Chat-GPT & Co in der Open-Source Entwicklung

Jürgen Groß - jgross@suse.com

Es geht nicht um:

oder

ChatGPT & Co in der Open-Source Entwicklung
eine kritische Betrachtung

Disclaimer:

- Dieser Vortrag basiert auf meiner eigenen Meinung (entspricht also nicht unbedingt der Meinung von SUSE)
- Er bezieht sich auf heute allgemein verfügbare KIs (wie z.B. ChatGPT, Claude, Copilot)
- Es geht lediglich um die Verwendung von KIs bei der Erstellung von Software
- Die Bilder dieses Vortrags wurden mit deepai.org erstellt

KI allgemein

Nvidia:

"Es ist unsere Aufgabe, Computertechnik so zu bauen, dass niemand mehr programmieren muss und dass die Programmiersprache menschliche Sprache ist."

Quelle:
https://www.heise.de/news/Nvidia-CEO-Wir-muessen-dafuer-sorgen-dass-niemand-mehr-programmieren-muss-9638720.html

KI Basis

- Heute: meistens LLMs (Large Language Models)
- LLMs bewerten Input auf vielerlei Art und Weise (“Dimensionen”), erzeugen daraus mit Hilfe von Statstik den Output
- Trainingsdaten bestimmen die Qualität des Outputs
- Problem: dem Output sieht man nicht sofort an, ob er auf Fakten beruht oder auf “Halluzinationen”, also falschen Verknüpfungen von Trainingsdaten
- Theoretisch sollten sich Halluzinationen mit den richtigen Trainingsdaten vermeiden lassen, aber: Anwender kennen die Traingsdaten für die KI nicht

Szenarien für KI Einsatz

- Erstellen von produktivem Code (Copilot, Claude, …)
- Auffinden von Problemen in Code
- Erstellen von Testfällen (Code in Qualitätssicherung)
- Erstellen von Texten (Kommentare, Dokumentation, sprachliche Verbesserungen)
- Erstellen von graphischen Elementen

Probleme durch KI bei Codeerstellung

- Fehlerhafter Code (z.B. Randbedingungen nicht beachtet, …)
- Code nicht kompatibel mit Lizenz (z.B. bei GPL-V2-only Code könnte die KI eine Kopie von GPL-V3 Code aus den Trainingsdaten ausspucken)
- Trainingsdaten könnten proprietären Code enthalten (z.B. bei KI von Microsoft), daher drohen potentiell sogar Rechteverletzungen (siehe SCO gegen IBM)
- Stimmt “Developer’s Certificate of Origin” (Entwickler garantiert damit, dass er den Code geschrieben hat und so freigeben darf)?
- Unerfahrene Programmierer können komplexen Code einreichen, den sie evtl. nicht wirklich verstehen und daher auch nicht richtig testen konnten
- Potentielle Überlastung von Maintainern durch große Menge an “Schrott Beiträgen durch KI”

Beispiele aus der Praxis

Qemu:

Current QEMU project policy is to DECLINE any contributions which are believed to include or derive from AI generated content. This includes ChatGPT, Claude, Copilot, Llama and similar tools.

Quelle: Alex Bennée, KVM-Forum 2025, Mailand
https://github.com/qemu/qemu/commit/3d40db0efc22520fa6c399cf73960dced423b048

Probleme durch KI bei Fehlersuche

- Durch KI verfasste Meldungen von Sicherheitslücken sehen sehr überzeugend aus, sind aber oft keine echten Probleme
- Security-Teams werden teilweise heute schon durch “falsche” Fehler-Reports be- bzw. überlastet
- Lokaler Einsatz von KI zur Erkennung von Fehlern kann okay sein, solange der KI-Nutzer die gefundenen Probleme selbst bewerten kann (keine Belastung Anderer)

Beispiele aus der Praxis

cURL:

"Ab sofort bannen wir umgehend jeden Berichterstatter, der Reports einreicht, die wir als KI-Müll erachten. Eine Grenze wurde überschritten. Wir werden effektiv geDDoSt. Wenn wir könnten, würden wir ihnen eine Rechnung für das Verschwenden unserer Zeit stellen", erklärt Stenberg weiter. "Wir haben noch keinen einzigen gültigen Sicherheitsbericht gesehen, der mittels KI-Hilfe erstellt wurde."

Quelle:
https://www.heise.de/news/cURL-Maintainer-Habe-die-Nase-voll-wegen-KI-Bug-Reports-10372739.html

Probleme durch/mit KI Training

- Gerade Open Source Repositories leiden unter DOS durch KIs:
  - KIs halten sich oft nicht an robots.txt
  - Web-Interfaces müssen gegen KI-Zugriffe gesichert werden um Überlastung der Server zu vermeiden
  - Potentielle Lösung: https://github.com/creativecommons/cc-signals
- Wie unterscheidet eine KI “gute” von “schlechten” Trainingsdaten?