Exposé

Nutzung von Sprachmodellen zur Verbesserung von Bezeichnern in Java-Programmen

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit wird untersucht, wie große Sprachmodelle (LLMs) automatisch die Namen von Variablen, Methoden und Klassen in Java verbessern können. Gute Namen machen den Code [3] leichter verständlich und einfacher zu pflegen. In der Praxis gibt es aber oft kurze, unklare oder falsche Namen.

Das Ziel ist, eine komplett automatische Lösung zu entwickeln:

1. Java-Beispiele mit schlechten Namen werden erstellt.
2. Die Beispiele werden an ein Sprachmodell geschickt.
3. Die Vorschläge werden automatisch ausgewertet.

Dazu wird geprüft, welche Art von Prompts [2] am besten funktioniert, um verständliche und passende Namen zu bekommen, die Java-Konventionen einhalten und sprachliche Fehler vermeiden [1].

> **Forschungsfrage:** Wie gut können Sprachmodelle automatisch bessere Bezeichner in Java vorschlagen, ohne dass ein Mensch eingreifen muss?

Die Arbeit ist relevant, weil Sprachmodelle zunehmend in der Softwareentwicklung eingesetzt werden – zum Beispiel für Code-Vervollständigung, Dokumentation oder Refactoring. Dadurch wird untersucht, wie LLMs die Codequalität langfristig verbessern können.

2 Gliederung der Arbeit

1. **Einleitung.** Vorstellung des Themas, Ziel und Forschungsfrage.
2. **Theoretischer Hintergrund.** Grundlagen zu Sprachmodellen (z. B. GPT, CodeBERT), sprachliche Fehler in Bezeichnern und Regeln für gute Namen.
3. **Analyse vorhandener Arbeiten.** Überblick über Forschung zu KI und Code-Verbesserung.
4. **Methodik und Durchführung.** Automatische Pipeline: Code generieren, an LLM senden, Ergebnisse automatisch auswerten. Prompts und Bewertung festlegen.
5. **Evaluation.** Ergebnisse prüfen: Verständlichkeit, Bedeutung, Einhaltung von Regeln, Vermeidung von sprachlichen Fehlern.
6. **Diskussion und Schluss.** Grenzen der Methode, Verbesserungsideen, Ausblick auf weitere Forschung.

3 Vorgehensplan

1. **Woche 1–2: Literatur.** Recherche zu LLMs, sprachlichen Fehlern in Bezeichnern und Regeln für gute Namen.

2. **Woche 3–4: Entwicklungsumgebung.** Einrichten von Java und Python. Analyse von APIs (z. B. OpenAI, HAWKI).

3. **Woche 5–6: Codegenerierung und Prompts.** Skript erstellt Java-Klassen mit schlechten Namen. Prompts für Variablen, Methoden und Klassen entwerfen.

4. **Woche 7–8: Experimente.** Code automatisch an LLM schicken, Vorschläge speichern und auswerten.

5. **Woche 9: Analyse.** Vergleich verschiedener Prompts, Bewertung der Vorschläge nach Verständlichkeit, Bedeutung und Regeln.

6. **Woche 10: Abschluss.** Dokument fertigstellen, Ergebnisse zusammenfassen und Abgabe vorbereiten.

4 Definitionen und Quellen

Wichtige Quellen

- **LLM (Large Language Model):** ein KI-Modell, das große Mengen an Text analysiert und auf dieser Basis neuen Text generiert, z. B. GPT-4 oder CodeBERT.
- **Pipeline:** eine Abfolge automatischer Verarbeitungsschritte – hier: Code generieren, an LLM senden und Ergebnisse auswerten.
- **Bezeichner (Identifier):** Namen von Variablen, Methoden oder Klassen in Java. Sie sollen kurz, eindeutig und verständlich sein.[**?**]
- **Sprachlicher Fehler (Linguistic Anti-Pattern):** unpassende, unklare oder doppeldeutige Bezeichner, die den Code schwerer verständlich machen.
- **Prompt:** eine textbasierte Eingabe, die bestimmt, wie das Sprachmodell reagiert und welche Vorschläge es generiert.
- **HAWKI-System:** eine Plattform der WHZ, die den Zugriff auf Sprachmodelle (LLMs) ermöglicht.

5 Schwierigkeiten und Risiken

1. **Qualität der Modelle.** Ergebnisse sind manchmal unterschiedlich, besonders bei komplexem oder unklarem Code.
2. **Zugriff.** API-Limits oder fehlende Internetverbindung können Experimente verzögern.
3. **Bewertung.** Eine automatische Auswertung kann semantische Qualität nur eingeschränkt erfassen.
4. **Zeit.** Prompts zu testen und zu optimieren, benötigt mehrere Versuche.

Literatur

[1] Arnaoudova, V.; Di Penta, M.; Antoniol, G. (2014): *Linguistic Anti-Patterns: What They Are and How Developers Perceive Them.* Empirical Software Engineering. Verfügbar unter: veneraarnaoudova.ca.

[2] Startup Creator (2023): *Die besten ChatGPT Prompts.* Verfügbar unter: startup-creator.com.

[3] Oracle: *Naming Conventions.* Verfügbar unter: Java Naming Conventions.