Automatisiertes Aktualisieren veralteter Stack-Overflow-Antworten mithilfe von Kommentaren: Eine dokumentenbasierte Analyse von SOUP

Albek, Obai
Hochschule Mannheim
Fakultät für Informatik
Paul-Wittsack-Str. 10, 68163 Mannheim

***Zusammenfassung*—Stack Overflow ist eine zentrale Wissensplattform für Softwareentwicklerinnen und -entwickler, deren Inhalte jedoch durch Bibliotheks- und Application Programming Interface (API)-Evolution sowie veränderte Best Practices veralten können. Veraltete Antworten sind verbreitet und werden oft nur begrenzt nachträglich aktualisiert; zugleich werden Code-Snippets häufig unkritisch wiederverwendet, was praktische Risiken (u.a. sicherheitsrelevante Fehlverwendungen) begünstigen kann. Diese Arbeit untersucht, wie Kommentare als Signal genutzt werden können, um veraltete oder fehlerhafte Antworten systematisch zu aktualisieren. Methodisch kombiniert die Arbeit eine literaturbasierte Synthese mit einer Fallstudie zum Framework Stack Overflow Updater for Post (SOUP), das kommentarbasiert Edit-Beziehungen modelliert und daraus Update-Vorschläge für Posts generiert. Die Analyse zeigt (i) dass Kommentare in Form von comment-induced updates und Update Request Comments (URCs) wiederholt als Auslöser für Verbesserungen dienen, (ii) dass automatisierte Aktualisierung durch Qualitätsfilterung und domänenspezifische Datensätze technisch praktikabel ist, und (iii) dass robuste Plattformpflege ein hybrides Vorgehen erfordert, in dem Künstliche Intelligenz (KI) Vorschläge liefert und Menschen Validierung und Verantwortung übernehmen. Ergänzend diskutiert die Arbeit, warum Plattformpflege auch im Umfeld großer Sprachmodelle (Large Language Models, LLM) relevant bleibt, da LLM-Antworten Qualitätsgrenzen aufweisen und zugleich Anreizverschiebungen die öffentliche Wissensweitergabe in Communities reduzieren können.**

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Problem und Motivation

Stack Overflow ist eine zentrale Wissens- und Problemlösungsplattform für Softwareentwicklerinnen und -entwickler. Im Stack Overflow Developer Survey 2023 wird berichtet, dass nahezu alle Befragten Stack Overflow besucht haben und ein Großteil die Plattform regelmäßig nutzt (z.B. mindestens wöchentlich oder mehrmals pro Monat) [vgl. 1]. Dieses Wissen ist jedoch nicht zeitlos: Bibliotheken, Frameworks und APIs entwickeln sich weiter, Sicherheitsanforderungen ändern sich und Best Practices werden angepasst. Dadurch können ehemals korrekte Antworten veralten und – insbesondere bei unkritischer Wiederverwendung – Fehlentscheidungen oder fehlerhafte Implementierungen begünstigen [vgl. 2].

Die empirische Untersuchung von Zhang et al. zeigt, dass veraltete Antworten auf Stack Overflow verbreitet sind und nur ein vergleichsweise kleiner Anteil tatsächlich nachträglich aktualisiert wird [2]. Hinzu kommt, dass Stack Overflow häufig als Quelle für direkt übernommene Code-Snippets dient. Fischer et al. zeigen am Beispiel sicherheitsrelevanter Snippets, dass unsichere Codebeispiele aus Stack Overflow in großem Umfang in reale Software kopiert werden und dadurch Sicherheitsrisiken in die Praxis gelangen können [3].

Als struktureller Treiber der Obsoleszenz wirkt zudem die API-Evolution: Wird eine API (teilweise) deprecated oder entfernt, können vormals funktionierende Beispiele später irreführend oder falsch werden [vgl. 4]. Die Analyse von Zhou und Walker zeigt für Android-bezogene Beiträge auf Stack Overflow, dass deprecated API-Nutzung in akzeptierten Antworten häufig vorkommt und nur selten explizit als solche kenntlich gemacht wird [4].

Damit wird deutlich, dass „Korrektheit“ auf Stack Overflow nicht nur eine Frage der ursprünglichen Antwortqualität ist, sondern auch der fortlaufenden Wartung. Die Pflege veraltender Inhalte ist auf Stack Overflow jedoch nicht trivial, da Beiträge über Zeit in einem komplexen Zusammenspiel aus Edits und Community-Interaktionen entstehen. Baltes et al. rekonstruieren mit SOTorrent die Evolution von Posts (u.a. auf Ebene einzelner Text- und Code-Blöcke) und machen dadurch sichtbar, dass Änderungen häufig eng mit der Diskussionshistorie verknüpft sind [5].

Gerade Kommentare sind hierbei relevant, weil sie häufig Hinweise auf Bugs, Obsoleszenz oder bessere Alternativen enthalten und als Auslöser für nachträgliche Verbesserungen dienen können [vgl. 6, 7]. Parallel dazu verändern große Sprachmodelle (LLMs) das Informationsverhalten von Entwicklerinnen und Entwicklern. Die empirische Studie von Kabir et al. vergleicht LLM-Antworten direkt mit Stack-Overflow-Antworten und zeigt, dass LLM-Ausgaben trotz hoher Nützlichkeit relevante Qualitätsprobleme aufweisen können (z.B. Korrektheit und Verlässlichkeit) [8].

Del Rio-Chanona et al. berichten Hinweise darauf, dass die Nutzung von LLMs die Aktivität in öffentlichen Q&A-Communities reduzieren kann, was die langfristige Pflege frei verfügbarer Wissensbestände zusätzlich erschwert [9]. [1]

1.2. Forschungsfrage und Beitrag

Vor diesem Hintergrund untersucht diese Arbeit die folgende Forschungsfrage: *Wie können Kommentare auf Stack Overflow genutzt werden, um veraltete oder fehlerhafte Antworten systematisch zu aktualisieren?* Als konkreter Ankerpunkt dient eine aktuelle Arbeit auf der ICSE 2025 (SOUP), die Aktualisierungen von Stack-Overflow-Posts automatisiert, indem sie Kommentar–Edit-Beziehungen modelliert und daraus Edit-Vorschläge generiert [vgl. 10].

Auf dieser Basis leistet das Paper drei Beiträge: (1) Einordnung des Forschungsstands zu veralteten Antworten, Plattform-Evolution und comment-getriebenen Updates [vgl. 2, 5, 6, 7], (2) strukturierte Darstellung der SOUP-Fallstudie (Ziel, Pipeline, Datengrundlage und Evaluation) [vgl. 10] und (3) Diskussion eines praktikablen, hybriden Vorgehens (KI-Vorschläge + menschliche Validierung) sowie einer kontextuellen Einordnung, warum Plattformpflege auch im Umfeld großer Sprachmodelle relevant bleibt [vgl. 8, 9].

1.3. Aufbau der Arbeit

Die Arbeit gliedert sich in mehrere Abschnitte, die vom Problemkontext zur Fallstudie und deren Einordnung führen. Nach der Einleitung fasst Abschnitt 2 die fachlichen Grundlagen zusammen: (i) Stack Overflow als Wissensplattform, (ii) typische Ursachen und Folgen veralteter Antworten sowie (iii) die Rolle von Kommentaren und Edit-Historien als Signal für Wartungsbedarf. Darauf aufbauend beschreibt Abschnitt 3 die Vorgehensweise dieser Arbeit, einschließlich Such- und Auswahlkriterien der Literatur sowie der Kriterien, nach denen die betrachteten Ansätze eingeordnet werden.

Abschnitt 4 bildet den Kern der Arbeit und stellt die Fallstudie SOUP strukturiert dar (Zielsetzung, Pipeline, Datengrundlage und Evaluation). Anschließend diskutiert Abschnitt 5 die Ergebnisse im größeren Kontext von Plattformpflege und dem Einfluss großer Sprachmodelle auf das öffentliche Wissensökosystem. Abschnitt 6 benennt zentrale Grenzen der betrachteten Evidenz und der Übertragbarkeit der Ergebnisse. Abschließend fasst Abschnitt 7 die wichtigsten Erkenntnisse zusammen und skizziert mögliche Weiterentwicklungen.

Der Anhang dokumentiert die Nutzung von LLMs und die verwendeten Prompts gemäß den Vorgaben der Veranstaltung.

2. Hintergrund

2.1. Stack Overflow als Wissensplattform

Stack Overflow ist eine kollaborative Q&A-Plattform, auf der technische Probleme in Form von Fragen und Antworten dokumentiert, bewertet und kuratiert werden. Die

1. Dieses Paper wurde mit Unterstützung eines LLM erstellt (ChatGPT) zur sprachlichen Überarbeitung und zur Strukturierung von Textpassagen. Inhaltliche Aussagen, Auswahl der Quellen und Schlussfolgerungen wurden vom Autor geprüft und verantwortet. Die verwendeten Prompts sind im Anhang dokumentiert.

Nutzung ist breit verankert: In der Stack Overflow Survey 2023 geben 92,5% der Befragten an, Stack Overflow mindestens wöchentlich oder mehrere Male pro Monat zu besuchen [vgl. 1].

Damit fungiert die Plattform in vielen Entwicklungsprozessen als erste Anlaufstelle und prägt sowohl individuelle Problemlösestrategien als auch kollektive Wissensbestände. Charakteristisch ist zudem, dass Inhalte nicht als statische Veröffentlichungen angelegt sind, sondern als Artefakte, die durch Community-Interaktionen fortlaufend weiterentwickelt werden können. Beiträge besitzen eine Versionshistorie (Edits) und werden typischerweise durch soziale Signale (z.B. Bewertungen, Akzeptieren von Antworten) sichtbar gemacht und priorisiert.

Dabei können auch Nutzerinnen und Nutzer ohne volle Bearbeitungsrechte Änderungen vorschlagen, die anschließend über Review-Prozesse bestätigt oder verworfen werden [vgl. 11]. In der Praxis entsteht so ein Spannungsfeld zwischen (i) schneller Problemlösung im Moment der Erstellung und (ii) langfristiger Wartbarkeit als öffentliches Wissensartefakt. Für wissenschaftliche Analysen ist außerdem relevant, dass ein großer Teil der Inhalte maschinell auswertbar verfügbar ist. Seit 2009 werden anonymisierte Daten-Dumps für Stack-Exchange-Seiten veröffentlicht und u.a. über das Internet Archive bereitgestellt [vgl. 12].

Die zugehörige Lizenzinformation wird zusammen mit dem Dump ausgeliefert [vgl. 13]. Zusätzlich war Stack-Overflow-Datenmaterial in der Vergangenheit als öffentlicher Datensatz auch über BigQuery verfügbar, was die Abfrage und Replikation empirischer Analysen erleichtert [vgl. 14]. Aus Sicht der Nachvollziehbarkeit ist schließlich wichtig, dass Stack-Overflow-Inhalte unter CCBYSA-Lizenzen stehen und der Lizenzstatus zeitlich bzw. versioniert differenziert ist [vgl. 15, 16].

Auf dieser Grundlage entstanden Forschungsinfrastrukturen wie SOTorrent, die die Evolution von Beiträgen rekonstruieren und Analysen der Veränderungen auf Ebene einzelner Text- und Code-Blöcke ermöglichen [5]. Solche Datensätze sind für Wartungs- und Aktualisierungsfragen besonders geeignet, weil sie die Plattform nicht nur als Sammlung von Endzuständen, sondern als Prozess historischer Veränderungen abbilden.

2.2. Veraltete Antworten und Pflegebedarf

Die Relevanz von Plattformpflege ergibt sich aus der zeitlichen Dynamik von Softwareentwicklung. Antworten können veralten, wenn sich Laufzeitumgebungen, Bibliotheken oder APIs verändern, wenn Sicherheitsanforderungen steigen oder wenn sich etablierte Praktiken weiterentwickeln. Zhang et al. untersuchen veraltete Antworten auf Stack Overflow und kommen zu dem Ergebnis, dass Obsoleszenz verbreitet ist, während Aktualisierungen nur in einem begrenzten Anteil der Fälle erfolgen [2]. Daraus ergibt sich das Risiko, dass hochsichtbare oder historisch erfolgreiche Antworten in späteren Kontexten falsche Handlungsanreize geben.

Besonders kritisch wird Obsoleszenz dort, wo Code direkt wiederverwendet wird. Fischer et al. analysieren sicherheitsrelevante Android-Szenarien und zeigen, dass Code-Snippets aus Stack Overflow in großem Umfang in reale Anwendungen kopiert werden und unsichere Beispiele so in Produktivsoftware landen können [3]. Als struktureller Treiber kommt API-Evolution hinzu: Zhou und Walker zeigen für Android-bezogene Inhalte, dass deprecated Nutzung auch in akzeptierten bzw. prominenten Beispielen vorkommt und sich automatisiert detektieren lässt [4].

Für Plattformpflege bedeutet dies: Selbst wenn eine Antwort zum Zeitpunkt der Erstellung korrekt war, kann sie später aufgrund externer Evolution inkorrekt oder irreführend werden. Warum Obsoleszenz schwer zu beherrschen ist, hängt zudem mit der Kopplung aus Beitragsstruktur und Community-Prozessen zusammen. Baltes et al. rekonstruieren die Post-Evolution und berichten, dass viele Verbesserungen klein sind, über mehrere Edits verteilt auftreten und häufig relativ früh nach der Erstellung eines Beitrags erfolgen [5].

Für Wartungsstrategien ist dieses Muster relevant: Wenn die „Pflegeenergie“ zeitlich stark auf die Anfangsphase konzentriert ist, bleiben spätere, durch externe Evolution ausgelöste Probleme tendenziell länger bestehen. Daraus folgt ein Bedarf an Mechanismen, die Wartungsbedarf später im Lebenszyklus zuverlässig sichtbar machen und Aktualisierungen effizient unterstützen.

2.3. Kommentare als Signal für Updates

Kommentare sind ein zentrales Interaktionsmedium auf Stack Overflow und werden häufig zur Einordnung, Korrektur und Verbesserung von Fragen und Antworten genutzt. Die Plattform beschreibt Kommentare explizit als Mittel, um Verbesserungen vorzuschlagen oder Klärungen anzufordern [vgl. 17]. Aus Wartungsperspektive sind Kommentare besonders wertvoll, weil sie häufig zeitnäher als formale Edits auf neue Probleme reagieren. So können Nutzerinnen und Nutzer auf Obsoleszenz, Bugs oder bessere Alternativen hinweisen, ohne sofort einen vollständigen Edit zu erstellen.

Soni und Nadi analysieren comment-induced updates und zeigen, dass Kommentare in signifikanter Zahl mit späteren Aktualisierungen von Antworten zusammenhängen; methodisch wird dabei u.a. auf SOTorrent-Daten zurückgegriffen [7]. Sheikhaei et al. untersuchen URCs und zeigen, dass ein großer Anteil der betrachteten Kommentare als Update-Anfragen klassifizierbar ist; zugleich bleibt ein substantieller Teil solcher Anfragen auch langfristig unbeantwortet [6]. Zusammengenommen legen diese Befunde nahe, dass Kommentare ein operationalisierbares Signal für Wartungsbedarf darstellen, dieses Signal jedoch im heutigen Prozess nicht zuverlässig in tatsächliche Pflege überführt wird. Für automatisierte Aktualisierung ist außerdem wichtig, dass Kommentare und Edits in zeitlichen und inhaltlichen Beziehungen auftreten. Baltes et al. berichten eine enge Beziehung zwischen Edits und Kommentaren und motivieren damit Forschung, die Kommentar–Edit-Zusammenhänge systematisch nutzt [5].

3. Methodik

Diese Arbeit ist als literaturbasierte Analyse mit einer fokussierten Fallstudie konzipiert. Ziel ist es, (i) den

Forschungsstand zu veralteten Antworten und kommentarbasierten Update-Signalen auf Stack Overflow einzuordnen und (ii) anhand der Fallstudie SOUP zu analysieren, wie solche Signale in automatisierte Aktualisierungsvorschläge überführt werden können. Die Methodik ist explorativ-deskriptiv angelegt und zielt auf nachvollziehbare Synthese statt statistischer Hypothesentests.

Die einzelnen Schritte der Literatursichtung und des Auswahlprozesses sind in Tabelle 1 zusammengefasst.

Tabelle 1. KOMPAKTER ÜBERBLICK ÜBER VORGEHEN UND AUSWAHLPROZESS.

Aspekt	Umsetzung in dieser Arbeit
Studiendesign	Literaturbasierte Synthese + dokumentenbasierte Analyse des Frameworks SOUP.
Suchraum	Primärquellen aus den im Paper zitierten Publikationen; ergänzend Snowballing (Backward/Forward) ausgehend von Kernarbeiten [18].
Screening	Zweistufig (Titel/Abstract → Volltext). Ausschluss bei fehlendem Bezug, Duplikaten oder fehlendem Volltext (PDF im ZIP).
Extraktion	Pro Quelle: Kontext/Datenbasis, Obsoleszenzbezug, Kommentarrolle, Interventionslogik, Evaluation und Limitationen.
Bewertung	Einordnung entlang D1–D3 (0=nein, 1=teilweise, 2=klar/evidenzbasiert).

3.1. Suchstrategie und Auswahlkriterien

Die Literaturerhebung erfolgte in zwei Schritten:

1) **Gezielte Primärsuche** anhand von Suchbegriffen, die direkt aus der Forschungsfrage abgeleitet wurden.
2) **Erweiterung durch Snowballing** (Backward/Forward) ausgehend von einem Startset zentraler Arbeiten [18].

Die Suche erfolgte primär in den Datenbanken Google Scholar, IEEE Xplore und der ACM Digital Library sowie ergänzend im direkten Umfeld der identifizierten Kernpublikationen. Das Vorgehen orientiert sich an etablierten Leitlinien zur strukturierten Literatursuche und transparenten Dokumentation von Ein- und Ausschlussentscheidungen [vgl. 19].

Suchbegriffe. Verwendet wurden Kombinationen aus Begriffen der folgenden Gruppen:

- *Plattform/Artefakte:* „Stack Overflow“, „posts“, „answers“, „comments“, „edits“
- *Wartung/Alterung:* „obsolete“, „outdated“, „deprecated“, „maintenance“, „update“
- *Kommentar-Signal:* „update request“, „comment-induced“, „comment-driven“
- *Automatisierung/LLM:* „automated post updating“, „post updating“, „ChatGPT“, „large language model“

Startset. Als Startset dienten Kernpublikationen, die die Kernthemen direkt adressieren [2, 5, 6, 7, 8, 9, 10].

3.2. Analyse- und Bewertungskriterien

Zur konsistenten Einordnung wurden drei Leitdimensionen definiert:

- **D1 Problembezug:** Wie klar werden Ursachen/Folgen von Obsoleszenz beschrieben und belegt?
- **D2 Signalqualität:** Wie wird das Kommentar-Signal operationalisiert und wie belastbar ist der Zusammenhang zu Updates?
- **D3 Umsetzbarkeit:** Wie realistisch ist die Überführung des Signals in tatsächliche Post-Änderungen (Prozess, Validierung, Akzeptanz)?

Zur Veranschaulichung werden die wichtigsten Quellen entlang dieser drei Dimensionen in Tabelle 2 eingeordnet.

Tabelle 2. EINORDNUNG ZENTRALER QUELLEN ENTLANG D1–D3 (0=NEIN, 1=TEILWEISE, 2=KLAR/EVIDENZBASIERT).

Quelle	D1	D2	D3
Zhang et al. (Obsoleszenz) [2]	2	0	1
Zhou & Walker (Deprecation) [4]	2	0	1
Fischer et al. (Copy/Paste-Risiko) [3]	2	0	1
Baltes et al. (SOTorrent) [5]	1	1	2
Soni & Nadi (comment-induced updates) [7]	1	2	1
Sheikhaei et al. (URC) [6]	1	2	1
Mai et al. (SOUP) [10]	2	2	2
Kabir et al. (LLM vs. Stack Overflow) [8]	1	0	1
Del Rio-Chanona et al. (Knowledge Sharing) [9]	1	0	1

4. Dokumentenbasierte Analyse: Automatisiertes Aktualisieren von Stack-Overflow-Posts (SOUP)

4.1. Problemstellung und Überblick

Im Mittelpunkt dieser Analyse steht SOUP, ein LLM-basiertes Framework, das Code-Snippets in Stack-Overflow-Antworten anhand der zugehörigen Kommentare automatisiert aktualisiert [vgl. 10]. Damit adressiert der Ansatz eine zentrale Lücke der Plattformpflege: Kommentare enthalten häufig konkrete Hinweise auf Fehler, Obsoleszenz oder Qualitätsprobleme, werden jedoch in der Praxis oft nicht in den eigentlichen Antworttext übernommen [vgl. 10]. Konzeptionell zerlegt SOUP das Problem des „automatischen Post-Updating“ in zwei aufeinander aufbauende Phasen (VCP und APU), die in Abbildung 1 schematisch dargestellt sind [vgl. 10].

Zunächst wird geprüft, ob ein Kommentar tatsächlich eine spätere Änderung (Edit) ausgelöst hat (VCP, Valid Comment-Edit Pair Prediction) [vgl. 10]. Dazu werden Kommentar–Edit-Paare, wie im oberen Teil von Abbildung 1 dargestellt, gefiltert und klassifiziert [vgl. 10]. Im

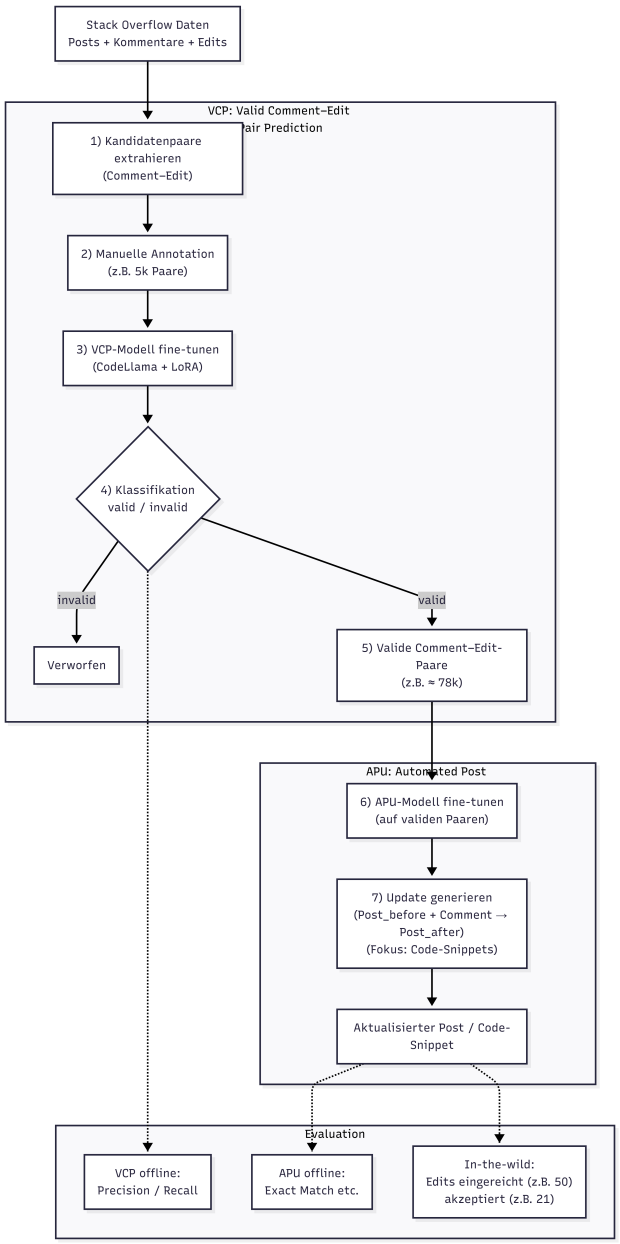

Abbildung 1. SOUP-Pipeline: VCP als Qualitätsfilter und APU zur Update-Generierung (eigene Darstellung basierend auf [10]).

Anschluss wird aus Kommentar und ursprünglichem Post eine konkrete Aktualisierung erzeugt (APU, Automated Post Update), sodass der aktualisierte Post beziehungsweise das Code-Snippet entsteht, wie im unteren Teil der Grafik skizziert [vgl. 10].

4.2. Aufgaben und Datengrundlage (VCP/APU)

4.2.1. Aufgabe 1: VCP als Qualitätsfilter für Kommentar–Edit-Paare. Die Phase VCP modelliert die Frage, ob ein gegebenes Kommentar–Edit-Paar als *valid* einzustufen ist [vgl. 10]. Als valide gelten Paare nur dann, wenn (i) der Kommentar für das Edit relevant ist, (ii) das Edit die im Kommentar angesprochenen Punkte tatsächlich adressiert und (iii) das Edit keine zusätzlichen, nicht durch den Kommentar motivierten Änderungen einführt [vgl. 10]. Zur Umsetzung werden Kommentar–Edit-Paare manuell annotiert und anschließend ein LLM (CodeLlama) mittels LoRA feinjustiert, um valide von nicht-validen Paaren zu unterscheiden [vgl. 10]. Auf dieser Grundlage entsteht ein großskaliger, qualitätsgefilterter Korpus an validen Kommentar–Edit-Paaren [vgl. 10].

4.2.2. Aufgabe 2: APU als Generierungsproblem. Aufbauend auf VCP ist APU als Generierungsaufgabe definiert [vgl. 10]. Gegeben sind ein Post *vor* der Änderung sowie ein zugehöriger Kommentar; das Ziel besteht darin, den Post *nach* der Änderung zu erzeugen [vgl. 10]. Der Ansatz fokussiert dabei die Aktualisierung von Code-Snippets und nutzt einen Datensatz mit rund 78000 validen Paaren für Java, die aus Stack-Overflow-Edit-Historien und Kommentaren abgeleitet und durch VCP qualitätsgesichert werden [vgl. 10].

4.3. Evaluation und zentrale Ergebnisse

4.3.1. Offline-Evaluation: VCP und APU. In der Offline-Evaluation erreicht das für SOUP trainierte VCP-Modell eine Präzision von 80,8% sowie einen Recall von 74,0% [vgl. 10]. Für APU wird unter anderem eine Exact-Match-Rate von 25,6% berichtet [vgl. 10].

4.3.2. In-the-wild-Evaluation: Praktische Umsetzbarkeit auf Stack Overflow. Zusätzlich wird eine In-the-wild-Evaluation direkt auf Stack Overflow durchgeführt [vgl. 10]. Dabei werden 50 von SOUP erzeugte Edits manuell eingereicht; 21 dieser Änderungen werden von Maintainers verifiziert und akzeptiert [vgl. 10].

4.3.3. Einordnung. Insgesamt deuten die Ergebnisse darauf hin, dass automatisierte Aktualisierung vor allem dann praktikabel ist, wenn (i) Kommentare zuverlässig mit konkreten Edit-Intentionen verknüpft werden können und (ii) die Generierung auf einem hinreichend sauberen, domänenspezifischen Datensatz basiert [vgl. 10]. Entsprechend ist SOUP nicht als Ersatz menschlicher Wartung zu verstehen, sondern als Mechanismus, der Wartungsbedarf sichtbar macht und Vorschläge liefert, die anschließend im Community-Prozess validiert werden [vgl. 10].

5. Diskussion

5.1. Warum Plattformpflege trotz LLMs relevant bleibt

Die Ergebnisse deuten darauf hin, dass Plattformpflege auf Stack Overflow auch im Zeitalter von LLMs relevant bleibt. Zum einen bleibt Obsoleszenz ein strukturelles Problem: Bibliotheken, Frameworks und APIs entwickeln sich weiter, sodass ehemals korrekte Antworten veralten [vgl. 2, 4]. Darüber hinaus zeigt die empirische Untersuchung von Zhang et al.dass veraltete Antworten verbreitet sind und nur ein relativ kleiner Anteil nachträglich aktualisiert wird [2].

Zum anderen verstärkt das Nutzungsmuster die praktische Relevanz: Code-Snippets werden häufig direkt übernommen. Die Studie von Fischer et al. zeigt, dass Copy-and-Paste von Stack-Overflow-Code in reale Software weit verbreitet ist und sicherheitsrelevante Risiken

übertragen kann [3]. Hinzu kommt, dass LLMs Plattformpflege nicht ersetzen, sondern die Rahmenbedingungen verändern. Die Studie von Kabir et al. vergleicht LLM-Antworten mit Stack-Overflow-Antworten und berichtet, dass LLM-Ausgaben zwar nützlich sein können, jedoch relevante Qualitäts- und Verlässlichkeitsprobleme aufweisen [8].

Die Arbeit von Del Rio-Chanona et al. zeigt Effekte auf die öffentliche Wissensweitergabe, wodurch langfristig die Aktualität frei verfügbarer Wissensbestände gefährdet werden kann [9]. Damit bleibt eine gepflegte und aktualisierte Wissensbasis auf Plattformen wie Stack Overflow weiterhin wichtig – auch dann, wenn LLMs in der Praxis intensiv genutzt werden.

5.2. Implikationen: Hybrides Modell (KI + menschliche Validierung)

Die Fallstudie SOUP legt nahe, dass ein hybrides Pflegekonzept praktikabler ist als eine vollständig automatisierte Aktualisierung. In der SOUP-Studie zeigt sich, dass kommentarbasiertes Post-Updating technisch umsetzbar ist und in realen Plattformprozessen Wirkung entfalten kann; zugleich bleibt menschliche Prüfung zentral, um Kontext, Korrektheit und Plattformstandards sicherzustellen [vgl. 10]. Automatisierung kann dabei helfen, Wartungsbedarf sichtbar zu machen und konkrete Vorschläge zu erzeugen, ersetzt aber nicht die abschließende Entscheidung durch die Community. Daraus lässt sich ableiten, dass LLM-basierte Systeme vor allem als Vorschlags- und Priorisierungshilfe geeignet sind, während Verantwortung und finale Entscheidung bei Menschen verbleiben sollten.

5.3. Praktische Empfehlungen für Community-Plattformen

Im Folgenden werden drei Empfehlungen (R1–R3) formuliert, die sich aus der Analyse der Literatur und der SOUP-Fallstudie ableiten.

5.3.1. (R1) Kommentar-Signale gezielt in Pflege-Workflows überführen. Kommentare mit Update-Charakter sollten (semi-)automatisch erkannt und in Wartungsaufgaben überführt werden. Die Analysen zu comment-induced updates und URCs zeigen, dass Kommentare häufig konkrete Änderungsbedarfe identifizieren und mit späteren Aktualisierungen von Antworten zusammenhängen [vgl. 6, 7]. Plattformen könnten solche Kommentar-Signale explizit nutzen, um potenziell veraltete oder fehlerhafte Beiträge zu markieren und in Review-Prozesse einzuspeisen.

5.3.2. (R2) Automatisierte Vorschläge nur mit Qualitätsgates und Review-Pflicht. Aus der SOUP-Fallstudie wird die Bedeutung von Qualitätsgates wie VCP für saubere Datensätze und belastbare Vorschläge deutlich [vgl. 10]. Automatisierte Aktualisierung sollte daher nur mit vorgelagerten Qualitätsfiltern und einer klaren Review-Pflicht durch menschliche Nutzerinnen und Nutzer eingesetzt werden. Gerade in sicherheitsrelevanten Kontexten ist eine konservative Qualitätssicherung entscheidend, da unsichere Codebeispiele reale Risiken in Anwendungen erzeugen können [vgl. 3].

5.3.3. (R3) Anreize und UI/UX-Unterstützung für Wartung erhöhen. Da ein wesentlicher Anteil veralteter Antworten nicht aktualisiert wird, sind Anreiz- und Sichtbarkeitsmechanismen für Wartung besonders relevant [vgl. 2]. Unterstützend wirken beispielsweise klare Edit-Guidelines [vgl. 11, 17]; ergänzend kommen sichtbare Wartungsqueues sowie die Belohnung qualitativ geprüfter Updates (z.B. Badges oder Reputation) in Frage. Im LLM-Zeitalter ist dies besonders wichtig, wenn öffentliche Aktivität und Wissensteilung in Communities tendenziell abnehmen [vgl. 9].

6. Limitationen

6.1. Generalisierbarkeit und Bias

Die Ergebnisse sind durch mehrere Faktoren begrenzt. Erstens basiert die Arbeit auf einer literaturbasierten Synthese mit einer fokussierten Fallstudie, sodass Befunde zu Obsoleszenz und kommentarbasierten Update-Signalen zusammengeführt werden [vgl. 2, 6, 7]. Die Einordnung bleibt jedoch abhängig von der verfügbaren Evidenz und von der Auswahl der betrachteten Arbeiten. Im Rahmen literaturbasierter Synthesen sind Zitations- und Publikationsbias möglich [vgl. 18, 19].

Zweitens ist Obsoleszenz in der Literatur nicht einheitlich operationalisiert. Die Studie von Zhang et al. betrachtet unterschiedliche Formen von Obsoleszenz [2], während die Analyse von Zhou und Walker Obsoleszenz über API-Deprecation adressiert [4]. Dies erschwert die direkte Vergleichbarkeit der Ergebnisse.

Drittens ist die betrachtete Evidenz teilweise domänenspezifisch. Die Studie von Fischer et al. fokussiert sicherheitsrelevante Android-Szenarien [3], und SOUP ist in dieser Arbeit primär als Java-zentrierter Ansatz relevant [vgl. 10]. Dies schränkt die Übertragbarkeit auf sprachempfindliche Sprachen (wie Python) ein, bei denen syntaktische Strukturen durch Einrückungen definiert werden und Syntaxfehler durch LLM-generierte Edits wahrscheinlicher sind. Viertens kann Selektionsbias in der Signalquelle „Kommentar“ auftreten, da nicht alle Posts in vergleichbarer Weise kommentiert werden [vgl. 6, 7].

6.2. Abhängigkeit von Daten, Modellen und Kontext

Empirische Analysen und automatisierte Pflegeansätze setzen voraus, dass Post-Evolution und Kommentarhistorie zuverlässig rekonstruierbar sind. SOTorrent bietet hierzu eine etablierte Datenbasis [5]. Die Reproduzierbarkeit bleibt jedoch vom Datenzugang und der Bereitstellung abhängig [vgl. 12, 13, 14]. Zudem unterliegt die Nutzung Stack-Overflow-spezifischen Lizenzbedingungen [vgl. 13, 15, 16].

Ansätze wie SOUP hängen von Modellwahl, Trainingsdaten und Fine-tuning ab [vgl. 10]. Der LLM-Kontext ist zudem zeitlich volatil. Die betrachtete Studie nutzt CodeLlama; leistungsfähigere Modelle (z.B. GPT-4o) könnten die berichtete Performance drastisch erhöhen, was die Vergleichbarkeit aktueller Benchmarks einschränkt. Qualitätsgrenzen bleiben dennoch relevant

[vgl. 8], und Anreizverschiebungen können die öffentliche Wissensproduktion beeinflussen [vgl. 9].

7. Fazit und Ausblick

Diese Arbeit untersuchte, wie Kommentare auf Stack Overflow als Signalquelle genutzt werden können, um veraltete oder fehlerhafte Antworten systematisch zu aktualisieren. Der Forschungsstand zeigt, dass Obsoleszenz auf der Plattform verbreitet ist und Aktualisierungen nur begrenzt erfolgen [vgl. 2]. API-Evolution und Copy-and-Paste verstärken praktische Risiken, einschließlich sicherheitsrelevanter Fehlverwendungen [vgl. 3, 4]. Kommentare liefern dabei ein verwertbares Wartungssignal, etwa über comment-induced updates und URCs [vgl. 6, 7].

Die Fallstudie SOUP zeigt, dass kommentarbasiertes Post-Updating technisch operationalisierbar ist und in realen Plattformprozessen Wirkung entfalten kann [vgl. 10]. Robuste Plattformpflege erfordert jedoch hybride Prozesse: Automatisierung liefert Priorisierung und Vorschläge, während Menschen Prüfung, Kontextvalidierung und Verantwortung übernehmen [vgl. 10]. Plattformpflege bleibt auch im LLM-Umfeld zentral, weil LLM-Antworten nicht durchgängig verlässlich sind [vgl. 8] und weil Anreizverschiebungen die öffentliche Wissensweitergabe reduzieren können [vgl. 9].

Zukünftige Arbeiten sollten (i) die Kommentar–Edit-Kausalität robuster modellieren [vgl. 10], (ii) die Übertragbarkeit über Sprachen und Domänen evaluieren [vgl. 4, 10], (iii) sicherheitskritische Updates priorisieren [vgl. 3] und (iv) hybride Workflows stärker in die Plattform-UI/UX integrieren [vgl. 11, 17].

Abkürzungsverzeichnis

API	Application Programming Interface
APU	Automated Post Update
CCBYSA	Creative Commons Attribution-ShareAlike
ICSE	International Conference on Software Engineering
KI	Künstliche Intelligenz
LLM	Large Language Model
LoRA	Low-Rank Adaptation
Q&A	Question and Answer
SOTorrent	SOTorrent-Datensatz (Evolution von Stack-Overflow-Posts)
SOUP	Stack Overflow Updater for Post
UI	User Interface
URC	Update Request Comment
UX	User Experience
VCP	Valid Comment-Edit Pair Prediction

Literatur

[1] Stack Overflow. „Stack Overflow Developer Survey 2023“, besucht am 20. Dez. 2025. Adresse: https://survey.stackoverflow.co/2023

[2] H. Zhang, S. Wang, T.-H. Chen, Y. Zou und A. E. Hassan, „An Empirical Study of Obsolete Answers on Stack Overflow“, *Empirical Software Engineering*, Jg. 24, Nr. 3, S. 1295–1333, 2019. Adresse: https://doi.org/10.1007/s10664-018-9667-8

[3] F. Fischer et al., „Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security“, in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, S. 121–136. Adresse: https://doi.org/10.1109/SP.2017.31

[4] J. Zhou und R. J. Walker, „API Deprecation: A Retrospective Analysis and Detection Method for Code Examples on the Web“, in *Proceedings of the 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE '16)*, ACM, 2016, S. 266–277. Adresse: https://doi.org/10.1145/2950290.2950298

[5] S. Baltes, L. Dumani, C. Treude und S. Diehl, „SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts“, in *Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18)*, ACM, 2018, S. 319–330. Adresse: https://doi.org/10.1145/3196398.3196430

[6] M. S. Sheikhaei, Y. Tian und S. Wang, „A study of update request comments in Stack Overflow answer posts“, *Journal of Systems and Software*, Jg. 198, S. 111 590, 2023. Adresse: https://doi.org/10.1016/j.jss.2022.111590

[7] A. Soni und S. Nadi, „Analyzing Comment-Induced Updates on Stack Overflow“, in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19)*, IEEE/ACM, 2019, S. 220–234. Adresse: https://doi.org/10.1109/MSR.2019.00044

[8] S. Kabir, D. N. Udo-Imeh, B. Kou und T. Zhang, „Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions“, in *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*, ACM, 2024, S. 1–17. Adresse: https://doi.org/10.1145/3613904.3642596

[9] R. M. del Rio-Chanona, N. Laurentsyeva, J. A. List, P. A. Novosad, A. Samek und K. Wüthrich, „Large language models reduce public knowledge sharing on online communities“, *PNAS Nexus*, Jg. 3, Nr. 9, pgae400, 2024. Adresse: https://doi.org/10.1093/pnasnexus/pgae400

[10] Y. Mai et al., „Towards Better Answers: Automated Stack Overflow Post Updating“, in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, IEEE/ACM, 2025, S. 591–603. Adresse: https://doi.org/10.1109/ICSE55347.2025.00024

[11] Stack Overflow. „Why can people edit my posts? How does editing work?“, besucht am 20. Dez. 2025. Adresse: https://stackoverflow.com/help/editing
[12] Stack Overflow Blog. „Stack Exchange Creative Commons data now hosted by the Internet Archive“, besucht am 20. Dez. 2025. Adresse: https://stackoverflow.blog/2014/01/23/stack-exchange-cc-data-now-hosted-by-the-internet-archive/
[13] Stack Exchange. „Stack Exchange Data Dump license.txt“. License file shipped with Stack Exchange data dumps (hosted by Internet Archive), besucht am 20. Dez. 2025. Adresse: https://archive.org/download/stackexchange/license.txt
[14] Stack Overflow Blog. „You Can Now Play with Stack Overflow Data on Google's BigQuery“, besucht am 20. Dez. 2025. Adresse: https://stackoverflow.blog/2016/12/15/you-can-now-play-with-stack-overflow-data-on-googles-bigquery/
[15] Stack Overflow. „What is the license for the content I post?“, besucht am 20. Dez. 2025. Adresse: https://stackoverflow.com/help/licensing
[16] Meta Stack Exchange. „Stack Exchange and Stack Overflow have moved to CC BY-SA 4.0“, besucht am 20. Dez. 2025. Adresse: https://meta.stackexchange.com/questions/333089/stack-exchange-and-stack-overflow-have-moved-to-cc-by-sa-4-0
[17] Stack Overflow. „Comment everywhere“, besucht am 20. Dez. 2025. Adresse: https://stackoverflow.com/help/privileges/comment
[18] C. Wohlin, „Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering“, in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, ACM, 2014. Adresse: https://doi.org/10.1145/2601248.2601268
[19] B. Kitchenham und S. Charters, „Guidelines for Performing Systematic Literature Reviews in Software Engineering“, Keele University und Durham University Joint Report, Techn. Ber. EBSE-2007-01, 2007. Adresse: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf

Anhang

Für die sprachliche Überarbeitung, Strukturierung sowie gelegentliche Unterstützung bei der Formulierung wurde das KI-basierte Sprachmodell ChatGPT (OpenAI, Modell GPT-4) herangezogen. Alle Modellvorschläge wurden kritisch geprüft und bei Bedarf angepasst. Die Verantwortung für Inhalt, Quellenarbeit und Argumentation liegt vollständig beim Autor. Nachfolgend sind beispielhafte Eingaben (Prompts) aufgeführt, die im Verlauf der Arbeit verwendet wurden:

- „Bitte formuliere den folgenden Abschnitt wissenschaftlicher, klarer und ohne Redundanz, ohne den Inhalt zu verändern: [TEXT]“
- „Bitte korrigiere den folgenden wissenschaftlichen Text hinsichtlich Grammatik und Stil: [TEXT]“
- „Ordne den folgenden Abschnitt so um, dass ein klarer roter Faden entsteht (Problem → Evidenz → Schluss). Keine neuen Behauptungen: [TEXT]“
- „Passe den folgenden Text so an, dass er in ein IEEEtran-Konferenzpaper passt (kurze Sätze, präzise Übergänge). Keine neuen Quellen/Fakten: [TEXT]“