KI-gestützte Modernisierung von Altanwendungen: (Sentiment-) Analysen im Diskurs des Anforderungsmanagements

Andreas Schmietendorf, Sandro Hartenstein, Sidney Leroy Johnson
Hochschule für Wirtschaft und Recht Berlin
andreas.schmietendorf@hwr-berlin.de, sandro.hartenstein@hwr-berlin.de, s_johnson20@stud.hwr-berlin.de

1. Motivation

Die Entwicklung von Software erfolgt zunehmend unter Einsatz von Algorithmen der künstlichen Intelligenz. Zumeist gilt das Interesse der Integration von endnutzerzentrierten KI-Funktionen, welche mit Hilfe von Frameworks und APIs, aber auch cloudbasierten Services realisiert werden. Weniger beachtet wird die Frage, inwieweit die Aufgabenstellungen des Software Engineerings selbst von KI-Ansätzen profitieren können. Bereits vor 35 Jahren waren derartige Überlegungen Gegenstand von Konferenzen (vgl. (Gotwalt 1987)). Aktuelle Erwartungen zum KI-Einsatz im Software Engineering werden zumeist mit einem hohen Automationsgrad wie unter (Henkes 2019) in Verbindung gebracht:

> „Vor allem KI und Machine Learning mischen die Karten am Markt neu, da sie in der Lage sind, die Software-Entwicklung und -Wartung weiter zu automatisieren und zu industrialisieren."

2. Ausgangslage

Die langjährige Pflege, Wartung und Weiterentwicklung bestehender Softwaresysteme verursachen im praktischen Umfeld massive Kosten und Risiken. Gerade bei gewachsenen Softwaresystemen zeigen sich vielfach Schwächen und Inkonsistenzen in Bezug auf existierende Dokumentationen und eingesetzte (UML-) Modelle. Im Diskurs der Modernisierung von Altanwendungen bedarf es häufig der Gewinnung von Informationen aus unzureichend gepflegten Dokumentationen, Review-Dokumenten, aber auch Quellcodes. Aufgrund der fragmentierten und umfänglichen Datenmengen innerhalb eines Softwareentwicklungsprojekts (u.a. Anforderungen, Schnittstellenspezifikationen, Modelle, Quellcode, Testdaten und Testfälle, Feedbacks, …) stellt sich die Frage, inwieweit Methoden der künstlichen Intelligenz eine Analyse zur Gewinnung von „Einsichten" sinnfällig unterstützen können. Im Folgenden seien Beispiele für mögliche Forschungsfragen im Diskurs der Modernisierung von Anwendungen genannt:

- In welcher Weise präsentieren sich Artefakte des Software Reengineerings, die als potentielle Quelldaten für KI-Ansätze genutzt werden können?
- Welche Zielstellungen des Software Reengineerings lassen sich durch den Einsatz von ML-Algorithmen erreichen?
- Wie können ML-Algorithmen auf die verfolgten Ziele trainiert werden und welche Daten können dafür verwendet werden?

Im vorliegenden Beitrag soll der Frage nachgegangen werden, wie über mehrere Versionen erstellte Spezifikationen im Diskurs des Anforderungsmanagements KI-basiert analysiert werden können. Mit Hilfe eines Vergleichs der gewonnenen Erkenntnisse mit dem tatsächlich realisierten Interaktionsverhalten lassen sich so Ansätze für eine Modernisierung aus Sicht der Kunden bzw. Nutzer identifizieren. Potenzielle Modernisierungen könnten sich z.B. die Vereinfachung von Interaktionszenarien, die Beseitigung von Redundanzen oder ggf. auch semantisch fehlerhaft implementierte Fachfunktionen beziehen.

3. Existierende Arbeiten

Eine erste Analyse existierender Arbeiten zum Einsatz künstlicher Intelligenz im Software Engineering brachte die folgenden Schwerpunkte:

- (Edjlali 2014) beschäftigt sich mit Big-Data-Einsatzszenarien beim Software Engineering. Diese beziehen sich auf Daten aus sozialen Netzwerken (z.B. Stack Overflow), erzeugte Dokumente und ausgeführte Programme.
- Beim Einsatz intelligenter Entwicklungstools, unterscheidet (Satish 2019) drei Komplexitätsstufen: 1. Automation manueller Aufgaben, 2. Regelbasierte Lösungen und 3. Selbstlernende Systeme.
- Die Identifikation benötigter Testszenarien auf der Grundlage einer KI-basierten Quellcodeanalyse wird von (Endres and Hopstock 2019) untersucht.
- (Zhao and Zhao 2019) beschäftigen sich mit der Vorhersage der Anforderungsentwicklung bei der Produktentwicklung. Analysiert werden Review-Dokumente mit Hilfe einer Kombination von neuronalen Netzen (supervised deep learning) und hierarchischen Themenmodellen (unsupervised learning).

4. KI im Anforderungsmanagement

Sehr allgemein können die folgenden Beispiele für dokumentenzentrierte (in Hinsicht auf die korrespondierenden Quelldaten) KI-Szenarien im Anforderungsmanagement identifiziert werden:

- Prognosen zum Interaktionsverhalten der Softwarenutzer (Mustererkennung) zwecks Erkennung von Optimierungspotentialen auf der Grundlage selbstlernender neuronaler Netzwerke (Yang et al. 2020)
- Prognosen zur Entwicklung von nutzerspezifischen Anforderungen auf der Grundlage über einen längeren Zeitraum erstellter Dokumente, aber auch allgemeiner Meinungsäußerungen/Bewertungen in z.B. Anwenderforen (Park and Kim 2020)
- Klassifikation existierender Anforderungen mit Hilfe von Sentimentanalysen (Natural Language Processing) zur Bewertung bzw. Gewichtung von geforderten Fachfunktionen (Dias Canedo and Cordeiro Mendes 2020)

5. Prototypische Sentimentanalysen

Von den formulierten Beispielen für KI-Szenarien im Anforderungsmanagement werden in dieser Untersuchung die Anwendungsmöglichkeiten von Sentimentanalysen adressiert. Sentimentanalysen dienen der Identifizierung und Klassifizierung von Emotionen und Meinungen in einem gegebenen Text. Es verwendet Textanalysetechniken, um die allgemeine Stimmung des Textes zu bestimmen.

Erste Versuche lexikonbasierter Techniken wurden mithilfe der Software SentiStrength (Islam and Zibran 2018) durchgeführt und brachten wenig aussagekräftige Ergebnisse hervor. Es wurde festgestellt, dass verfügbare Sentimentanalyse-Tools eine sehr geringe Spezialisierung auf Software-Anforderungen besitzen und daher nicht zielführend sind.

Die Konzeption eines auf SE-Anforderungen spezialisierten neuronalen Netzes lässt aussagekräftigere Ergebnisse erwarten. Für einen ersten Prototypen wurde das bestehendes KI-Framework *PyTorch* importiert, modifiziert, trainiert, getestet und evaluiert (Ranaweera 2022). Der Konfigurationsaufbau des Prototyps ist in Abbildung 1 dargestellt.

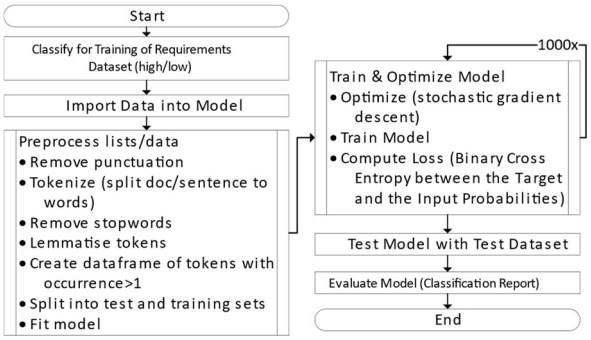

Abbildung 1 Konfigurationsaufbau des Prototyps

Der Prototyp hat mit Hilfe eines modifizierten Trainingsdatensatzes (Souvik 2019) erste Outputs eines NLP-Ansatzes produziert. Dabei wurde unter Verwendung von 48 Trainings- und 12 Testanforderungen eine Vorhersage-Genauigkeit von etwa 65 % erreicht. Mit Erhöhung der Qualität und Quantität des Trainingsdatensatzes wird in Zukunft eine deutliche Verbesserung der Vorhersagegenauigkeit erwartet. Die genutzten Trainingsdaten sind eine Sammlung von Software-Engineering-Anforderungen ergänzt durch die jeweilige Aufwandsschätzung [h für high/l für low]. Der Aufwand wurde manuell nach eigener Einschätzung zugeordnet. Die Aufwand-Zuordnung ersetzt die verbreitete Sentiment-Klassifizierung von positiv und negativ.

6. Reflektion und Ausblick

Die Auswertung der bisher erlangten Ergebnisse durch Prototypen lässt darauf schließen, dass eine Bewertung von Anforderungen bzw. geforderten Fachunktionen mithilfe von ML-Algorithmen durchaus realistisch ist, sofern geeignete Daten für den Trainingsprozess vorliegen. Eine vom Prototyp generierte Verlustkurve, welche in Abbildung 2 zu sehen ist, zeigt, dass die Qualität der Vorhersagen pro Epoche stetig verbessert wird. Der Trainingsprozess ist dementsprechend geeignet.

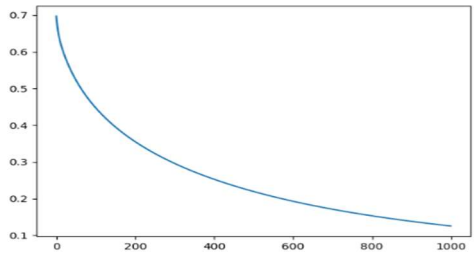

Abbildung 2 Verlustfunktion des Prototyps

Eine geplante Optimierung der Testdatenqualität besteht darin, die Trainingsdaten weiter zu unterteilen, etwa mit einem Aufwand-Score statt einer Einteilung in hoch und niedrig oder einer Aufteilung nach Anforderungsbereich. Darüber hinaus kann der bisher genutzte fiktive Testdatensatz durch aktuelle oder historische Echtdaten ersetzt werden. Im weiteren Verlauf der Forschungsarbeit wird der bestehende Prototyp durch die Bereitstellung der trainierten Modelle, im Sinne der Operationalisierung des Ansatzes, erweitert.

Quellenverzeichnis

Dias Canedo, Edna/Cordeiro Mendes, Bruno (2020). Software Requirements Classification Using Machine Learning Algorithms. Entropy (Basel, Switzerland) 22 (9). https://doi.org/10.3390/e22091057.

Edjlali, R. (2014). Shrinking Big Data. Gartner Symposium/ITxpo,

Endres, T./Hopstock, S. (2019). Machine Learning on Source Code. Available online at www.informatik-aktuell.de/betrieb/kuenstliche-intelligenz/machine-learning-on-source-code.html (accessed 4/19/2022).

Gotwalt, Susan (1987). KI-Methoden für das Software-Engineering. Computerwoche. Available online at www.computerwoche.de/a/amp/ki-methoden-fuer-das-software-engineering,1161793 (accessed 1/4/2023).

Henkes, Heiko (2019). KI mischt die Karten auch im Software-Engineering neu. Das E3-Magazin. Available online at https://e-3.de/ki-mischt-die-karten-auch-im-software-engineering-neu/ (accessed 1/4/2023).

Islam, Md Rakibul/Zibran, Minhaz F. (2018). SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. Journal of Systems and Software 145, 125–146. https://doi.org/10.1016/j.jss.2018.08.030.

Park, Bo Kyung/Kim, R. Young Chul (2020). Effort Estimation Approach through Extracting Use Cases via Informal Requirement Specifications. Applied Sciences 10 (9), 3044. https://doi.org/10.3390/app10093044.

Satish, Sharath (2019). Künstliche Intelligenz in der Softwareentwicklung. Available online at www.thoughtworks.com/de-de/insights/blog/augmenting-software-development-artificial-intelligence (accessed 1/4/2023).

Souvik (2019). Software Requirements Dataset. Available online at www.kaggle.com/datasets/iamsouvik/software-requirements-dataset (accessed 1/12/2023).

Yang, Bin/Wei, Long/Pu, Zihan (2020). Measuring and Improving User Experience Through Artificial Intelligence-Aided Design. Frontiers in psychology 11, 595374. https://doi.org/10.3389/fpsyg.2020.595374.

Zhao, Lingling/Zhao, Anping (2019). Sentiment Analysis Based Requirement Evolution Prediction. Future Internet 11 (2), 52. https://doi.org/10.3390/fi11020052.