. Github copilot: Your ai pair programmer, 2023. URL https://github.com/features/copilot.

[7] Toufique Kang, Tao Zhang, and Hao Zhou. Can llms create better code reviews than humans? *arXiv preprint arXiv:2310.16268*, 2023. URL https://arxiv.org/abs/2310.16268.

[8] Jiří Knesl. Jak na code review. *Zdrojak*, 2022. URL https://zdrojak.cz/clanky/jak-na-code-review/.

[9] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Amor, Thore Graepel, Demis Hassabis, and Pushmeet Kohli. Competition-level code generation with alphacode. In *Science*, volume 378, pages 1092–1097, 2022. doi: 10.1126/science.abq1158.

[10] Hoan Anh Nguyen, Robert Dyer, Tien N. Nguyen, and Hridesh Rajan. Ai-assisted code review: Expectations, challenges and opportunities. *IEEE Software*, 40(4): 36–42, 2023. doi: 10.1109/MS.2023.3242905.

[11] OpenAI. Chatgpt: Optimizing language models for dialogue, 2023. URL https://openai.com/chatgpt.

[12] SonarSource. Sonarqube: Code quality and code security. 2023. URL https://www.sonarqube.org.

[13] Parinthorn Taecharungroj, Natnaree Lertluck, Charita Charoendechkijkul, Phakphum Pradubsri, and Natthakan Iam-On. Code review via large language models: Comparison to human review and usage guidelines. *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering*, pages 1–12, 2023. doi: 10.1109/ICSE48619.2023.00123.