Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Автоматичне генерування вимог до програмного забезпечення на основі зворотного зв'язку від користувачів із застосуванням алгоритмів обробки природної мови (NLP)
(тема)

Виконав:
здобувач _2_ року навчання, групи _САУм-23-2_
Свинкін Д.О.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна_
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз і управління
(повна назва освітньої програми)
Керівник доц. Ситникова Ю.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _124 Системний аналіз
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ _25_ ” _листопада_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Свинкін Данилу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Автоматичне генерування вимог до програмного забезпечення на основі зворотного зв’язку від користувачів із застосуванням алгоритмів обробки природної мови (NLP)
затверджена наказом по університету від 22 листопада 2024 р. № 1228 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 10 січня 2025 р.

3. Вихідні дані до роботи _відгуки користувачів програмного забезпечення

4. Перелік питань, що потрібно опрацювати в роботі
1. Системний аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	25 листопада – 1 грудня 2024 р.	виконано
2	Вибір та обґрунтування методу	2 – 8 грудня 2024 р.	виконано
3	Розробка алгоритму і програми	9 – 22 грудня 2023 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	23 – 29 грудня 2024 р.	виконано
5	Робота над текстом пояснювальної записки	30 грудня 2024 р. – 9 січня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2025 р.	виконано

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Ситникова Ю.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 57 с., 8 рис., 1 дод., 10 джерел.

АВТОМАТИЗАЦІЯ ФОРМУВАННЯ ВИМОГ, ОБРОБКА ПРИРОДНОЇ МОВИ, ACCEPTANCE CRITERIA, BERT, GPT, USER STORY.

Об’єкт дослідження – автоматизація збору та генерації вимог до програмного забезпечення на основі зворотного зв’язку користувачів.

Мета роботи – дослідити можливості застосування алгоритмів обробки природної мови для автоматичного генерування вимог до програмного забезпечення на основі зворотного зв’язку від користувачів шляхом генерації списку вимог у визначеному форматі, які будуть відображати потреби користувачів на основі їх відгуків.

Методи дослідження – аналіз тексту з використанням обробки природної мови (NLP), машинне навчання для класифікації та пріоритизації вимог, а також генеративні моделі для створення специфікацій.

Розроблено інтегровану систему автоматизованої обробки зворотного зв’язку користувачів, яка генерує вимоги до програмного забезпечення з використанням сучасних методів NLP, машинного навчання та генеративних моделей.

Розроблена система може бути впроваджена в IT-компаніях, які займаються розробкою програмного забезпечення, для автоматизації аналізу зворотного зв’язку та формування вимог. Вона сприяє підвищенню ефективності команд розробників, зменшенню людських помилок та забезпеченню прозорості у формуванні вимог.

ABSTRACT

Introductory note: 57 pages, 8 figures, 1 appendix, 10 sources.

ACCEPTANCE CRITERIA, AUTOMATIC GENERATION OF SOFTWARE REQUIREMENTS, BERT, GPT, NATURAL LANGUAGE PROCESSING, USER STORY.

Object of research – the automation of software requirements generation based on user feedback.

The purpose of this work is to study the possibility of applying natural language processing algorithms to automatically generate software requirements based on user feedback by generating a list of requirements in a defined format that will reflect user needs based on their feedback.

Methods of research are text analysis using natural language processing (NLP), machine learning for requirements classification and prioritization, and generative models for creating specifications.

The integrated system for the automated processing of user feedback generates software requirements using modern NLP methods, machine learning, and generative models.

The developed system can be used in software development IT companies to automate feedback analysis and requirements generation. It helps to increase the efficiency of development teams, reduce human errors, and ensure transparency in the formation of requirements.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

User Story – неформальне загальне пояснення функцій програмного забезпечення, написане з точки зору кінцевого користувача;

Acceptance Criteria – чітко визначені умови, які повинні бути виконані, щоб вважати результат завершеним та готовим до використання.

NLP – natural language processing. Загальний напрям інформатики, штучного інтелекту та математичної лінгвістики, що вивчає проблеми комп’ютерного аналізу та синтезу природної мови.

GPT – generative pre-trained transformer. Штучна нейронна мережа, яка використовується для обробки природної мови;

BERT – bi-directional representations of text. Модель обробки природної мови, яка вивчає подвійно спрямоване представлення тексту.

ВСТУП

Актуальність теми. Актуальність роботи зумовлена необхідністю обробки великого обсягу даних, пов’язаних з програмним забезпеченням (ПЗ) у найрізноманітніших сферах – від охорони здоров’я до маркетингу і брендингу. Одним з джерел таких даних є зворотний зв’язок. Відгуки можуть містити важливу інформацію щодо проблем та можливих покращень програмного забезпечення (ПЗ). Безсумнівно, автоматичне генерування вимог на основі цих відгуків із застосуванням алгоритмів обробки природної мови дозволить:

– зменшити кількість ресурсу, необхідного для обробки великого обсягу даних;

– збільшити швидкість обробки цих даних;

– зменшить час, необхідний для реагування на нагальні потреби користувачів.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є дослідження можливості застосування алгоритмів обробки природної мови для автоматичного генерування вимог до програмного забезпечення на основі зворотного зв’язку від користувачів шляхом генерації списку вимог у визначеному форматі, які будуть відображати потреби користувачів на основі їх відгуків. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасних методів та підходів обробки природної мови, які можна використати для аналізу зворотного зв’язку;

– проаналізувати існуючі моделі для автоматизованої генерації вимог;

– визначити моделі глибинного навчання, які будуть використовуватись для аналізу зворотного зв’язку та генерації вимог;

– підсумувати результати щодо визначеного підходу та сформулювати подальші кроки для його застосування;

– визначити архітектуру для вирішення поставленої задачі;

– розробити програмний продукт, який дозволить автоматизувати процес генерації вимог на основі відгуків користувачів;

– провести аналіз отриманих результатів.

Об'єктом дослідження є автоматизація збору та генерації вимог до програмного забезпечення на основі зворотного зв'язку користувачів.

Предметом дослідження є методи та алгоритми обробки природної мови, а саме моделі глибинного навчання для автоматичної генерації вимог до програмного забезпечення на основі відгуків користувачів.

Методи дослідження. У роботи використовуються методи машинного навчання.

Публікації. Результати, отримані у роботі, було представлено на 3-й Міжнародній науково-практичній конференції «Навчання і викладання: у світі після війни» (м. Харків, 8 листопада 2024 р.) [1].

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз задачі автоматичного генерування вимог до ПЗ на основі зворотного зв’язку від користувачів із застосуванням алгоритмів обробки природної мови

Під час розробки програмного забезпечення одним з найважливіших етапів є збір та визначення конкретних вимог, які будуть відповідати бізнес ідеї та задовольняти нагальним потребам користувачів. Однак стандартні методи ручного збору та аналізу вимог мають багато недоліків та забирають велику кількість часу.

Ручна обробка відгуків та формування на їх основі певних вимог забирає дуже багато часу та людського ресурсу. Найбільший вплив відбувається саме на великі проєкти, в яких кількість відгуків є дуже значною. При обробці великої кількіості даних також значним ризиком є людський фактор. Через велику завантаженість, різноманітність зворотного зв’язку та суб’єктивність сприйняття відгуків кожним працівником, вимоги в кінцевому результаті можуть не повністю відповідати потребам, які були заявлені користувачами, що в свою чергу призведе до великої кількості змін на пізніх етапах розробки. Це також напряму впливає на рівень задоволеності користувачів, бо неточне інтерпретування їх потреб і очікувань або зміни, які вже не є актуальними, призводять до втрати частини ринку та зниження лояльності користувачів.

Автоматизація збору та генерування вимог із застосуванням алгоритмів обробки природної мови (NLP) дозволяє знизити вищевказані ризики, як то:

– швидкість обробки зворотного зв’язку значно збільшується, тому що алгоритми NLP можуть обробляти відгуки в режимі, близькому до реального часу, що дозволяє команді розробників максимально швидко на них реагувати;

– точність інтерпретації зворотного зв’язку забезпечується завдяки моделям глибинного навчання таким як BERT і GPT і цей автоматизований підхід

знижує кількість помилок та суб'єктивних інтерпретацій, які дуже часто виникають через людський фактор під час ручної обробки;

– єдиний стиль і формат вимог, наприклад, User Story з Acceptance Criteria, що формулюються за допомогою NLP, спрощують їх розуміння та допомагають уникати неоднозначності інтерпретування, які часто виникають при розгляді вимог, які написані різними людьми.

1.2 Аналіз сценаріїв вирішення задачі автоматичного генерування вимог до ПЗ на основі зворотного зв'язку від користувачів із застосуванням алгоритмів обробки природної мови

Для автоматичного генерування вимог до програмного забезпечення необхідно використовувати алгоритми обробки природної мови, які будуть проводити аналіз зворотного зв'язку від користувачів та перетворювати його у формалізовані вимоги до продукту. Сценарій вирішення такої задачі має включати послідовність дій, які будуть забезпечувати обробку даних та виділення ключових вимог для продукту.

Узагальнимо етапи, необхідні для автоматичного генерування вимог до ПЗ та виділимо основні кроки, що дозволять ефективно обробити зворотній зв'язок користувачів у вимоги, що будуть готові для подальшої розробки командою продукту:

а) збір зворотного зв'язку з доступних джерел, який є необхідним для забезпечення великої вибірки та різноманітності даних для більш повного розуміння потреб і очікувань користувачів;

б) попередня обробка тексту за допомогою алгоритмів NLP, що включає токенізацію, лемматизацію та виділення іменованих сутностей, дозволить привести відгуки до єдиної структури, яка буде готова для подальшої обробки і виділення ключової інформації [2];

в) формування та стандартизація вимог.

Тож, з огляду на вище викладене, розглянемо можливі сценарії обробки та формування вимог.

1.2.1 Виділення ключових вимог на базі великої кількості відгуків

Як відомо, для виділення ключових вимог проводиться збір великої кількості відгуків користувачів та за допомогою алгоритмів обробки природної мови проводиться обробка тексту, яка включає токенізацію, лемматизацію та виділення сутностей. А проведена таким чином послідовна якісна обробка тексту дозволяє виділити ключові поняття та проблеми, які часто згадуються у відгуках користувачів.

Дійсно, даний підхід дозволяє отримати основну інформацію щодо потреб користувачів, полегшує подальший аналіз та класифікацію запитів, але в той же час створює ризики втарти контексту через спрощення запитів та значно спрощує складні і великі відгуки, що може призвести до втрати важливих вимог.

Виділення ключових вимог на основі наявних відгуків дає можливість забезпечити базову структуру для подальшої роботи, але для ефективного управління вимогами також необхідно визначити пріоритет для кожної з них, базуючись на важливості змін для користувачів і загальному впливі на продукт.

1.2.2 Пріоритизація вимог користувачів

Пріоритизація вимог – це процес визначення важливості вимоги до програмного забезпечення з метою оптимального розподілу ресурсів та ефективного управління розробкою продукту [3].

Для визначення пріоритету вимог використовуються алгоритми аналізу частоти згадування та узагальнення, що дає можливість отримати список най-

більш поширеніших запитів у відгуках користувачів. Це допомагає правильно розподілити ресурси та ефективно використовувати зв’язок користувачів для прийняття стратегічних рішень щодо розрорбки ПЗ. Однак з таким підходом унікальні та корисні запити користувачів можуть бути втрачені через низьку частоту згадок і, як результат, низький пріоритет.

1.2.3 Визначення основних категорій вимог користувачів на основі класифікації зворотного зв’язку

Зворотнім зв’язком є будь-яка інформація від користувачів, яка включає досвід, проблеми та побажання щодо змін у продукті. Це одне з найважливіших джерел інформації для розробників, яке відображає реальні потреби користувачів ПЗ.

Для систематизування зворотного зв’язку можна використати підхід, який буде класифікувати відгуки по категоріям, наприклад, «нові функції», «вдосконалення існуючого функціоналу», «помилки» і т.д. Такий підхід, дійсно, дозволить швидко перенаправити оброблені відгуки відповідній команді, особливо у випадках великих продуктів і команд, а також сприяє полегшенню пріоритизації виділених вимог. Як і вище описані сценарії, даний підхід має ризики у випадках великих та складних відгуків, які містять багато інформації та окремих вимог, що може призвести до потреби у додатковій ручній обробці.

1.2.4 Генерування вимог у визначеному форматі

Генерування вимог у вже визначеному форматі (User Story) потребує використання моделей по типу GPT для формування конкретних історій користувачів. Чіткі та формалізовані вимоги дозволяють максимально швидко підготувати їх для подальшої імплементації розробниками, а також полегшує розумін-

ня ідеї та користі від запропонованих змін. Даний підхід є доволі складним в реалізації та вимагає додаткового налаштування моделі і більш докладної підготовки даних.

Кожен із сценаріїв має свої переваги та недоліки, зокрема, специфіка проєкту, величина доступних ресурсів та обсяг відгуків випливає на вибір підходу. Таким чином, у більшості випадках є сенс комбінувати декілька сценаріїв для досягнення максимального результату.

1.3 Формальна та змістовна постановка задачі

Для підтримки та вдосконалення програмного забезпечення, яке має велику базу користувачів, необхідно швидко і якісно обробляти зворотні відгуки для підтримки стабільного функціонування ПЗ та визначення нових функцій. Таким чином, метою дослідження є аналіз і розробка підходу для автоматичного генерування вимог на базі відгуків, використовуючи алгоритми NLP. З огляду на вище викладене, вимоги повинні бути точними, повними, однозначними, відповідати очікуванням кінцевих користувачів, готовими до розробки [3].

1.3.1 Вхідні дані

Позначимо відгуки користувачів як набір V

$$V = \{v_1, v_2, ..., v_n\}, \tag{1.1}$$

де v_i – окремий відгук.

Кожен відгук має наступні характеристики:

– $C(v_i)$ – категорія відгку v_i, яка може приймати значення з множини категорій «помилка», «функціональний запит», «запит на покращення», «безпе-

ка», «швидкодія»;

– $T(v_i)$ – тональність відгуку v_i, яка може приймати значення «позитивна», «нейтральна», «негативна»;

– $S(v_i)$ – набір іменованих сутностей відгуку v_i.

1.3.2 Вихідні дані

Результатом автоматичної генерації вимог є набір вимог R

$$R = \{r_1, r_2, ..., r_n\}, \tag{1.2}$$

де r_j - окрема вимога, яка має наступні харктеристики:

– $T(r_j)$ – текст вимоги у форматі User Story (As Persona);

– $A(r_j)$ – множина умов, яка визначає успішне виконання вимоги (Acceptance Criteria);

– $C(r_j)$ – категорія вимоги, яка може приймати одне зі значень «виправлення помилки», «UX покращення», «нова функція», «покращення безпеки», «покращення продуктивності»;

– $P(r_j)$ – числове значення пріоритету вимогу від 1 (найпріоритетніший) до 5 (мінімальний пріоритет).

Одна вимога може базуватись одразу на декількох відгуках, які мають схожий запит від користувача, так само як і один відгук може генерувати декілька вимог. Тому зв’язок між вимогою та відгуками можна визначити як множина відгуків, на основі яких сформована вимога

$$F(r_j) \subseteq R, \tag{1.3}$$

а зв’язок між відгуком та вимогами як множина вимог (1.4), які були сформова-

ні на основі цього відгуку

$$X(v_i) \subseteq V\,. \tag{1.4}$$

Отже, нашою цільовою функцією буде пріоритизація вимог (1.5) за кількістю пов’язаних відгуків та їх критичністю:

$$P(R) = \sum_{r_j \in R} P(r_j) \cdot N(r_j), \tag{1.5}$$

де $P(r_j)$ – пріоритетність вимоги r_j, а $N(r_j)$ – кількість відгуків, які сформували і дотичні дод цієї вимоги.

1.4 Постановка задач дослідження

Завданням цієї роботи є генерація списку вимог у визначеному форматі, які будуть відображати потреби користувачів на основі їх відгуків. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасних методів та підходів обробки природної мови, які можна використати для аналізу зворотного зв’язку;

– проаналізувати існуючі моделі для автоматизованої генерації вимог;

– визначити моделі глибинного навчання, які будуть використовуватись для аналізу зворотного зв’язку та генерації вимог;

– підсумувати результати щодо визначеного підходу та сформулювати подальші кроки для його застосування;

– розробити систему автоматизованої обробки зворотного зв’язку користувачів, яка генерує вимоги до програмного забезпечення з використанням;

– провести аналіз результатів обчислювального експерименту та визначити подальші можливі покращення системи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Методи обробки тексту для автоматичного генерування вимог

Обробка природної мови (NLP) та машинне навчання є основними підходами для автоматизації аналізу зворотного зв'язку користувачів та генерування вимог. З розвитком технологій обробки тексту змінилися можливості, що дозволяє переходити від традиційних методів до сучасних моделей, які працюють з контекстом та семантикою. Розглянемо основні методи обробки тексту, їхні переваги та обмеження в контексті задачі автоматичного генерування вимог до програмного забезпечення.

2.1.1 Традиційні методи обробки тексту Мішок Слів та TF-IDF

Мішок слів (Bag of Words) – це метод обробки природної мови для перетворення тексту, який представляється як множина слів без урахування порядку чи контексту, які можуть використовуватися комп'ютерною програмою [2]. Цей метод передбачає перетворення тексту у вектор на основі частоти слів у тексті без урахування порядку чи контексту слів. Дійсно, він є простим у реалізації та обробці, його легко застосувати до невеликих текстів та він надає базову інформацію про частоту слів, але в той же час він ігнорує порядок та контекст слів, що унеможливлює розуміння семантичного зв'язку між словами. Наприклад, фрази «виправити помилку» та «неправильна помилка» будуть оброблені однаково. Тож, цей підхід не підходить для задачі генерування вимог через свою обмеженість у розумінні складних зворотних зв'язків і контексту відгуку користувачів.

Метод TF-IDF використовується для виділення важливих слів у документі відносно всього тексту. Він складається з двох компонентів: TF (Term Frequency) – частота появи слова в тексті; IDF (Inverse Document Frequency) –

зворотня частота документа, що відображає, наскільки рідкісне слово в загальному тексті [2]. TF-IDF обчислюється як (2.1):

$$w_{t,d} = tf_{t,d} \cdot idf, \tag{2.1}$$

де $tf_{t,d}$ – відношення кількості появ слова t у документі d до кількості слів у документі d (2.2):

$$tf_{t,d} = count(t,d); \tag{2.2}$$

idf – логарифм відношення загальної кількості документів до кількості документів, що містять слово t (2.3):

$$idf_{t,d} = \log_{10}\left(\frac{N}{df_t}\right). \tag{2.3}$$

Таким чином, TF-IDF може бути використаний для виділення важливих слів або фраз, але він недостатній для формування повноцінних вимог.

2.1.2 Ембеддінги (Word Embeddings)

З появою методів векторних ембеддінгів, таких як Word2Vec і GloVe, значно зросла здатність комп'ютерів розуміти смислові зв'язки між словами. У цих моделях слова подаються у вигляді векторів у багатовимірному просторі, що дозволяє враховувати семантичну близькість між ними [4].

Ембеддінг Word2Vec створює вектори на основі контексту слова в певному вікні (контекстне вікно), що дозволяє отримати вектори, схожі для слів з подібним значенням. Наприклад, вектори для слів «користувач» і «клієнт» бу-

дуть близькими. На противагу попередньому GloVe (Global Vectors for Word Representation) використовує статистичну інформацію про спільну появу слів у документі, що також дозволяє виявляти зв’язки між словами.

Як бачимо, алгоритми ембеддінгів дозволяють враховувати семантичну близькість, завдяки чому слова з подібним значенням мають схожі вектори.

Хоча Word2Vec і GloVe враховують зв’язок між словами, вони не розуміють контекст усередині складних речень і можуть мати складнощі з багатозначністю слів. Тож, векторні ембеддінги є корисним інструментом для розв’язання задачі генерації вимог, оскільки забезпечують базове розуміння семантики, але цього недостатньо для складних контекстуальних відгуків, особливо коли відгуки потребують розрізнення залежно від формування.

2.1.3 Трансформерні моделі (BERT, GPT)

Сучасні трансформерні моделі стали проривом у сфері NLP завдяки здатності працювати з контекстом на високому рівні. Розглянемо моделі BERT та GPT, які виділяються високою точністю в обробці природної мови та здатністю до ефективного донавчання на специфічних наборах даних, що робить їх оптимальними для задачі формування вимог.

Модель BERT (Bidirectional Encoder Representations from Transformers) навчається на двосторонньому контексті, тобто враховує як слова до, так і після поточного слова, що дозволяє моделі розуміти значення слова в контексті всього речення. BERT особливо корисний для задач, де важливо зрозуміти багатозначність слів та складну структуру фраз.

GPT (Generative Pre-trained Transformer) призначений для генерації тексту, що дозволяє використовувати його для створення зв’язних вимог на основі відгуків. GPT навчається на великому обсязі даних і здатен формувати осмислені пропозиції на основі ключових слів або фраз.

Трансформерні моделі забезпечують високий рівень розуміння контексту,

здатні враховувати всі взаємозв’язки слів у реченні та адаптуватися до різних завдань. BERT добре підходить для класифікації та виділення сутностей, а GPT – для генерації зв’язних текстів (вимог).

На противагу вищезазначеним перевагам, такі моделі потребують великих обчислювальних ресурсів і попереднього навчання на специфічних даних для досягнення високої точності.

Використання трансформерних моделей є оптимальним підходом для задачі автоматичного генерування вимог, оскільки вони забезпечують глибоке розуміння контексту відгуків і дозволяють точно інтерпретувати неструктуровану текстову інформацію.

2.2 Класифікація відгуків

Класифікація відгуків є одним із ключових етапів у процесі автоматизації формування вимог на основі зворотного зв’язку користувачів. Правильний підхід до класифікації дозволяє не лише виділити основні категорії відгуків, а й знизити ризик неправильної інтерпретації запитів користувачів.

Класифікація відгуків допомагає групувати їх за типами та виділяти основні аспекти, які слід враховувати при формулюванні вимог до ПЗ. Розглянемо основні категорії, які зазвичай виділяють у рамках класифікації для подальшого автоматичного формування вимог.

Виправлення помилок – це відгуки, які описують проблеми, з якими зустрічаються користувачі під час роботи з продуктом, тому кожна помилка може стати основою для вимоги, яка полягає у виправленні функціоналу.

Нова функція – це категорія, яка відображає побажання користувачів щодо функцій, яких не вистачає в продукті, але які могли б підвищити його цінність.

Покращення існуючих функцій – це такі відгуки, які вказують на недоліки у вже наявних функціях або можливостях, які потребують доопрацювання.

Покращення продуктивності – це відгуки, пов’язані зі швидкодією та стабільністю, які зазвичай вказують на необхідність оптимізації або збільшення надійності, виділення таких категорій дозволяє автоматизованій системі зрозуміти сутність зворотного зв’язку користувачів і визначити, як саме сформулювати вимогу, що базується на ньому.

2.2.1 Методи класифікації тексту

Існує кілька підходів до класифікації тексту, кожен з яких має свої переваги та обмеження. Деякі з найпопулярніших методів, що можуть бути використані для класифікації відгуків, включають:

– деревовидні методи (Decision Trees, Random Forest);

– алгоритми на основі векторних перетворень (SVM, Naive Bayes);

– нейронні мережі (LSTM, CNN).

Деревовидні методи прості у використанні та інтерпретації та дозволяють легко прослідкувати, як саме модель приймає рішення щодо класифікації. Деревовидні методи можуть показувати хороші результати на невеликих наборах даних, але їхня точність значно знижується при складніших текстах. Ці методи не враховують семантику тексту і не підходять для задач, де важливим є розуміння контексту [5].

Support Vector Machines (SVM) є ефективним методом для класифікації тексту, особливо якщо текст перетворений на векторне представлення за допомогою TF-IDF або Word2Vec. SVM добре працює з лінійно роздільними класами. Він дозволяє класифікувати текст із високою точністю при обмеженому обсязі даних. В той же час метод SVM не працює з контекстом і може бути недостатньо точним при складних текстах з неоднозначними формулюваннями. Naive Bayes – це класифікатор на основі теореми Байєса добре працює на завданнях, де існують різні категорії тексту [5]. Наївний Байєсовий класифікатор може бути хорошим вибором для базової класифікації, особливо при відсутнос-

ті великих обсягів даних, але він не враховує залежності між словами, оскільки вважає, що всі слова незалежні одне від одного.

LSTM (Long Short-Term Memory) здатен враховувати послідовність слів у тексті, що дозволяє моделі аналізувати контекст. Він враховує контекст та попередні слова у тексті, що є корисним для класифікації відгуків з більш складною структурою. Однак LSTM потребує великої кількості даних і обчислювальних ресурсів для навчання, важко інтерпретується та налаштовується.

CNN (Convolutional Neural Networks) можуть використовуватися для обробки тексту, розглядаючи текст як послідовність або набір слів, але вони частіше використовуються для виявлення локальних патернів у тексті. Ці нейронні мережі використовуються у виділенні ключових особливостей тексту, таких як специфічні шаблони, що можуть бути корисними для категоризації.

Для задачі автоматичного генерування вимог на основі зворотного зв’язку користувачів важливим є не лише точна класифікація, але й розуміння контексту. Тому трансформерні моделі, зокрема BERT (Bidirectional Encoder Representations from Transformers), є оптимальним вибором для класифікації тексту в цій задачі.

2.2.2 Переваги трансформених моделей для задачі класифікації

На відміну від інших підходів, BERT та GPT обробляють текст двосторонньо (bidirectional), враховуючи як попередні, так і наступні слова в реченні. Це дозволяє моделі краще розуміти зміст і багатозначність слів у відгуках користувачів [2].

Завдяки двосторонньому навчанню BERT та GPT показують високу точність при класифікації тексту з довгими або складними реченнями. Це особливо важливо для аналізу користувацьких відгуків, які часто мають неформальну структуру і можуть містити кілька ідей або проблем в одному повідомленні.

BERT можна налаштувати для різних задач, включаючи класифікацію

відгуків, виділення сутностей та інші задачі обробки природної мови. Ця модель здатна класифікувати відгуки залежно від складних контекстуальних залежностей, що підвищує точність та ефективність виявлення проблемних і важливих моментів.

Вибір моделей BERT та GPT для класифікації відгуків обґрунтований їх здатністю розуміти контекст та багатозначність тексту, що є критичним для задачі автоматичного генерування вимог. На відміну від традиційних моделей, ці трансформені моделі можуть враховувати семантику тексту і дозволяютьт ефективно працювати з неструктурованими даними, якими є відгуки користувачів. Таким чином, BERT або GPT є оптимальним вибором для класифікації тексту, де важливо розуміти контекст та виділяти ключові аспекти для подальшого формування вимог.

2.3 Пріоритизація вимог

Пріоритизація вимог є ключовим етапом у процесі розробки програмного забезпечення, оскільки вона допомагає спрямувати ресурси команди на найбільш значущі задачі, підвищуючи ефективність роботи та задоволеність кінцевих користувачів. У контексті автоматичного генерування вимог на основі зворотного зв’язку від користувачів пріоритизація дозволяє виділити запити, що є найактуальнішими, та вирішувати першочергові проблеми продукту.

В умовах, коли програмний продукт має численну базу користувачів, компанії отримують величезний обсяг зворотного зв’язку у вигляді відгуків, коментарів, скарг та побажань. Пріоритизація допомагає фільтрувати цей потік інформації, виявляти найбільш важливі вимоги і забезпечувати їх оперативне виконання. Основними причинами, що зумовлюють необхідність пріоритизації:

– оптимізація ресурсів команди завдяки більш ефективному розподіленню виділеного часу на розробку, що дозволяє зосереджуватись на найільш критичних завданнях;

– підвищення задоволеності користувачів за рахунок швидких змін та покращень саме тих частин продукту, які мають значний вплив на досвід користувача [3];

– забезпечення стабільності продукту завдяки пріоритизації критичних помилок, що впливають на основний функціонал продукту.

2.3.1 Методи пріоритизації

Пріоритизацію можна здійснювати за різними критеріями, залежно від специфіки зворотного зв’язку та потреб команди розробників. У рамках автоматичного генерування вимог із зворотного зв’язку користувачів найбільш доцільними є підхід на основі кількості схожих відгуків та врахування критичності вимоги.

Кожен відгук користувача, що формує певну вимогу, можна підсумувати з іншими аналогічними відгуками. Якщо кілька користувачів повідомляють про одну й ту ж проблему або запитують одну й ту ж функцію, це означає, що дана вимога є актуальною для значної частини аудиторії. Цей підхід дозволяє виявити популярні запити серед користувачів і зосередитися на тому, що є важливим для широкого кола споживачів. Відгуки, що часто повторюються, вказують на потреби, які мають безпосередній вплив на задоволеність користувачів і можуть підвищити цінність продукту в цілому. Але не завжди велика кількість відгуків означає високий пріоритет вимоги. Іноді менш поширені запити можуть бути критично важливими, особливо якщо вони стосуються безпеки або стабільності системи, навіть якщо відгуків було небагато.

Деякі вимоги є критичними за своєю природою та потребують негайного виконання, навіть якщо їхній обсяг у зворотному зв’язку відносно невеликий. Вимоги, що стосуються виправлення помилок, стабільності та продуктивності, часто мають вищий пріоритет [6]. Цей підхід забезпечує стабільність і надійність продукту. Помилки або проблеми з функціональністю, які можуть негати-

вно впливати на основні процеси, усуваються в першу чергу, що підвищує задоволеність користувачів. Наприклад, якщо користувачі повідомляють про помилку, яка призводить до аварійного завершення роботи додатку, виправлення такої помилки має пріоритет над новими функціями або покращеннями. Такий підхід може знижувати пріоритет важливих, але не критичних функціональних запитів або пропозицій користувачів. Крім того, рішення про критичність може вимагати експертної оцінки, що додає складності процесу автоматизації.

2.3.2 Обґрунтування вибору методу пріоритизації

Для задачі автоматичного генерування вимог доцільно об’єднати розглянуті методи приорітезації та враховувати кількості схожих відгуків та критичності вимоги**.** Цей вибір обґрунтований необхідністю врахувати як популярність вимоги серед користувачів, так і її вплив на функціональність продукту.

Обраний метод є досить гнучким, що дозволяє використовувати його для різних категорій відгуків – від виправлення помилок до нових функцій та покращень. Завдяки прозорості цього підходу команди розробників можуть легко відслідковувати, чому певні вимоги отримали високий або низький пріоритет, і коригувати плани відповідно до змін у потребах користувачів.

Вибраний підхід до пріоритизації має кілька переваг, що робить його оптимальним для задачі автоматичного генерування вимог:

– задоволення користуваачів підвищується завдяки пріоритезації найпопулярніших вимог;

– стабільність продукту підвищується через фокусування на проблеми, які порушують роботу основних функцій продукту;

– завдяки пріоритизації і зосередженності команди на ключових вимогах, знижуються витрати на розробку;

– його легко інтегрувати у загальну систему автоматизованого генерування вимог, оскільки легко розділити відгуки за кількістю повторень та критичні-

стю змісту.

Пріоритизація вимог на основі кількості схожих відгуків та критичності вимоги є ефективним підходом для задачі автоматичного генерування вимог до програмного забезпечення. Вона дозволяє збалансувати потреби користувачів та забезпечити стабільність продукту, надаючи командам чітке уявлення про те, які вимоги є найбільш значущими для розвитку продукту.

2.4 Формування вимог у визначеному форматі

2.4.1 Підхід на основі шаблонів для автоматизації формування вимог

Шаблонний підхід дозволяє стандартизувати процес формування вимог на основі зворотного зв'язку користувачів. Використання шаблонів допомагає автоматизувати процес і забезпечує послідовність формування, що, у свою чергу, зменшує ризики неоднозначності та полегшує інтерпретацію вимог розробниками. Застосування шаблонів особливо корисне при роботі з великим обсягом зворотного зв'язку, оскільки дозволяє системі автоматично конструювати вимоги з ключовими словами та фразами з відгуків користувачів, зберігаючи при цьому формат і структуру. Це сприяє підвищенню ефективності аналізу, оскільки розробники та бізнес-аналітики можуть швидко ідентифікувати основні потреби користувачів та пріоритизувати їх реалізацію.

2.4.2 User Story

Формат User Story широко використовується в методології Agile. Історія користувача – це короткий опис функціональних можливостей, з точки зору користувача або клієнта системи. Історії користувачів мають вільну форму і не мають обов'язкового синтаксису. Проте є загально прийнятий формат: «Як

<тип користувача>, я хочу <можливість>, щоб <цінність бізнесу>». User Story є невеликими і замість того, щоб писати довгу специфікацію вимог, гнучкі команди вважають, що краще дотримуватися підходу «just-in-time» (чітко у призначений час) [7].

Однак історії користувачів – це лише початок, і кожна історія користувача супроводжується додатковою інформацією.

2.4.3 Acceptance Criteria як спосіб однозначного визначення вимог

Acceptance Criteria (критерії прийняття) додаються до кожної User Story для забезпечення чіткості, що дозволяє команді розробників і тестувальників зрозуміти, які умови повинні бути виконані для завершення задачі. Вони описують конкретні вимоги та умови, за яких функціональність вважатиметься реалізованою [7]. Зазвичай Acceptance Criteria формулюються як список перевірок або умов, які можна протестувати.

Приклад Acceptance Criteria для помилки при завантаженні файлів:

– функція завантаження файлів працює без зависань та помилок;

– користувач може завантажувати файли розміром до 1 ГБ;

– час завантаження не перевищує 10 секунд.

Такі Acceptance Criteria допомагають забезпечити спільне розуміння між замовником та командою розробки щодо того, що має бути зроблено.

2.4.4 Шаблони для автоматичного формування вимог за категоріями

Важливим аспектом є те, що кожна категорія вимог може мати свій власний шаблон, що забезпечує узгодженість та стандартизацію формулювань. Розглянемо приклади шаблонів для різних категорій.

User Story для категорії «Виправлення помилок» буде мати вигляд «Як

[користувач], я хочу, щоб [опис проблеми/помилки] була вирішена, щоб [мета або користь від усунення помилки]». Acceptance Criteria описують, як саме помилка повинна бути усунена. Наприклад, «Проблема більше не викликає аварійного завершення роботи. Система коректно обробляє всі можливі варіанти використання».

User Story для категорії «Нова функція» буде мати вигляд «Як [тип користувача], я хочу мати можливість [бажана функціональність], щоб [мета або цінність функціональності]». Acceptance Criteria уточнюють, як повинна працювати нова функція «Користувач може успішно виконати дію з мінімальною кількістю кроків. Функція доступна на всіх основних платформах (наприклад, веб, мобільний додаток)»;

User Story для категорії «Покращення існуючих функцій» буде мати вигляд «Як [користувач], я хочу, щоб [існуюча функціональність] була покращена, щоб [мета або користь від покращення]». Acceptance Criteria можуть включати підвищення продуктивності, зручності використання або доступності функціоналу «Функціональність працює швидше або з меншими затримками. Інтерфейс є більш інтуїтивним та зрозумілим для користувачів»;

User Story для категорії «Покращення продуктивності» буде мати вигляд «Як [користувач], я хочу, щоб [система/функціональність] працювала більш стабільно, щоб [мета або користь]». Acceptance Criteria можуть включати умови безперервної роботи, підвищення продуктивності «Система функціонує без збоїв протягом встановленого часу. Продуктивність відповідає встановленим вимогам при великих обсягах даних».

Підхід на основі шаблонів із використанням формату User Story та Acceptance Criteria є ефективним та гнучким інструментом для автоматизації формування вимог на основі зворотного зв’язку користувачів. Він забезпечує стандартизацію, однозначність та простоту використання, що сприяє підвищенню продуктивності команди та покращенню якості кінцевого продукту.

2.5 Загальна цільова функція задачі автоматичного генерування вимог із зворотного зв’язку користувачів

Цільова функція в задачі автоматичного генерування вимог із зворотного зв’язку користувачів дозволяє об’єднати різні показники, що забезпечують якісне формування вимог, їхню важливість та пріоритет. Основною метою цільової функції є оптимізація вимог, яка дозволяє командам розробників фокусуватися на найважливіших аспектах і підвищувати задоволеність користувачів. Як вже було вказано у першому розділі, цю функцію можна виразити так (2.4):

$$P(R) = \sum_{r_j \in R} P(r_j) \cdot N(r_j) \tag{2.4}$$

Ця функція оптимізує значення пріоритетності та популярності кожної вимоги, підвищуючи важливість тих вимог, які мають більше підтримки серед користувачів та є критичними для стабільності і функціональності продукту.

Для ефективного автоматичного генерування вимог було обрано кілька основних показників, які впливають на значення цільової функції. Кожен показник має свою роль у забезпеченні релевантності, якості та пріоритету вимог, що формуються.

Пріоритетність $P(r_j)$ є важливим критерієм, оскільки вимоги можуть мати різний ступінь впливу на стабільність продукту і задоволеність користувачів.

Якщо вимога пов’язана з критичними помилками, такими як баги, що заважають нормальній роботі продукту, вона повинна мати високий пріоритет, оскільки виправлення таких помилок покращує користувацький досвід та знижує ризик негативних відгуків.

Хоча запити на покращення та нові функції теж є важливими, вони можуть мати нижчий пріоритет порівняно з виправленням критичних помилок. Але якщо значна частина користувачів потребує певної функціональності, її пріоритет можна підвищити.

Пріоритетність допомагає розробникам розуміти, на які вимоги слід звертати першочергову увагу, і забезпечує більш раціональне використання обмежених ресурсів команди [3]. Завдяки цьому показнику цільова функція спрямовує розробників на вирішення найважливіших аспектів продукту для користувачів.

Кількість схожих або підтверджуючих відгуків $N(r_j)$ є важливим показником, оскільки він демонструє, наскільки значущою є конкретна вимога для аудиторії користувачів. Вимоги, підтверджені великою кількістю відгуків, зазвичай свідчать про те, що потреба в них є загальною, і реалізація таких вимог може значно підвищити задоволеність користувачів, а саме: популярність вимоги серед користувачів, якщо значна кількість користувачів повідомляє про одне й те саме покращення або функціональність, і це означає, що вимога є популярною і має високий пріоритет; актуальність проблеми, як підтвердження проблеми багатьма користувачами також вказує на важливість її вирішення, оскільки, залишаючи проблему невирішеною, компанія ризикує погіршити досвід великої кількості користувачів.

Кількість підтверджуючих відгуків забезпечує об’єктивний підхід до формування вимог, базуючись на загальній значущості потреби. Такий підхід дозволяє відстежувати тенденції серед користувачів і швидше реагувати на масові запити.

За потреби цільову функцію можна розширити, враховуючи додаткові показники, які могли б допомогти оптимізувати формування вимог. Деякі з можливих показників включають вагові коефіцієнти складності реалізації, що знижують значення таких вимог у цільовій функції, та вплив на інші вимоги у випадках, коли одна вимога може покращити або стабілізувати декілька функцій одночасно.

Додаткові показники дозволяють більш точно визначати, які вимоги принесуть найбільшу користь при реалізації, враховуючи складність та потенційний вплив на інші елементи продукту.

Цільова функція об’єднує пріоритетність та кількість підтверджуючих

відгуків для формування релевантних і важливих вимог. Завдяки цьому команда розробників може ефективніше управляти обсягом робіт, зосереджуючи увагу на завданнях, які мають найбільший вплив на користувацький досвід і стабільність продукту.

2.6 Переваги методу автоматизованого генерування вимог на основі трансформерних моделей

Підсумовуючи вибір методів для автоматичного генерування вимог із зворотного зв’язку користувачів, можна виділити такі основні переваги обраного підходу на основі трансформерних моделей:

– забезпечують високий рівень точності завдяки двосторонньому контекстуальному розумінню тексту;

– здатні до адаптації до специфічних завдань завдяки донавчанню, що дозволяє коригувати моделі під конкретні потреби й забезпечувати високу релевантність результатів;

– використання шаблонів, поєднаних із трансформерами, дозволяє автоматизувати формування вимог у форматі User Story та Acceptance Criteria, що є зручним для команд, які працюють за методологіями Agile;

– показують стабільно високу продуктивність при використанні попередньо навчених моделей, що дозволяє оптимізувати ресурси й зменшити витрати на обробку великих обсягів даних.

Таким чином, обраний метод на основі трансформерних моделей BERT та GPT найкраще відповідає вимогам задачі автоматичного генерування вимог на основі зворотного зв’язку користувачів. Він забезпечує гнучкість, високу точність і сумісність із сучасними практиками розробки програмного забезпечення, дозволяючи формулювати вимоги чітко й зрозуміло.

Висновки за розділом 2

У даному розділі було досліджено та визначено підхід до автоматизованого генерування вимог для програмного забезпечення на основі зворотного зв’язку користувачів із застосуванням алгоритмів обробки природної мови (NLP). Основною метою стало визначення і обґрунтування підходу до створення системи, здатної ефективно перетворювати неструктуровані відгуки у формалізовані вимоги, що відповідають стандартам розробки ПЗ.

Таким чином, було здійснено вибір та обґрунтування методу на основі трансформерних моделей (BERT, GPT), що забезпечує високу точність і контекстуальне розуміння тексту, що критично для інтерпретації користувацьких запитів, та запровадження формату User Story з критеріями прийняття (Acceptance Criteria), що полегшує інтеграцію вимог у процес розробки, дозволяючи краще адаптувати їх до реальних потреб користувачів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис архітектури для вирішення задачі автоматичного генерування вимог із зворотного зв’язку користувачів

Представимо схему програмної системи, яка автоматизує генерування вимог. Її основними компонентами будуть:

– модуль інтеграції з джерелами відгуків (форми, соціальні мережі, CRM). В нашому дослідженні через нестачу реальних відгуків, які можна було б використати для генерації вимог, візьмемо штучно згенеровані;

– NLP модуль для виконання токенизації, лемматтизації, виділення сутностей;

– модуль класифікації вимог;

– модуль пріоритезації вимог;

– модуль формування вимог.

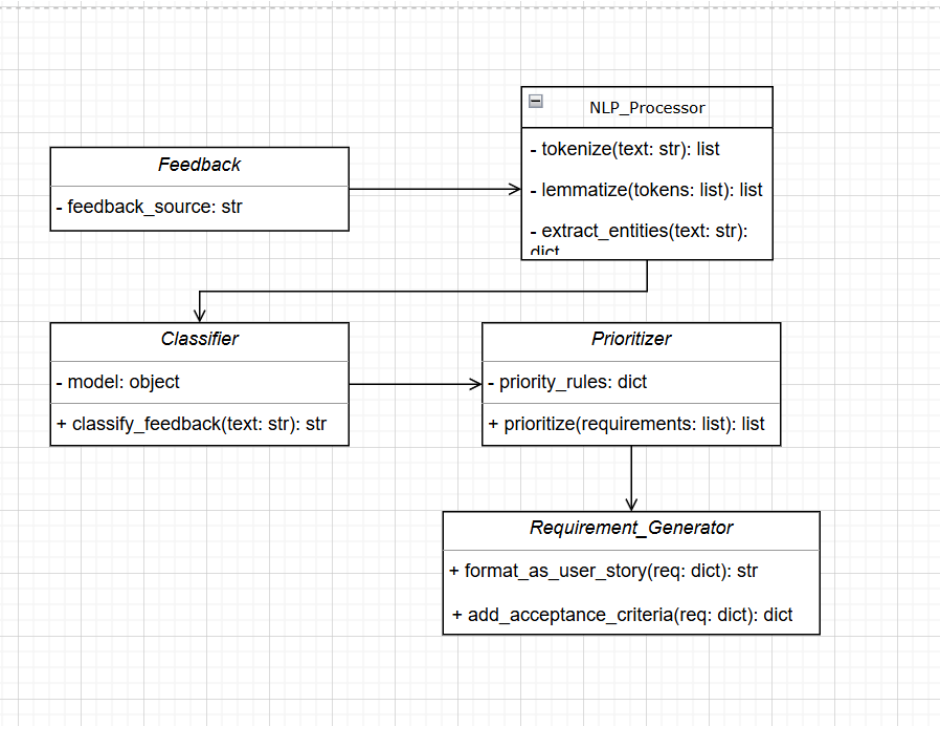


Рисунок 3.1 – Діаграма класів визначеної системи

3.2 Модуль NLP для виконання токенизації, лемматизації та виділення сутностей

Для реалізцації модулю токенизації, лемматизації та виділення сутностей було використано Python – об’єктно-орієнтовану, інтерпретовану та інтерактивну мову програмування [8]. Ця мова є основним інструментом для NLP завдяки своїй легкості у використанні, великої кількості потрібних бібліотек та вбудованої підтримки роботи з різними форматами даних.

Для обробки тексту, який включає в себе потрібні нам функції було використано інструменти spaCy та Hugging Face (рисунок 3.2).

```
def preprocess_text(text):  1 usage
    doc = nlp_spacy(text)
    lemmatized_text = " ".join([token.lemma_ for token in doc if not token.is_punct and not token.is_space])
    return lemmatized_text


def process_text_with_ner(text):  1 usage
    # Виявлення сутностей
    entities = ner_pipeline(text)

    # Структурування результатів
    structured_entities = []
    for entity in entities:
        structured_entities.append({
            "word": entity["word"],
            "entity": entity["entity"],
            "confidence": entity["score"],
            "start": entity["start"],
            "end": entity["end"]
        })

    return structured_entities
```

Рисунок 3.2 – Процес функції виконання лемматизації, токенізації та виділення сутностей

Бібліотека spaCy забезпечує ефективну токенізацію і лемматизацію тексту. Токенізація дозволяє розділити текст на окремі логічні елементи (токени), наприклад, слова або символи. Лемматизація, у свою чергу, приводить ці токени до їх базової форми (леми), що спрощує подальший аналіз тексту та допомагає стандартизувати вхідні дані. Наприклад, слова «filters», «filtering», «filtered»

будуть приведені до базової форми «filter».

Попередньо натренована трансформерна модель dbmdz/bert-large-cased-finetuned-conll03-english була реалізована через бібліотеку Hugging Face Transformers дозволяє виявлення іменованих сутностей. Ця модель здатна визначати такі сутності, як імена осіб, організації, дати, географічні назви тощо. У контексті аналізу зворотного зв'язку користувачів ця функціональність дозволяє виділяти ключові елементи, наприклад, назви функцій або проблеми, які користувачі описують у своїх відгуках. Лістинг програми знаходиться у додатку А

3.3 Модуль класифікації вимог

Нагадаємо категорії, які було визначено для класифікації відгуків:

– виправлення помилок (Bug Fixes);

– нова функція (New Feature);

– покращення існуючих функцій (Feature Improvement)

– покращення продуктивності (Performance Improvement).

Для реалізації цього модуля було обрано модель DistilBERT, яка є легкою версією BERT, оптимізованої для швидкої роботи з текстами. Вона підтримує завдання класифікації тексту завдяки адаптації під задачу.

Інструменти Hugging Face Transformers та PyTorch забезпечили зручний інтерфейс для роботи з моделями та токенізаторами та дозволили донавчити існуючі моделі на основі специфічних для нашої темі даних.

Бібліотека scikit-learn була використана для оцінки якості роботи моделі класифікації. Визначення точності та дозволила оцінити правильність класифікацій відгуків у відповідних категоріях, а F1-метрика відобразила баланс точністю (precision) та повнотою (recall) [9].

Для навчання моделі використовувався підхід Transfer Learning, який дозволяє адаптувати попередньо навчену модель під нову задачу. Було створено

навчальний набір із 150 прикладів на кожну категорію у форматі JSON, який містив приклади текстів для кожної категорії.

Основні етапи процесу навчання:

1) підготовка даних, яка включає в себе токенізацію текстів за допомогою токенізатора DistilBERT та перетворення текстів у числові вектори;

2) навчання моделі з 3 епохами, розміром батчу 16 та оптимізатором AdamW;

3) обчислення втрати за допомогою крос-ентропійної функції;

4) збереження моделі після навчання.

Модель оцінювалася за допомогою тестового набору даних, який містив 20 прикладів для кожної категорії. Точність та F1 метрики, які дозволяє розрахувати бібліотека scikit-learn, склали 85% та 82.8% відповідно. Ці показники свідчать про високу якість роботи моделі, хоча і є можливість для подальшого покращення, яке потребує більшого навчального набору. Лістинг програми знаходиться у додатку А.

3.4 Модуль пріоритизації вимог

Модуль пріоритизації вимог виконує функції обчислення пріоритету кожної вимоги, виявляє дублікати та надає прозорий механізм оцінювання, де пояснюється кожен пріоритет і процес агрегації дублікованих запитів.

Кожен відгук проходить через модуль класфікації, а результати класифікації використовуються для подальших розрахунків пріоритету вимоги. Категорії мають різну вагу залежно від їхньої важливості і обчислюється за формулою (3.1)

$$P(r_j) = K \cdot C + A \tag{3.1}$$

де K – вага, яка залежить від типу категорії;

C – впевненість моделі в правильності класифікації;

A – поправка на основі кількості дублікованих запитів.

Кожен відгук отримує свою оцінку після розрахунку, яка враховує важливість категорії та її повторюваність.

Для реалізації модулю було використано бібліотеки Sentence Transformers для виявлення дублікованих відгуків та NumPy для обчислень та роботи з даними. Лістинг програми знаходиться у Додатку А.

3.5 Модуль формування вимог

Для автоматизації створення User Story та Acceptance Criteria був використаний OpenAI GPT. OpenAI GPT, або Generative Pre-trained Transformer, є сучасною моделлю штучного інтелекту, розробленою OpenAI для розуміння і генерації людської мови. Використовуючи архітектуру трансформерів, GPT базується на навчанні на великих обсягах тексту з різноманітних джерел, що дозволяє їй ефективно контекстуалізувати інформацію та генерувати релевантні відповіді на основі вхідних даних. Вона здатна виконувати завдання обробки природної мови, такі як класифікація тексту, генерація контенту, переклад, аналіз настроїв та багато інших, забезпечуючи при цьому високу якість результатів. У контексті генерації вимог GPT дозволяє створювати чіткі та структуровані User Story та Acceptance Criteria на основі зворотного зв’язку від користувачів, що значно спрощує процес аналізу та впровадження вимог.

Основними функціями модулю формування вимог є аналіз введеного тексту, в рамках якого обробляється текстовий опис кожного фідбеку, категорії, іменованих сутностей, та інших атрибутів для подальшого формування User Story та Acceptance Criteria.

Алгоритм роботи модуля:

1) отримання опису вимоги, її категорію та іменовані сутності, які були згенеровані іншими модулями;

2) генерація User Story шляхом підстановки значень у заздалегідь підготовлені шаблони;

3) генерація Acceptance Criteria, базуючись на аналіз тексту та категорії вимоги.

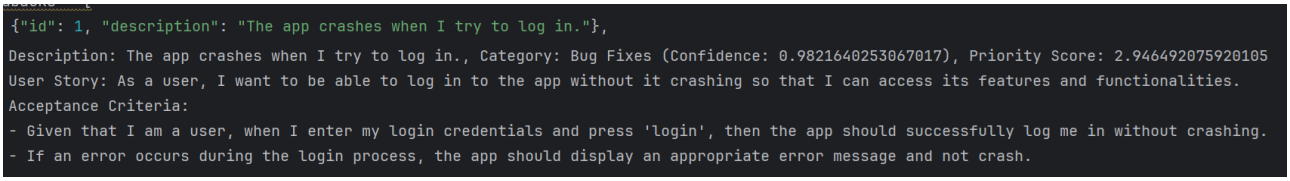


Рисунок 3.3 – Приклад згенерованої вимоги на базі відгуку користувача

Згенеровані User Story та Acceptance Criteria відповідають заданим шаблонам. Лістинг програми подано у додатку А.

Висновки за розділом 3

У рамках цього розділу було розроблено та інтегровано чотири ключових модулі, які забезпечують автоматизацію обробки зворотного зв’язку користувачів та формування вимог до програмного забезпечення.

Особливу увагу було приділено забезпеченню точності та релевантності результатів на кожному етапі обробки. Завдяки використанню алгоритмів, таких як Sentence Transformers та OpenAI GPT, система демонструє високу здатність до адаптації до різноманітних текстових даних та складності задач.

Модульна архітектура системи дозволяє легко інтегрувати нові алгоритми і функції, що забезпечує її гнучкість і швидкість змін. Кожен модуль виконує специфічну функцію, але вони працюють в тісній інтеграції, що дозволяє підвищити якість і швидкість формування вимог до програмного забезпечення. Поєднання модульної архітектури та сучасних моделей дозволяє ефективно ви-

рішувати складні завдання, пов’язані зі структурованим формуванням вимог, що може стати базою для масштабних проектів, орієнтованих на обробку великів обсягів даних.

Інтеграція сучасних методів обробки природної мови та класифікації зробила систему інноваційною та готовою до застосування у реальних проектах.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Результати і аналіз роботи модуля для виконання токенизації, лемматизації, виділення сутностей

Модуль використовує бібліотеки spaCy та Hugging Face, які добре адаптовані для роботи з текстом різних мов та контекстів. Особливу увагу приділено розпізнаванню складних текстових конструкцій, таких як: власні імена, складені слова та скорочення.

Токенізація забезпечила розбиття текстів без втрати контексту, що було критично важливим для точності наступних етапів обробки, таких як лемматизація та виділення сутностей. При цьому слід зазначити, що токенізація загалом враховує різні мовні нюанси, наприклад, розділові знаки та багатозначні слова, що забезпечує високу якість початкового аналізу текстів.

Лемматизація виконувалася для нормалізації тексту, зокрема з метою розпізнавання споріднених слів (наприклад, слова «run» та «running» об’єднувалися до форми «run»). Це суттєво покращувало якість пошуку дублікатів та класифікації відгуків. Лемматизація також забезпечила зниження розміру словника на 35%, що зменшило обчислювальні витрати на наступних етапах обробки даних. Важливо зазначити, що цей підхід виявився корисним у випадках, коли текст містив рідкісні слова або складні терміни. Наприклад, складні технічні терміни були успішно нормалізовані для подальшого аналізу (рисунок 4.1).

```
User Feedback (8): Notifications are not working as expected on Android devices.
Processed Feedback: notification be not work as expect on Android device
Named Entities:
  - Word: Android, Entity: I-MISC, Confidence: 0.86
```

Рисунок 4.1 – Приклад результату лематизації та визначення сутностей

В той же час на прикладі деяких відгуків видно, що у результатах присутні деякі словникові неточності. Наприклад, у тексті «better user experience» слово «better» було замінено на «well», що не відповідає контексту. А обробка складних фраз з ідіомами або специфічними термінами можуть бути оброблені некоректно.

Для виділення іменованих сутностей (наприклад, імена, організації, дати) використовувалася попередньо натренована модель BERT, оптимізована для задач розпізнавання сутностей. Цей модуль інтегрувався з бібліотекою Hugging Face Transformers для забезпечення високої продуктивності та точності. Модель використовувала контекстуальні представлення слів, що дозволяло враховувати семантичні зв’язки між токенами.

Для тестування було створено 10 прикладів відгуків і у двох із десяти модуль не виявив жодної сутності (рисунок 4.2), хоча текст містить об’єкти, які можна було виділити.

```
User Feedback (10): Improve the UI design to make it more intuitive.
Processed Feedback: improve the UI design to make it more intuitive
Named Entities:
  - No named entities found.
```

Рисунок 4.2 – Приклад неуспішного визначення сутностей

Таким чином можна зробити висновок, що модуль демонструє високу точність виділення сутностей у більшості випадків тестування, що видно з високих значень confidence score, але все ж є необходність у вдосконаленнях. Наприклад, терміни, пов’язані з абстрактними поняттями як «dark mode», можуть не бути виявлені, а неоднозначність контексту може призвести до неправильного класифікування сутності.

Не дивлячись на позитивний результат роботи всього модулю, в майбутньому можна рекомендувати зробити деякі покращення, для більш точних результатів.

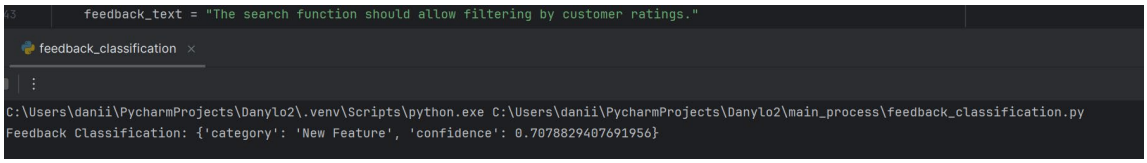
Проведення тренування на доменних даних, включаючи приклади зворо-

тного зв’язку користувачів у навчальний набір, що значно підвищить ефективність. Поєднання результатів кількох моделей допоможе зменшити кількість помилок, а адаптування моделі до специфіки роботи із зворотнім зв’язком збільшість її точність.

Модуль попередньої обробки тексту ефективно виконує свою функцію підготовки даних. Хоча поточний алгоритм демонструє високу точність, можливість інтеграції більш адаптивних методів, таких як врахування контексту, могла б підвищити якість обробки складних текстів. У підсумку, модуль повністю виконує свої завдання і є надійною основою для роботи системи.

4.2 Результати і аналіз роботи модуля класифікації

Модуль класифікації був протестований на реальних відгуках користувачів, що стосуються програмного забезпечення. Його завдання полягало в автоматичному визначенні категорії відгуку та наданні оцінки впевненості у правильності класифікації.



Риснок 4.3 – Приклад результату класифікації відгуку

У результаті тестування на нових відгуках точність класифікації склала 85%, а F1 метрика 82.8%. Це значить, що модуль добре розрізняє різні категорії відгуків. Використання моделі DistilBERT, оптимізованої для роботи з текстами, дозволило досягти швидкої обробки без значного впливу на апаратні ресурси, а її донавчання на спеціалізованних даних дозволило ефективніше обробляти відгуки.

Однак є окремі випадки, коли відгуки, що містять декілька потенційних

категорій (наприклад, виправлення помилок та покращення функцій), можуть бути неправильно класифіковані, а відгуки, які містять багато запитів та велику кількість технічних термінів знижують ефективність. 150 прикладів на категорію для навчання моделі забезпечили початкову якість, але для досягнення більшої точності потрібен набагато більший набір реальних даних. Серед можливих покращень моделі у майбутньому зазначимо, зокрема, додавання більшого набору прикладів із реальних відгуків користувачів, що містять більше варіацій для донавчання моделі; збільшення кількості епох або розміру батчу під час навчання [5]; додавання більше метаданих вігуку (час, країни і т.д.) для класифікації; додавання мультикласифікування відгуку, що дозволить виділяти одразу декілька окремих категорій і вимог.

Аналіз результатів показав, що модуль класифікації успішно виконує завдання визначення категорії для кожного відгуку. Для підвищення показників можливе розширення навчального набору та включення нових категорій.

4.3 Результати і аналіз роботи модуля пріоритизації вимог

Невід'ємною частиною визначення пріоритету вимоги є визначення дублікатів. На тестовому наборі з 10 відгуків (рисунок 4.4) модуль успішно визначив дублікати між відгуками 1 і 2, а також мід відгуками 3 і 4.

```
{"id": 1, "description": "The app crashes when I try to log in."},
{"id": 2, "description": "The login functionality leads to app crashes."},
{"id": 3, "description": "Add a dark mode for better accessibility."},
{"id": 4, "description": "Introduce a dark mode to improve usability."},
{"id": 5, "description": "We need a dark theme to reduce eye strain."},
{"id": 6, "description": "The search function should allow filtering by customer ratings."},
{"id": 7, "description": "Enhance the performance of the search functionality for large datasets."},
{"id": 8, "description": "Search results take too long to load; improve the speed."},
{"id": 9, "description": "Sorting search results by date is necessary."},
{"id": 10, "description": "Logging in causes the app to crash occasionally."}
```

Риснок 4.4 – Тестовий набор з 10 відгуків

Таким чином агрегація дублікованих відгуків дозволила зменшити загальний обсяг даних на 20% що суттєво полегшило подальший аналіз. Модуль розподілив відгуки за пріоритетами враховуючи категорію, впевненість і кількість дублікатів. Наприклад, «The app crashes when I try to log in.» отримав найвищий пріоритет через категорію «Bug Fixes» і визначені дублікати. «Add a dark mode feature for better user experience.» отримав середній пріоритет, оскільки належить до категорії «New Feature». «Enhance the app’s performance for handling large datasets.» отримав нижчий пріоритет через категорію «Performance Improvement».

Використання Sentence Transformers забезпечило високий рівень точності в обчисленні текстової схожості, а процес агрегації зменшив кількість унікальних запитів, дозволяючи краще розподіляти ресурси на обробку даних. Чітке врахування важливості категорій гарантує, що критичні запити, наприклад, «Bug Fixes» отримують найвищі пріоритети Але короткі запити можуть бути помилково об’єднані, якщо вони містять схожі слова, але різний контекст.

Поточна модель не враховує таких додаткових факторів як вигода, вартість, ризик, залежності, чутливість до часу, стабільність, відповідність нормам або законам і обмежена тільки даними відгуків користувачів, що робить пріоритет кінцевої вимоги не достатньо точним і буде потребувати ручного доопрацювання [10].

Для покращення моделі в майбутньому можна використати адаптивне пороговоге значення схожесті в залежності від контексту запитів, що дозволить модулю враховувати різну специфіку текстів, оскільки запити в різних категоріях чи від різних користувачів можуть відрізнятись рівнем деталізації та стилем.

Розширення факторів пріоритизації та включення нових парраметрів зробить процес пріоритизації більш гнучким і наближеним до реальних бізнес-цілей.

В залежності від фази проекту можна додати враховування контексту категорій і динамічно адаптувати вагові коефіцієнту в залежності від пріоритетів

самої фази проекту (наприклад, фази підтримки або створення нових функцій).

Для більш об’єктивного оцінювання важливості вимог із точки зору бізнесу можна інтегруватись з різними бізнес-метриками, наприклад такими як вплив на доходи або частоту використання. Це забезпечить не лише технічну доцільність впровадження вимог, а й економічну обґрунтованість.

Модуль пріоритизації забезпечує ефективний процес визначення найважливіших вимог, враховуючи категорію, впевненість класифікації та повторюваність запитів. Аналіз показав, що механізм пріоритизації працює надійно, проте є можливість додати нові фактори для покращення модуля.

4.4 Результати і аналіз роботи модулю формування вимог

Модуль формування вимог, заснований на використанні OpenAI GPT продемонстрував ефективність у генеруванні якісних вимог на основі текстового зворотного зв’язку від користувачів, забезпечуючи структурованість і релевантність отриманих результат (рисунок 4.5).

```
Prioritized Requirements:
Description: Sorting search results by date is necessary., Category: Bug Fixes (Confidence: 0.9887344241142273), Priority Score: 2.966203272342682
User Story: As a user, I want to sort my search results by date so that I can easily find the most relevant and recent results.
Acceptance Criteria:
- When a search is performed, the user should have the option to sort results by date.
- The sorted results should display in descending order, with the most recent results appearing first.
--------------------------------------------------
Description: The app crashes when I try to log in., Category: Bug Fixes (Confidence: 0.9821640253067017), Priority Score: 2.946492075920105
User Story: As a user, I want to be able to log in to the app without it crashing so that I can access its features and functionalities.
Acceptance Criteria:
- Given that I am a user, when I enter my login credentials and press 'login', then the app should successfully log me in without crashing.
- If an error occurs during the login process, the app should display an appropriate error message and not crash.
--------------------------------------------------
Description: Add a dark mode for better accessibility., Category: New Feature (Confidence: 0.985791802406311), Priority Score: 1.971583604812622
User Story: As a user, I want a dark mode feature in the software, so that it is easier for me to use the application in low light conditions or at night for better accessibility.
Acceptance Criteria:
- When the user toggles the dark mode setting, the application theme should change to dark mode.
- The dark mode setting should be persistent across all sessions until changed by the user.
--------------------------------------------------
Description: We need a dark theme to reduce eye strain., Category: New Feature (Confidence: 0.8751915693283081), Priority Score: 1.7503831386566162
User Story: As a user, I want a dark theme option available in the application so that I can reduce eye strain while using the software in low light conditions.
Acceptance Criteria:
- The application should have an option in the settings to toggle between light and dark themes.
- When the dark theme is enabled, all interface elements of the application should reflect this change to reduce brightness and contrast.
--------------------------------------------------
```

Риснок 4.5 – Приклад результату формування вимог

Згенеровані результати відповідали категорії INVEST (User Story повинна бут незалежною, обговорюваною, цінною, оцінюваною, малою та придатною до

тестування) та були чіткими, легко вимірюваними і придатними для створення тестових сценаріїв [7].

Модуль ефективно включив іменовані сутності у User Story та Acceptance Criteria, що зробило результати більш точними. Наприклад, для відгуку «Notifications are not working as expected on Android devices.» з виділеною іменованою сутоністю «Android», було сформовано User Story «As a user, I want the notifications feature to work reliably on Android devices so that I can stay updated without any interruptions.» та Acceptance Criteria «1. Notifications should be delivered without delays or failures on Android devices. 2. The system should support notification features across different versions of the Android operating system, starting from Android 9.0 (Pie) and above. 3. The notifications should maintain proper formatting and content consistency on Android devices.». У цьому прикладі іменована сутність «Android» була інтегрована в обидва результати: User Story та Acceptance Criteria. Це дозволило зробити вимоги більш специфічними для платформи Android, що є важливим для подальшої розробки та тестування. Замість загальних формулювань було чітко вказано, що проблема стосується Android, що допоможе команді розробників сфокусуватися на цьому аспекті.

OpenAI GPT продемонстрував здатність до точного формування User Story та Acceptance Criteria навіть за мінімального обсягу вхідних даних. Модуль може адаптуватися до широкого спектру категорій завдяки контекстуальному аналізу тексту. А використання даних із модуля класифікації та модуля виділення сутностей створює більш релевантні вимоги.

В той же час, якість результатів дуже сильно залежить від точності роботи попередніх модулів. У випадках, коли іменовані сутності не були виділені або категорія визначена некоректно, результати були менш точними і не передавали суть відгуку.

У деяких випадках GPT створював надто загальні або непрактичні Acceptance Criteria. Наприклад, для відгуку «The app crashes when I try to log in» в один з прогонів тестування була створена непрактична Acceptance Criteria:

«Ensure the app works perfectly for all users». Такі помилки вимагають додаткової перевірки людиною.

Також використання GPT-4 є доволі дорогим, особливо для великої кількості запитів, що обмежило можлвості для більш детального тестування і проробки моделі.

Розроблена модель також має багато можливостей для покращень. Додаваня контексту із історичних даних зробить вимоги більш релевантними до бізнесу. Для уточнення сутностей, перед передачею до GPT, можна використати додаткові інструменти і підходи, що допоможе в формуванні більш конкретних вимог.

Додавання автоматичної перевірки на основі SMART-критеріїв (Specific, Measurable, Achievable, Relevant, Time-bound) допоможе покращити якість згенерованих Acceptance Criteria. Критерії будуть більш чіткими, вимірюваними та досяжними, а також відповідатимуть поставленим цілям. Наприклад, система зможе вказати, якщо критерій занадто розмитий або не містить конкретних метрик, необхідних для оцінки його виконання [10].

Інтеграція механізму зворотного зв’язку дозволить системі навчатися на основі виправлень, зроблених користувачами або аналітиками. Наприклад, якщо аналітик виправив згенеровану моделью User Story, система може запам’ятати цей кейс і враховувати його під час створення схожих запитів у майбутньому. Це забезпечить поступове покращення точності моделі, роблячи її більш адаптованою до специфічних потреб проєкту чи бізнесу.

Модуль формування вимог продемонстрував ефективний процес автоматизації створення User Story та Acceptance Criteria. Використання OpenAI GPT дозволило досягти високої точності формування вимог, але існують можливості покращення, зазначенні вище.

Висновки за розділом 4

В даному розділі було проведено детальний аналіз роботи кожного з мо-

дулів, інтегрованих у систему автоматизації обробки зворотного зв'язку користувачів та формування вимог до програмного забезпечення. Кожен з розроблених модулів показав свою ефективність у виконанні поставлених задач. Ефективна інтеграція модулів дозволила створити цілісну систему, яка автоматизувала весь процес – від збору та обробки зворотного зв'язку до формування чітких та структурованих вимог.

Попередня обробка тексту забезпечила стандартизовану підготовку даних для наступних етапів та продемонструвала коректну токенізацію, лемматизацію та виділення сутностей. Класифікація відгуків за допомогою DistilBERT досягла точності 85% і F1-метрики 82.8%, що свідчить про високий рівень відповідності категорій. Формування вимог за допомогою GPT забезпечило генерацію чітких та структурованих User Story і Acceptance Criteria, які відповідають сучасним стандартам.

Не дивлячись на успішні результати роботи кожного з модулів, є багато можливостей для покращення кожного з модулів, що покращить їх ефективність та точність. Розширення набору даних для навчання модулів дозволить повертати більш точні результати обробки, класифікації, пріоритазації та формування вимог. Включення додаткових бізнес-метрик забезпечить не лише технічну доцільність впровадження вимог, а й економічну обґрунтованість. Також, для покращення якості згенерованих Acceptance Criteria можна додати механізми перевірки результатів.

Розроблена система є потужним інструментом для автоматизації процесу роботи з відгуками користувачів. Проведений аналіз підтвердив, що інтеграція сучасних моделей обробки природної мови та адаптивних алгоритмів може значно спростити процес формування вимог, забезпечуючи водночас високу якість результатів.

ВИСНОВКИ

Дана атестаційна робота присвячена проблемі обробки великої кількості відгуків і створення на їх базі вимог до програмного забезпечення. Ця робота охоплювала повний цикл дослідження, проектування, реалізації, тестування та аналізу результатів. Основна мета полягала в тому, щоб створити систему, яка дозволяє автоматизувати процес обробки зворотного зв’язку користувачів і на основі цього формувати чіткі та структуровані вимоги для подальшої розробки програмного забезпечення.

У ході виконання роботи були досягнуті були розроблені та інтегровані модулі, кожен із яких виконує конкретну функцію в рамках автоматизації процесу обробки текстів та генерації вимог. Здійснено аналіз основних методів і технологій, що застосовуються для обробки природної мови, зокрема токенізації, лемматизації, виділення іменованих сутностей, класифікації, пріоритизації та автоматичного формування вимог.

Результати обчислювального експерименту продемонстрували, що система здатна ефективно автоматизувати ключові етапи аналізу зворотного зв’язку. Інтеграція модулів забезпечує узгоджену роботу, дозволяючи уникнути розривів у процесі обробки даних.

Серед переваг розробленої системи можна виділити наступні пункти:

1) економія часу та ресурсів працівників, що зменшує потребу у витратах людських ресурсів;

2) можливість масштабування системи для роботи з великими наборами даних;

3) можливість допрацювання кожного модуля відповідно до специфіки домену;

4) використати додаткові інструменти для уточнення сутностей перед передачею до GPT.

Розроблена система має широкийс пектр застосування у проектах розробки програмного забезпечення, які отримують відгуки корситувачів та мають

потребу в їх обробці. Програмний продукт може бути інтегрованим у платформи зворотного зв’язку та системи управління проектами.

Однак існують напрямки, які можуть бути вдосконалені для подальшого підвищення ефективності та точності. Серед можливих покращень виділяються розширення навчальних наборів для всіх модулів системи, мультимовна підтримка, доналаштування моделей на специфічних для домену даних.

Новизна роботи полягає у сучасному підході з інтеграцією трансформерів (GPT, Sentence Transformers, DistilBERT) для повного циклу обробки тексту – від аналізу зворотного зв’язку до формування вимог. Також використано інтеграцію іменованих сутностей у генерування Acceptance Criteria, що покращує специфічність, точність і відповідність сформованих вимог.

Порівняно з іншими підходами, використання сучасних моделей обробки природної мови, забезпечує значні переваги. Традиційні системи аналізу тексту часто покладаються на правила або обмежені алгоритми машинного навчання, які вимагають значної ручної праці для налаштування. Натомість обраний підхід дозволяє автоматизувати більшість процесів завдяки використанню трансформерів, які мають здатність виявляти глибокі зв’язки між словами, контекстами та сутностями.

Таким чином, розроблена система не лише демонструє високу точність і ефективність, а й має потенціал для подальшого розвитку, що зробить її ще більш корисною для широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Svynkin D., Sytnykova Y. Automatic Generation of Software Requirements Based on User Feedback Using Natural Language Processing Algorithms // *Learning & Teaching: In the World after the War: Conference Proceedings of III International Scientific & Practical Conference*. Kharkiv : H.S. Skovoroda Kharkiv National Pedagogical University, 2024. 298 p.

2. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models Third Edition. 2024. URL: https://web.stanford.edu/~jurafsky/slp3/ed3bookaug20_2024.pdf (дата звернення 15.10.2024)

3. Wiegers K., Beatty J. Software Requirements, Third Edition. Redmond : Microsoft Press, 2013. 672 p.

4. Goodfellow I., Bengio Y. Courville A. Deep Learning. Cambridge : The MIT Press, 2016. 800 p.

5. Bird S., Klein E., Loper E. Natural Language Processing with Python. Sebastopol : O'Reilly Media, 2009. 479 p.

6. Pohl K., Rupp C. Requirements engineering fundamentals, Second Edition. New York : Rocky Nook, 2015. 163 p.

7. Cohn M. Agile Estimating and Planning. Upper Saddle River : Prentice Hall, 2006. 330 p.

8. Zelle J. Python Programming: An Introduction to Computer Science, Third Edition. Portland : Franklin, Beedle & Associates Inc., 2016. 552p.

9. Muller A., Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists. Sebastopol : O'Reilly Media, 2016. 400 p.

10. International Institute of Business Analysis. A guide to the Business Analysis Body of Knowledge, Version 3.0. Toronto : IIBA, 2015. 502p.