Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2024 **Milan Horínek**

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 24. října 2023

. . . . . .
Milan Horínek

PODĚKOVÁNÍ

ANOTACE

Autor se zabývá problematikou přistávání raket a tím, že většina prací na toto téma používá model RL. Autor se však rozhodl použít pro řízení dopředný výpočet přistávací trajektorie. Autor stručně popisuje dvě metody řízení, které byly testovány - zpětnou vazbu a numerické řešení ze simulace. Bylo zjištěno, že druhá metoda je přesnější, ale první je vhodnější pro model 3DoF. Autor navrhuje možná rozšíření práce, například přidání rušivých vlivů, jako je vítr, a zvýšení robustnosti a přesnosti řídicího systému.

Klíčová slova: raketa, přistání, VTVL, pinpoint landing

ABSTRACT

The author discusses the problem of rocket landing, and how most works on the topic use the RL model. However, the author decided to use forward calculation of the landing trajectory for control. The author briefly describes two control methods that were tested – feedback and numerical solution from the simulation. The latter was found to be more accurate, but the former was more suitable for the 3DoF model. The author suggests possible extensions for the work, such as adding disturbances such as wind, and increasing the robustness and accuracy of the control system.

Key words: raketa, přistání, VTVL, pinpoint landing

ZADÁNÍ

1. Prozkoumejte dostupné práce na téma řízení polohy rakety a přistávacího manévru
2. Navrhněte zjednodušený model rakety včetně potřebných příslušných technických prostředků (aktuátorů, senzorů, atd.)
3. Navrhněte algoritmus řízení letu rakety včetně potřebných letových módů
4. Implementujte model a regulační smyčku ve vybraném simulačním prostředí
5. Zhodnoťte získané výsledky

ASSIGNMENT

1. Explore available papers on rocket attitude control and landing maneuver
2. Create a simplified model of rocket including the necessary technical means (actuators, sensors, etc.)
3. Design a rocket flight control algorithm including the necessary flight modes
4. Implement the model and control loop in the chosen simulation environment
5. Evaluate the obtained results

Obsah

Slovník pojmů

LLM Velký jazykový model (LLM - large language model) je typ modelu strojového učení, který je navržen tak, aby rozuměl a generoval text v přirozeném jazyce. Tyto modely jsou trénovány na obrovských datasetech a jsou schopny provádět různé úkoly, jako je textová klasifikace, generování textu, strojový překlad a další. 7

testové pachy Unit Test Smells jsou špatné návyky nebo postupy, které se mohou objevit při psaní jednotkových testů. Tyto ”smelly” často vedou k neefektivním nebo nespolehlivým testům a mohou ztížit údržbu testovacího kódu. Příklady zahrnují Duplicated Asserts, Empty Tests a General Fixture. 7

1 Úvod

2 Provedená práce v problematice

2.1 Vydané publikace

Jeden z poměrně nedávno vydaných článků (září 2023) nazvaný „An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation“ [1] se zabývá využitím velkých jazykových modelů (LLM) pro automatizované generování jednotkových testů v jazyce JavaScript. Implementovali nástroj s názvem **TESTPILOT**, který využívá LLM *gpt3.5-turbo*, *code-cushman-002* od společnosti OpenAI a také model StarCoder, který vznikl jako komunitní projekt [2]. Vstupní sada pro LLM obsahovala signatury funkcí, komentáře k dokumentaci a příklady použití. Nástroj byl vyhodnocen na 25 balíčcích npm obsahujících celkem 1684 funkcí API. Vygenerované testy dosáhly pomocí gpt3.5-turbo mediánu pokrytí příkazů 70,2% a pokrytí větví 52,8%, čímž překonaly nejmodernější techniku generování testů v jazyce JavaScript zaměřenou na zpětnou vazbu, Nessie.

Zmíněný model *StarCoder* byl představen v článku „StarCoder: may the source be with you!“ [2] z května 2023. Vytvořeny byly konkrétně 2 verze, *StarCoder* a *StarCoderBase*, s 15,5 miliardami parametrů a délkou kontextu 8K. Tyto modely jsou natrénovány na datové sadě nazvané *The Stack*, která obsahuje 1 bilion tokenů z permisivně licencovaných repozitářů GitHub. *StarCoderBase* je vícejazyčný model, který překonává ostatní modely open-source LLM modely, zatímco *StarCoder* je vyladěná verze speciálně pro Python, která se vyrovná nebo překoná stávající modely zaměřené čistě na Python. Článek poskytuje komplexní hodnocení, které ukazuje, že tyto modely jsou vysoce efektivní v různých úlohách souvisejících s kódem.

Článek „Exploring the Effectiveness of Large Language Models in Generating Unit Tests“ [3] z dubna 2023 hodnotí výkonnost tří LLM - *Codex*, *CodeGen* a *GPT-3.5* - při generování jednotkových testů pro třídy jazyka Java. Studie používá jako vstupní sady dva benchmarky, HumanEval a Evosuite SF110. Klíčová zjištění ukazují, že *Codex* dosáhl více než 80% pokrytí v datové sadě *HumanEval*, ale žádný z modelů nedosáhl více než 2% pokrytí v benchmarku *SF110*. Kromě toho se ve vygenerovaných testech často objevovaly tzv. testové pachy, jako jsou *duplicitní tvrzení* a *prázdné testy* [4].

2.1.1 Srovnání výsledků

2.2 Modely

Jedním z často používaných modelů v předchozích pracích je *StarCoder* a *StarCoderBase*, diskutovaný již v sekci 2. *Base* verze je schopna generovat kód pro více jak 80 programovacích jazyků. Model je navržen pro širokou škálu aplikací obsahující *generování*, *modifikaci*, *doplňování* a *vysvětlování* kódu. Jeho distribuce je volná a licence **CodeML OpenRAIL-M 0.1** [5] umožňuje ho využívat pro množšství aplikací včetně komerčních nebo edukačních. Jeho uživatel však má povinnost uvádět, že výsledný kód byl vygenerován modelem. Licence

má své restrikce z obavy tvůrců, protože by model mohl někoho při nepsrávném použití ohrozit. Tyto restrikce se aplikují na na všechny derivace projektů pod touto licencí. Zároveň není kompatibilní s Open-Source licencí právě kvůli těmto restrikcím.

Nedávno vydaným modelem je *Code Llama* od společnosti Meta. Jedná se o evoluci jejich jazykového modelu *Llama* specializovaný však čistě na úlohy kódování. Je postaven na platformě *Llama 2* a existuje ve třech variantách: *základní Code Llama*, *Code Llama - Python* a *Code Llama - Instruct*. Model podporuje více programovacích jazyků, včetně jazyků jako Python, C++, Java, PHP, Typescript, C nebo Bash. Je určen pro úlohy, jako je generování kódu, doplňování kódu a ladění. Code Llama je zdarma pro výzkumné i komerční použití a je uvolněn pod licencí MIT. Uživatelé však musí dodržovat zásady přijatelného použití, ve kterýcg je uvedeno, že model nelze použít k vytvoření služby, která by konkurovala vlastním službám společnosti Meta.

Velmi populárním nástrojem pro generování kódu za pomocí LLM je GitHub Copilot, který je postaven na modelu *codex* od OpenAI. Původní model však byl přestal být zákazníkům nabízen a namísto něj OpenAI doporučuje ke generování kódu využívat chat verze modelů GPT-3.5 a GPT-4. Na architektuře GPT-4 je také postavený nástupce služby Copilot, Copilot X. Zmíněné modely chat GPT-3.5 a GPT-4 jsou primárně určeny pro generování textu formou chatu. Zvládají však zároveň i dobře generovat kód a jsou vhodné i úlohu generování jednotkových testů. Narozdíl od předchozích modelů však nejsou volně distribuovány a jsou poskytovány pouze jako služba společností OpenAI skrze API nebo je lze hostovat v rámci služby Azure společnosti Microsoft, která zajišťuje větší integritu dat. Jedná se tedy o uzavřený model a jeho uživatelé musí souhlasit s jeho podmínkami použití.

2.2.1 Srovnání modelů

Práce	Model	Benchmark	Pokrytí testy	Úspěšnost
An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation	*gpt-3.5-turbo*	Sada NPM balíčků	70.2%	48%
	code-cushman-002		68.2%	47.1%
	StarCoder		54%	31.5%
Exploring the Effectiveness of Large Language Models in Generating Unit Tests	*gpt-3.5-turbo*	HumanEval	69.1%	52.3%
		SF110	0.1%	6.9%
	CodeGen	HumanEval	58.2%	23.9%
		SF110	0.5%	30.2%
	Codex (4k)	HumanEval	87.7%	76.7%
		SF110	1.8%	41.1%
Java Unit Testing with AI: An AI-Driven Prototype for Unit Test Generation	*gpt-3.5-turbo*	JUTAI - Zero-shot, temperature: 0	84.7%	71%

Tabulka 1: Přehled a srovnání studií

Model	Otevřenost	Licence	Počet parametrů	Počet programovacích jazyků	Datum vydání
StarCoderBase	Volně dostupný	CodeML OpenRAIL-M 0.1	15.5 miliardy	80+	5/2023
Code Llama	Volně dostupný	Llama 2 Licence	34 miliard	8+	8/2023
gpt-3.5-turbo	Uzavřený	Proprietární	175 miliard?	?	5/2023
gpt-4	Uzavřený	Proprietární	1.76 bilionu	?	3/2023

Tabulka 2: Přehled a srovnání modelů generujících kód

Reference

[1] M. Schafer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," *arXiv preprint arXiv:2302.06527*, 2023.

[2] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, *et al.*, "Starcoder: may the source be with you!," *arXiv preprint arXiv:2305.06161*, 2023.

[3] M. L. Siddiq, J. C. S. Santos, R. H. Tanvir, N. Ulfat, F. A. Rifat, and V. C. Lopes, "Exploring the effectiveness of large language models in generating unit tests," *arXiv preprint arXiv:2305.00418*, 2023.

[4] "Test smells." `https://testsmells.org/pages/testsmells.html`, 2021. Čteno: 23.10.2023.

[5] B. Project, "Codeml openrail-m 0.1 license," 2023. Čteno: 23.10.2023.