LLM gestütztes Debugging von Workflows

Exposé zur Bachelorarbeit

Alexandria Arnold

Institut für Informatik
Wissensmanagement in der Bioinformatik
Humboldt-Universität zu Berlin

Gutachter: Prof. Dr. Ulf Leser

18. März 2024

Inhaltsverzeichnis

1 Einleitung

Sprachmodelle oder auch Large Language Models (LLMs) die Welt der KI und des natural language processing (NLP) maßgeblich verändert [1]. So beantworten Sie beispielsweise bereits umfangreiche Benutzeranfragen, korrigieren und gestalten Texte und führen menschlich anmutende Konversationen [2].
Auch im Bereich der Informatik finden sich zahlreiche Anwendungsgebiete. Hier helfen LLMs beispielsweise bei der Dokumentation von Code [3], beim Deuten von Fehlermeldungen [4] und generieren sogar eigenen Programmcode [5]. Gerade beim Debugging kommt es häufig vor, dass Fehlermeldungen nicht aussagekräftig genug sind und sich somit von Entwicklern schwer deuten lassen, Fehlerquellen, sogenannte Bugs, schwer identifizierbar sind und die Fehlerbehebung teilweise einen großen Mehraufwand an Zeit und Arbeit bedeutet. Sprachmodelle können dabei helfen, Fehlermeldungen zu verstehen und dem Entwickler in verständlicher Form zu vermitteln und Fehler im Programmcode eigenständig zu identifizieren und zu berichtigen [6].
Von dieser Unterstützung würden auch Entwickler und Nutzer von wissenschaftlichen Workflows, die in sogenannten wissenschaftlichen Workflow-Management-Systemen, kurz Workflow-Systeme, definiert sind, profitieren. Workflow-Systeme bieten die Möglichkeit, einzelne Datenverarbeitungsaufgaben in einer Pipeline zu binden und die Datenanalyse so zu vereinheitlichen und zu erleichtern. [7]
Doch während bezüglich populärer Programmiersprachen wie beispielsweise Python [8] oder Java [9], zu diesen Themen bereits intensive Forschung betrieben wird, gibt es auf dem Gebiet von Workflow-Systeme diesbezüglich bisher nur wenig Forschung. Auch lassen sich die Ergebnisse und Erkenntnisse von gebräuchlichen Programmiersprachen kaum oder nur teilweise auf Workflow-Systeme übertragen. Das liegt daran, dass bei der Entwicklung von Workflows meist externe Tools benutzt und, selbst für den Entwickler, schlecht erkennbare Kontrollstrukturen verwendet werden [10]. Somit besitzen Workflow-Systeme auch andere Fehlerquellenarten als klassische Programmiersprachen.
Das Debugging von Workflows ist aus vielen Gründen herausfordernd. Zuerst erschweren die je nach Workflow-System unterschiedlichen Domain-Specific Languages, die Suche nach vergleichbaren Code-Beispielen. Auf der wohl populärsten Plattform zum Austausch von Problemen und Fehlern in Programmen, Stack Overflow, finden sich aktuell lediglich ca. 430 Einträge mit dem Tag "nextflow" und ca. 1720 Einträge mit dem Tag "snakemake" . Zum Vergleich: populäre Sprachen wie beispielsweise Python und JavaScript kommen jeweils auf über 2 Mio. Einträge [11]. Dies liegt natürlich vor allem an einer deutlich kleineren Benutzerzahl, bietet somit jedoch auch nur einen kleinen Wissenspool an potenziell vergleichbaren Fehlern und deren Lösungen, was das Identifizieren und Beheben von Bugs deutlich erschwert. Zudem handelt es sich bei Entwicklern und Benutzern von Workflows eher selten um hauptberufliche Informatiker. Die fehlende Expertise kann den Code somit fehleranfälliger machen und Fehlermeldungen können von Entwicklern und Benutzern schwerer zu deuten sein.
Die Arbeit soll eine Einsicht in die Debugging- und Support-Fähigkeiten von LLMs bezüglich in Nextflow verfasste wissenschaftliche Workflows bieten. Besonders Nutzer und Entwickler von Workflow-Systemen können stark von einem User-Support durch einen auf künstlicher Intelligenz basiertem Chatbot profitieren. Da Workflow-Systeme vorrangig von Datenwissenschaftlern und -analysten verwendet und entwickelt werden, bedeutet das oft, dass diese nicht zwangsläufig eine informationstechnische Ausbildung besitzen [12]. Somit ist es fachfremden Benutzern überlassen, die Sprache, Programme und Funktionen von individuellen Workflow-Systemen zu verstehen, um sie benutzen und weiterentwickeln zu können. Gerade für sie kann es besonders ärgerlich sein, Teil ihrer Zeit und Ressourcen in die Entwicklung und Implementierung von Workflows investieren zu müssen, anstatt sich auf ihre eigentliche Forschung konzentrieren zu können.

2 Hintergrund

2.1 Debugging

Als Debugging bezeichnet man den Prozess der Fehleridentifizierung, -analyse und -behebung in Computerprogrammen und -technik. Bugs können die System- bzw. Code Performance beispielsweise negativ beeinflussen, Code am Compilieren hindern oder Sicherheitslücken darstellen [13]. Auch deshalb ist Debugging ein riesiges Forschungsgebiet der Informatik, das sich unter anderem mit einer Vielzahl von Debugging-Techniken, wie beispielsweise trace Debugging oder omniscient Debugging, [14] und Debugging-Tools, wie zum Beispiel Airbrake [15] oder ReSharper [16], beschäftigt. Ein Teilgebiet des Debuggings ist das sogenannte automatisierte Debugging. Hierbei handelt es sich um Funktionen oder Tools, die den Programm-Code analysieren, Fehler identifizieren und Berichtigungen durchführen können [17].

2.2 Wissenschaftliche Workflow-Management-Systeme

Wissenschaftliche Datenanalysen benötigen meist mehrere, verschiedene Schritte. Wissenschaftliche Workflow-Management-Systeme, kurz Workflow-Systeme, sind eine Art von Software Tools, die es ermöglichen, die gesamte oder Teile einer Datenanalyse in einer Pipeline zu definieren [18]. Die Pipelines in Workflow-Systemen werden Workflows oder wissenschaftliche Workflows genannt. Mit ihnen können die Schritte einer Datenanalyse entworfen, geordnet, gemanagt und automatisiert werden. Workflows bieten somit die Möglichkeit, Datenanalysen zu vereinheitlichen und reproduzierbar zu machen [7]. Deshalb werden Workflow-Systeme in vielen wissenschaftlichen Fachrichtungen genutzt, allen voran der Bioinformatik [19]. Es gibt eine Vielzahl an unterschiedlichen Workflow-Systemen. Populäre Beispiele sind Nextflow [20], Snakemake [21] oder Apache Airflow [22].

2.3 Nextflow

Nextflow ist ein Workflow-System, das primär für bioinformatische Datenanalysen genutzt wird und sich steigender Popularität erfreut [12]. Workflows werden in einer Domain-Specific Language, kurz DSL, implementiert, die auf Apache Groovy basiert [23]. Für jeden Prozess in einer Pipeline muss in Nextflow der gewünschte Input und Output definiert werden. Bei Ausführung des Workflows übernimmt Nextflow auf Grundlage des zuvor definierten In- und Ouputs dann automatisch die Koordination der Prozesse [24]. Ein Workflow besteht meist aus mehreren Prozessen. In Nextflow führt jeder Prozess ein Skript oder ein Tool aus. Somit können in einer Pipeline für die jeweiligen Prozesse unterschiedliche Tools und Skriptsprachen verwendet werden. Nextflow bietet die Möglichkeit, auf beliebige Skriptsprachen zurückzugreifen; wie zum Beispiel Python, Ruby oder Pearl [25].

2.4 nf-core

nf-core ist ein 2018 ins Leben gerufenes Projekt von Wissenschaftlern und Bioinformatikern, um Pipelines zu archivieren, teilen und vereinheitlichen. Das Projekt ist eine der größten Sammlungen von in Nextflow implementierten wissenschaftlichen Workflows. Stand 17.03.2024 beinhaltet das Projekt 100 Pipelines, 1162 Module und 56 Workflows [26].

3 Forschungsziel

Ziel der Arbeit ist eine Evaluierung der Leistungen von LLMs beim Support von Entwicklern und Nutzeren von Workflow-Systemen bezüglich fehlerhafter Workflows. Die LLMs sollen dabei sowohl den Fehler identifizieren, als auch einen geeigneten Lösungsvorschlag klar und korrekt kommunizieren können.
Hierfür werden zwei unterschiedlichen Sprachmodellen, verschiedene fehlerhafte, in Nextflow verfasste Workflows übergeben. Die Fehler werden dabei nach der sogenannten Mutation-Testing-Methode automatisiert eingefügt. Daraufhin werden die LLMs aufgefordert, in diesen Workflows die Fehlerquelle zu identifizieren, eine Fehlerlösung auszugeben und diese verständlich zu begründen. Dabei werden auch verschiedene Promptstrategien entworfen und untersucht. Anschließend werden die Antworten der Sprachmodelle nach Richtigkeit (über die Accuracy) und Qualität (über ein Ratingsystem) bewertet.
Die Ergebnisse sollen dabei helfen, die Arbeit mit Workflow-Systemen zu erleichtern und effizienter zu gestalten.

4 Arbeitsverlauf

Zuerst werden, zur späteren Evaluierung der Performance der LLMs, verschiedene Fehlerkategorien designt. Dabei sollten die Kategorien einen möglichst großen Teilbereich der Arten von Bugs abdecken (z.B. Syntaxfehler, Semantikfehler, Laufzeitfehler, etc.).
Um zur Evaluierung der LLMs ein geeignetes Datenset zu erstellen, werden unterschiedliche, bereits existierende und frei verfügbare Workflows ausgewählt, beispielsweise aus nf-core, und in diese dann Fehler (Bugs) eingebaut. Defür wird das Mutation-Testing-Verfahren angewandt. Hierbei werden aus einem Programm automatisiert sogenannte Mutanten erzeugt. Mutanten sind Programm-Code Artefakte, in die zufällig beliebige Bugs bzw. Veränderungen eingebaut wurden [27]. Somit ergibt sich ein Datenset aus einer Vielzahl von Bugs-beinhaltenden Codebeispielen.
Anschließend werden 2 geeignete LLMs ausgewählt und implementiert. Hier kommen zwei der populärsten LLMs zum Einsatz: ChatGPT-3.5 und LLaMA. Aufgrund der schnellen Entwicklungen und Weiterentwicklungen auf dem Gebiet der Sprachmodelle ist ein Austauschen einer oder beider Modelle zum Zeitpunkt der Konfigurierung der Sprachmodelle möglich. Wichtig ist, dass die LLMs stets mit keinen oder geringfügigen Kosten und freier Verfügbarkeit verbunden sind, um eine spätere Nutzung möglichst attraktiv zu machen und die Arbeit möglichst repräsentativ zu gestalten.
Der Kern der Arbeit ist schließlich die Evaluierung der Antwortqualität der zuvor ausgewählten LLMs bezüglich des im ersten Schritt erarbeiteten Datensets. Hierbei wird geprüft, ob die LLMs sowohl die Fehler erkennen, als auch einen korrekten Lösungsvorschlag unterbreiten können. Dabei wird neben der Gesamtperformance die jeweilige Performance in den zuvor definierten Fehlerkategorien gemessen und evaluiert. Dabei wird zur Evaluierung der Korrektheit der Antworten die accuracy berechnet und der Qualität der Antworten ein Ratingsystem entworfen.
Wie in anderen Arbeiten bereits behandelt, könnte eine klare Promptstrategie und einheitliche -architektur zu verbesserten Ergebnissen der LLMs führen [10]. Ob und wie diese designt und entworfen werden müssen, lässt sich am besten ergebnisbasiert, während des Experiments bestimmen.

5 Verwandte Forschung

Automatisiertes Debugging, oder auch automated program repair, ist sowohl für Forschung als auch für die Industrie [28] von großem Interesse. Ein Teilbereich ist die Forschung zu Debugging mit Hilfe von LLMs [29]. Ein Vorteil dieser Technik ist die Möglichkeit, auf die Fragen der Entwickler einzugehen und Fehlerbereinigungen verständlich erklären zu können. Fehler des automatisierten Debuggings könnten so schneller identifiziert und die Akzeptanz der automatisierten Fehlerbehebung bei Entwicklern und Benutzern gesteigert werden [30].
Auch mit Bug-Kategorisierung befassen sich bereits einige Forscher[31]. Arbeiten, die sich mit den Supportmöglichkeiten von Sprachmodellen beim Debugging von Programmcode beschäftigen, haben ebenfalls meist eine strukturierte Einteilung und Kategorisierung der Fehlerarten [32]. Gerade beim Entwurf der Fehlerkategorien kann sich an diesen orientiert werden, um eine möglichst große Bandbreite an Fehlerarten abzudecken. Aufgrund der, in der Einleitung bereits erwähnten, erschwerten Vergleichbarkeit von wissenschaftlichen Workflow-Systemen und populären Programmiersprachen, müssen die Kategorien jedoch an die Eigenschaften von Wokflow-Systemen angepasst werden.
Da es sich bei Workflow-Nutzern und -Entwicklern meist um Personen handelt, deren Wissens- und Forschungsschwerpunkt nicht auf dem Gebiet der Softwareentwicklung liegt, könnte eine Debugging-Unterstützung auf Sprachmodell-Basis von besonders großem Nutzen sein. Forschungen haben gezeigt, dass gerade Entwickler, deren Code überdurchschnittlich viele Fehler enthält, da sie gerade erst am Anfang ihrer Ausbildung stehen oder eine neue Sprache erlernen, von einem LLM gestützten Support beim Debugging von Code stark profitieren können [33]. Somit könnte nicht nur Benutzern die Entwicklung und Nutzung von Workflows erleichtern werden, sondern auch die Attraktivität von Workflows durch einen erleichterten Einstieg in neue Workflow-Systeme gesteigert werden.
Auch Forschungen zum Einsatz von LLMs in Bezug auf Workflows und Workflow-Systemen wird bereits betrieben. Während die Leistungen von populären LLMs bei manchen Aufgaben in einer Stichprobe von Experten als positiv wahrgenommen wurden, beispielsweise beim Verstehen und Erklären von Workflows, schnitten sie bei anderen Aufgaben, wie bei der Erweiterung von Code auf Grundlage der Entwicklerbeschreibung, eher schlecht [10]. Das Thema LLM gestütztes Debugging von Workflows wurde dabei jedoch noch nicht betrachtet.

6 Literaturverzeichnis

[1] Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Bing, Y., Hu, X. (2023). Harnessing the power of llms in practice: A survey on chatgpt and beyond. ACM Transactions on Knowledge Discovery from Data.
[2] Achiam, J., Adler, et al. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.
[3] Luo, Q., et al. (2024). RepoAgent: An LLM-Powered Open-Source Framework for Repository-level Code Documentation Generation. arXiv preprint arXiv:2402.16667.
[4] Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., Becker, B. A. (2023, March). Using large language models to enhance programming error messages. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (pp. 563-569).
[5] Gulwani, S., Polozov, O., Singh, R. (2017). Program synthesis. Foundations and Trends® in Programming Languages, 4(1-2), 1-119.
[6] Taylor, A., Vassar, A., Renzella, J., Pearce, H. (2024, March). dcc–help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (pp. 1314-1320).
[7] Ahmed, A. E., et al. (2021). Design considerations for workflow management systems use in production genomics research and the clinic. Scientific reports, 11(1), 21680.
[8] Yu, H., et al. (2024, February). Codereval: A benchmark of pragmatic code generation with generative pre-trained models. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (pp. 1-12).
[9] Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N., Svyatkovskiy, A. (2023, November). Inferfix: End-to-end program repair with llms. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1646-1656).

[10] Sänger, M., De Mecquenem, N., Lewińska, K. E., Bountris, V., Lehmann, F., Leser, U., Kosch, T. (2023). Large Language Models to the Rescue: Reducing the Complexity in Scientific Workflow Development Using ChatGPT. arXiv preprint arXiv:2311.01825.

[11] https://stackoverflow.com/tags [zuletzt besucht 17.03.2024]
[12] https://seqera.io/blog/state-of-the-workflow-2022-results/ [zuletzt besucht 17.03.2024]
[13] https://www.computerweekly.com/de/definition/Debugging [zuletzt besucht 21.03.2024]
[14] Ghosh, D., Singh, J. (2020). A systematic review on program debugging techniques. Smart Computing Paradigms: New Progresses and Challenges: Proceedings of ICACNI 2018, Volume 2, 193-199.
[15] https://www.airbrake.io/ [zuletzt besucht 21.03.2024]
[16] https://www.jetbrains.com/resharper/ [zuletzt besucht 21.03.2024]
[17] Cleve, H., Zeller, A. (2000). Finding failure causes through automated testing. arXiv preprint cs/0012009.
[18] Cavalcanti, M. C., Targino, R., Baiao, F., Rössle, S. C., Bisch, P. M., Pires, P. F., Campos, M. L. M., Mattoso, M. (2005). Managing structural genomic workflows using Web services. Data & Knowledge Engineering, 53(1), 45-74.
[19] https://seqera.io/blog/the-state-of-the-workflow-2023-community-survey-results/ [zuletzt besucht 17.03.2024]
[20] Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., Notredame, C. (2017). Nextflow enables reproducible computational workflows. Nature biotechnology, 35(4), 316-319.

[21] Köster, J., Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. Bioinformatics, 28(19), 2520-2522.
[22] Harenslak, B. P., de Ruiter, J. (2021). Data pipelines with apache airflow. Simon and Schuster.
[23] https://www.nextflow.io/docs/latest/developer/nextflow.config.html [zuletzt besucht 17.03.2024]
[24] https://www.nextflow.io/docs/latest/index.html [zuletzt besucht 18.03.2024]
[25] https://www.nextflow.io/docs/latest/process.html [zuletzt besucht 18.03.2024]
[26] https://nf-co.re/ [zuletzt besucht 17.03.2024]
[27] Papadakis, M., Kintis, M., Zhang, J., Jia, Y., Le Traon, Y., Harman, M. (2019). Mutation testing advances: an analysis and survey. In Advances in computers (Vol. 112, pp. 275-378). Elsevier.
[28] Marginean, A., Bader, J., Chandra, S., Harman, M., Jia, Y., Mao, K., Mols, A., Scott, A. (2019, May). Sapfix: Automated end-to-end repair at scale. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 269-278). IEEE.
[29] Surameery, N. M. S., Shakor, M. Y. (2023). Use chat gpt to solve programming bugs. International Journal of Information Technology & Computer Engineering (IJITC) ISSN, 24555290(3.01), 17-22.
[30] Kochhar, P. S., Xia, X., Lo, D., Li, S. (2016, July). Practitioners' expectations on automated fault localization. In Proceedings of the 25th international symposium on software testing and analysis (pp. 165-176).
[31] Barr, A. (2004). Find the Bug: A Book of Incorrect Programs. Addison-Wesley Professional.
[32] Tian, R., Ye, Y., Qin, Y., Cong, X., Lin, Y., Liu, Z., Sun, M. (2024). Debugbench: Evaluating debugging capability of large language models. arXiv preprint arXiv:2401.04621.
[33] Taylor, A., Vassar, A., Renzella, J., Pearce, H. (2024, March). dcc–help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (pp. 1314-1320).