УДК 004.8

Вонсович Б.А., Багрій Р.О., Пасічник О.А., Скрипник Т.К.

Хмельницький національний університет

МЕТОД ВИЯВЛЕННЯ НЕОДНОЗНАЧНОСТЕЙ У ВИМОГАХ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Запропоновано метод автоматизованого аналізу вимог до програмного забезпечення, що базується на двоагентній LLM-архітектурі. Підхід вирішує проблему низької ефективності прямих запитів, розділяючи процес на два етапи: ідентифікацію потенційних неоднозначностей та їх подальшу глибоку класифікацію з поясненням. В основі методу лежить технологія доповненої генерації (RAG) для надання агентам контекстуальних знань та використання промптингу ланцюжок думок (CoT) для підвищення прозорості та точності результатів.

A method for the automated analysis of software requirements based on a two-agent LLM architecture is proposed. This approach addresses the problem of low efficiency in direct queries by dividing the process into two stages: identification of potential ambiguities and their subsequent deep classification with explanation. The method is based on Retrieval-Augmented Generation (RAG) technology to provide agents with contextual knowledge and the use of Chain-of-Thought (CoT) prompting to increase transparency and accuracy of results.

Якість текстових вимог до програмного забезпечення є критичною для успіху проєкту – неоднозначність у формулюваннях призводить до неправильної інтерпретації, помилок у дизайні та значного зростання витрат на виправлення дефектів [1, 2].

Сучасні великі мовні моделі (LLM) демонструють високий потенціал у обробці природної мови, але дослідження підтверджують, що в задачах, які вимагають багатокрокового міркування, їхня ефективність без структурованих підходів суттєво знижується [3].

У задачах аналізу вимог до ПЗ необхідно не лише виявляти потенційно неоднозначні фрагменти тексту, а й класифікувати тип неоднозначності та оцінювати її вплив на якість вимоги. Практика показує, що виконання всіх етапів аналізу одним запитом до LLM призводить до зниження точності, нестабільності результатів і втрати інформативності [4]. Розділення процесу на послідовні етапи з чітко визначеними ролями є доцільним підходом. Відсутність такого поділу з високою ймовірністю спричиняє або надмірну кількість хибних спрацьовувань на

етапі виявлення, або поверховий аналіз під час класифікації [5]. Запропонований двоетапний підхід усуває ці недоліки: поділ завдань знижує когнітивне навантаження на модель, підвищує точність і забезпечує прозорість процесу.

В основі методу (рис. 1) лежить технологія доповненої генерації (RAG). Кожен агент у системі використовує RAG для отримання динамічних прикладів, що є критично важливим для точного аналізу вимог. Замість статичних прикладів, RAG-система активно шукає у спеціалізованому наборі даних найбільш релевантну інформацію для поточного завдання. Ця інформація динамічно додається до промпту агента, що є ефективною формою навчання в контексті.

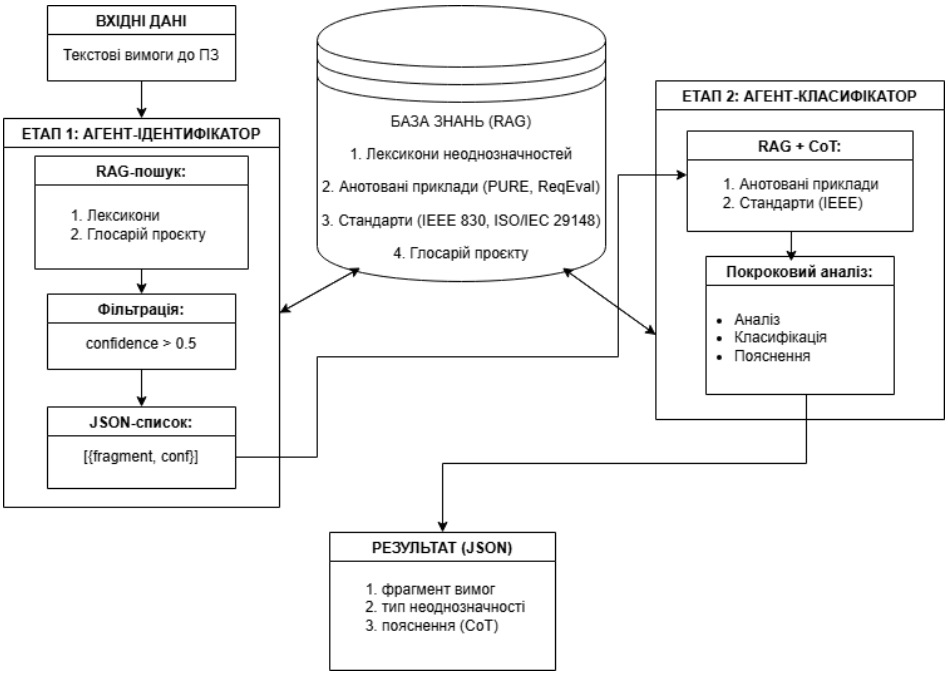

Рисунок 1 – Схема методу виявлення неоднозначності у вимогах

Ефективність системи залежить від якості наповнення цієї бази знань. Для задачі аналізу вимог база знань має бути структурована та містити декілька ключових компонентів. По-перше, це лексикони неоднозначностей – куровані списки слів та фраз, які часто вказують на проблеми (наприклад, «легко», «швидко», «ефективно», «тощо»). По-друге, це анотовані приклади – реальні речення з вимог, розмічені експертами. Для цього можуть бути використані публічні бенчмарки, такі як PURE або спеціалізовані набори даних, як-от ReqEval,

що фокусується на референційній неоднозначності. По-третє, база знань повинна містити стандарти та принципи – витяги з академічних джерел та галузевих стандартів (наприклад, IEEE 830), що описують характеристики якісних вимог, зокрема верифікованість. Нарешті, найважливішим компонентом є база знань проекту/домену – внутрішні глосарії, специфікації та акроніми, специфічні для конкретного проекту, що дозволяє системі адаптуватися до контексту та зменшити кількість помилкових спрацьовувань.

На першому етапі функціонує Агент-ідентифікатор, роль якого полягає в оцінці ризику неоднозначності. Його завдання – провести первинний аналіз тексту вимог та ідентифікувати фрагменти, що мають високий потенціал неоднозначності. Агент виконує роль QA-аналітика з метою ранжування виявлених фраз за шкалою впевненості (від 0.0 до 1.0). Для виконання цього завдання агент використовує технологію RAG, звертаючись до двох спеціалізованих наборів даних: лексикону неоднозначностей та база знань проекту.

Ключова перевага такого підходу полягає у можливості фільтрації. Встановлення порогового значення (наприклад, confidence > 0.5) дозволяє відсіяти шум – кандидати з низькою впевненістю. Це ефективно вирішує поширену проблему автоматизованих аналізаторів, пов'язану з низькою точністю та перевантаженням системи помилковими спрацьовуваннями. Результатом цього етапу є відфільтрований JSON-список кандидатів з високою впевненістю (наприклад, [{"fragment": "швидко реагувати", "confidence": 0.9}]), який передається на наступний, обчислювально дорожчий етап.

На другому етапі в роботу вступає Агент-Класифікатор. Він отримує на вхід відфільтрований список кандидатів і проводить їх глибокий аналіз. Агент діє в ролі експерта з інженерії вимог, що знайомий з галузевими стандартами, такими як IEEE 830. Робота агента базується на методології ланцюжка думок (CoT), яка вимагає від моделі генерації покрокового обґрунтування для кожного рішення. Процес CoT складається з трьох обов'язкових кроків:

Аналіз – ідентифікація конкретного терміну, фрази чи граматичної структури, що спричиняє проблему.

Класифікація – присвоєння точного типу неоднозначності згідно з попередньо визначеною класифікацією неоднозначностей.

Пояснення – формулювання чіткого обґрунтування, чому фраза є неоднозначною в даному контексті, та опис потенційних негативних наслідків (наприклад, «неможливість тестування», «ризик неправильної реалізації»).

Використання CoT є вирішальним, оскільки дослідження показують, що прості запити до LLM можуть не дати адекватного пояснення , тоді як CoT змушує модель генерувати покрокову логіку. Простий тег [Туманність] може бути проігнорований розробником. Однак, покрокове пояснення (CoT), яке логічно

обґрунтовує висновок і посилається на галузеві принципи (наприклад, «не підлягає тестуванню» згідно IEEE 830), демонструє компетентність системи та значно підвищує довіру до її результатів. Для підвищення точності аналізу цей агент також використовує RAG, але вже з іншими джерелами – анотованими прикладами та стандартами. Фінальним результатом роботи системи є структурований JSON-об'єкт з повним детальним аналізом кожної виявленої неоднозначності.

У роботі запропоновано двоагентний LLM-метод для аналізу вимог до ПЗ, який вирішує проблему низької ефективності прямих запитів до моделей. Ключем є декомпозиція задачі: Агент 1 (Ідентифікатор) проводить ефективну фільтрацію "шуму", ранжуючи проблеми за впевненістю, тоді як Агент 2 (Класифікатор) проводить глибокий аналіз відфільтрованих кандидатів. Контекстуальна точність досягається через RAG, що надає агентам доступ до спеціалізованих баз знань (лексикони, стандарти IEEE 830, знання проекту). Прозорість та довіра забезпечуються ланцюжком думок, який змушує другого агента генерувати покрокові пояснення замість простих міток. Таким чином, запропонований метод перетворює LLM на надійний, прозорий та адаптивний інструмент для підвищення якості вимог.

Перелік посилань

1. Femmer, H., Fernández, D. M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with Requirements Smells. Journal of Systems and Software, 123, 190–213. https://doi.org/10.1016/j.jss.2016.02.047
2. Berry, D. M., & Kamsties, E. (2004). Ambiguity in Requirements Specification. In Perspectives on Software Requirements (pp. 7–44). Springer. https://doi.org/10.1007/978-1-4615-0465-8_2
3. Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS 2022. https://arxiv.org/abs/2201.11903
4. Yao, S., Zhao, J., Yu, D., et al. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023. https://arxiv.org/abs/2210.03629
5. Kojima, T., Gu, S. S., Reid, M., et al. (2022). Large Language Models are Zero-Shot Reasoners. NeurIPS 2022. https://arxiv.org/abs/2205.11916