Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog https://seblog.cs.uni-kassel.de/ss24/generative-ki-in-der-software-technik/ zu berücksichtigen.

Abgabefrist ist der XX.07.2024 – 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. In eurem Repository soll sowohl der aktuelle Stand des Berichtsheftes, als auch die zu den jeweiligen Aufgaben gehörenden Ergebnisse an generiertem Code oder anderen Artefakten persistiert werden.

Wir nutzen **weiterhin** das Repository, was in Hausaufgabe 1 bereits unter folgendem Link angelegt wurde:

https://classroom.github.com/a/XeAtEq1Z

Die Bearbeitung der gestellten Aufgaben muss wie in Vorlesung und Übung besprochen in einem einheitlichen Berichtsheft dokumentiert werden.

- **Handschriftliche (Teil-)Berichte werden nicht gewertet.**
- **Nicht dokumentierte (Teil-)Berichte werden mit 0 Punkten bewertet!**
- **Nicht persistierte Lösungen für Aufgaben wie Code oder andere Artefakte führen zu 0 Punkten für diese Aufgabe.**
- **Das Berichtsheft muss als PDF-Datei (.pdf) abgegeben werden. Jedes Experiment ist in einem eigenen Kapitel anzulegen.**

Bericht

Jedes Experiment soll im Berichtsheft auf einer neuen Seite beginnen, sodass eine klare Unterteilung zwischen den einzelnen Experimenten gegeben ist. Ein Experiment ist schriftlich in die Abschnitte **Vorbereitung**, **Durchführung** und **Auswertung** zu unterteilen (je nach Aufgabentyp kann dies ergänzt werden).

Die Gesamtlänge eines Berichts (**minus der Abbildungen**) sollte in etwa bei 1–2 Din A4 Seiten (*Schriftgröße 11–12*) liegen. Wir empfehlen die Anfertigung des Berichtsheftes mit Latex.

Vorbereitung

Während der Erstellung dieser Hausaufgabe ist uns ein Fehler unterlaufen. Wir haben diese in der Annahme geplant, dass die API von Gemini free-to-use ist. Dies gilt für einige Länder (Deutschland inbegriffen) nicht.

Wir befinden uns in der Klärung, wie wir diesem Problem entgegnen, da wir weiterhin an der Aufgabenstellung festhalten möchten. Sobald wir eine Lösung gefunden und mit euch kommuniziert haben, beginnt die zwei-wöchentliche Bearbeitungszeit. Solange steht es euch frei, die Aufgabe so gut es geht vorzubereiten (Setup, Dateien einlesen, Text in Dateien schreiben sowie das automatische Prüfen auf Lauffähigkeit bieten sich hier an).

Aufgabe 1 – Automatisierung von Mutationstesten

In dieser Aufgabe soll mithilfe der KI das Mutationtesting aus Hausaufgabe 03 automatisiert werden.
Mittels der KI sollen Mutanten des gegebenen Codes generiert werden. Basierend auf den generierten Mutanten soll ermittelt werden, welche die gegebene Testsuite bereits eliminiert und welche nicht. Ziel ist hier, möglichst Mutanten zu generieren, die überleben. Diese überlebenden Mutanten sollen (falls nicht äquivalent zum Ausgangsprogramm) als Grundlage genutzt werden, um die Testsuite zu erweitern, also Tests zu ergänzen, die diese Mutanten töten. Der gegebene Code ist in Java geschrieben.

Als Szenario für diese Aufgabe nehmen wir an, dass eine Codebasis vorliegt, die nur wenig dokumentiert wurde und deren Testsuite wir erweitern müssen. Wir wissen aus unserer vergangenen Erfahrung mit KI und Mutationstesten, dass diese Arbeit per Hand sehr mühsam ist. Daher wollen wir das tun, was wir als Informatiker immer tun und diesen Vorgang automatisieren. Folgendes soll unser System leisten:

- Die Codebasis soll aus den Java-Dateien ausgelesen werden. Mittels der ausgelesenen Daten sollen Prompts erstellt werden, die für den eingebauten Codeabschnitt Mutanten erzeugen. Die Prompts sollen versuchen, möglichst kurze Antworten des LLMs zu erzeugen.
- Die Antwort des LLMs soll ausgelesen werden und die Mutante in eine geeignete Datei geschrieben werden.
- Die Mutante soll auf automatisch auf Lauffähigkeit geprüft werden.
- Die existierende Testsuite soll gegen die lauffähigen Mutanten ausgeführt werden

Hier enden die Anforderungen für Bachelor-Studierende?

- Überlebende Mutanten sollen genutzt werden, um die Testsuite zu erweitern. (Also wieder aus der Datei einlesen und in einen Prompt einbauen.)
- Der daraus entstandene Mutanten-Test soll in eine Datei geschrieben und automatisch auf Lauffähigkeit geprüft werden. Wenn der Test durchläuft, der Ausgangscode den Test erfüllt und der Test die Branch-Coverage erhöht, soll er in die existierende Testsuite integriert werden. Andernfalls wird er verworfen.
- Am Ende eines solchen Durchlaufes soll das System ausgeben, wie viele Mutanten generiert wurden, wie viele davon lauffähig waren, und wie viele davon die Testsuite überlebt haben.
- Dieses Vorgehen soll wiederholt werden, bis eine Branchcoverage von **95%** erreicht wurde oder eine Grenze von zu sendenden bzw. zu empfangenden Token überschritten wurde. **Wir halten während der Entwicklung mit euch Rücksprache und legen dann, basierend auf euren Erfahrungen, gemeinsam eine Grenze fest, mit der ihr das Experiment endgültig durchführen könnt.**

Ziel ist es, das Mutationstesten wie angegeben zu automatisieren. Hierzu darf die Programmiersprache verwendet werden, die euch am besten liegt.

Codebasis

Mit der folgenden, bereits auf Blatt 3 verwendeten Codebasis sollt ihr euer entwickeltes Programm am Ende laufen lassen und über die Ergebnisse berichten. **Für Tests während der Entwicklung empfiehlt es sich, auch andere Java-Programme zu verwenden, um das System nicht aus Versehen auf den Beispielcode zu optimieren. Hier dürft ihr frei eigene Programme verwenden.**

Die zugrundeliegende Javadatei können hier heruntergeladen werden:

`https://github.com/sekassel/gkistss24-files`

Folgende Dateien sind für die Aufgabe relevant:

- `Aufgabenblatt 3/sekasselmaps/MapService.java`
- `Aufgabenblatt 3/sekasselmaps/MapServiceTest.java`
- `Aufgabenblatt 3/sekasselmaps/model/City.java`
- `Aufgabenblatt 3/sekasselmaps/model/Location.java`
- `Aufgabenblatt 3/sekasselmaps/model/Order.java`
- `Aufgabenblatt 3/sekasselmaps/model/Street.java`

Programmverhalten
Das Programm sucht einen möglichen Pfad von einer Stadt zu einer anderen.

Bericht

Das programmierte Projekt soll im Berichtsheft kurz (max. 2 Seiten) dokumentiert werden. Hierbei soll ein Durchlauf des Programmes und dessen Ergebnis an einem Beispiel beschrieben werden. Eine detaillierte Quelltextdokumentation ist nicht notwendig. Dokumentieren Sie in Ihrer Evaluation explizit, inwieweit die im Szenario festgelegten Ziele erreicht werden konnten (eventuelle weitere Qualitätseigenschaften der Tests können informell diskutiert werden).

Wichtig!
(Sollte während der Entwicklung klar werden, dass der Aufwand den zeitlichen Rahmen sprengt, sprecht uns bitte an. Der Schwierigkeitsgrad ist sehr stark von bereits belegten Modulen und Vorkenntnissen abhängig und kann deshalb auch nicht pauschal für jeden Studierenden vorausgesagt werden.)