МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Моцал Володимир Романович

Програмний модуль автоматизації формування тестових запитів для генеративних систем/ Software module for automating of the test requests for generative systems

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Випускна кваліфікаційна робота

Виконав: студент групи КІ-41
Моцал Володимир Романович

Керівник Батько Ю.М.

2025

АНОТАЦІЯ

Моцал В.Р. Програмний модуль автоматизації формування тестових запитів для генеративних систем. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2025.

Робота написана обсягом 73 сторінки і містить 15 рисунків, 3 таблиці, 3 додатки та 25 джерел за переліком посилань. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою кваліфікаційної роботи є розробка програмного модуля автоматизації формування тестових запитів для генеративних систем на основі додавання ключових слів.

В кваліфікаційній роботі на основі аналізу існуючих алгоритмів обробки текстів на природній мові та корекції вхідних запитів відповідно до стандартів генеративних моделей реалізовано програма-помічник для формування та обробки текстових запитів до генеративних моделей. Проведено тестові дослідження та моделювання архітектури програмного додатку, а також програмно реалізовано прикладний програмний додаток на мові програмування Java з використання цифрової бібліотеки Swing, що підтвердило коректність обраної структури та запропонованого алгоритму.

Розроблений програмний продукт є ефективним засобом з простим інтерфейсом, що дозволяє вирішувати проблему формування та опрацювання текстових запитів до генеративних моделей.

Ключові слова: ШТУЧНИЙ ІНТЕЛЕКТ, ГЕНЕРАТИВНА МОДЕЛЬ, ПРИРОДЛЯ МОВА, КОРИСТУВАЦЬКИЙ ЗАПИТ.

ANNOTATION

Motsal V.R. Software module for automating the formation of test queries for generative systems. – Manuscript.

Qualification work for obtaining the degree of "bachelor" in specialty 123 "Computer Engineering", educational and professional program. Western Ukrainian National University, Ternopil, 2025.

The work is written in 73 pages and contains 15 figures, 3 tables, 3 appendices and 25 sources according to the list of references. The volume of graphic material is 2 sheets of A3 format.

The purpose of the qualification work is to develop a software module for automating the formation of test queries for generative systems based on the addition of keywords.

In the qualification work, based on the analysis of existing algorithms for processing natural language texts and correcting input queries in accordance with the standards of generative models, an assistant program for forming and processing text queries to generative models was implemented. Test studies and modeling of the software application architecture were conducted, and an application software application was also implemented in the Java programming language using the Swing digital library, which confirmed the correctness of the selected structure and the proposed algorithm.

The developed software product is an effective tool with a simple interface that allows solving the problem of forming and processing text queries to generative models.

Keywords: ARTIFICIAL INTELLIGENCE, GENERATIVE MODEL, NATURAL LANGUAGE, USER REQUEST.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПКК	–	Персональний компактний комп’ютер
ЕП	–	Електронний помічник
UGI	–	User Graphics Interface
GPT	–	Generative Pre-trained Transformer
NL	–	Nature language

ВСТУП

Актуальність роботи. Штучний інтелект в сучасному світі стає все більш поширеним, оскільки він забезпечує спосіб покращення практики навчання та виконання рутинних завдань. Нове покоління штучного інтелекту виводить дані технології на новий історичний етап. Це не тільки глибоко впливає та формує виробництво, життя та способи комунікації всього суспільства, але й фундаментально змінює саме людство. З моменту появи перших чат-ботів на основі технології генерованого штучним інтелектом контенту вони постійно розвиваються та впроваджують інновації. Зокрема, генеративний попередньо навчений трансформатор (GPT) це тип штучного інтелекту, розроблений для полегшення обробки та розуміння природної мови. Ця інноваційна технологія має потенціал революціонізувати спосіб навчання та роботи, надаючи нові інструменти для покращення якості та практики. GPT – це тип штучного інтелекту, який використовує методи глибокого навчання для аналізу та генерації тексту. Він використовувався для різних застосувань, включаючи переклад мови, реферування тексту та відповіді на запитання. Дана модель базується на архітектурі трансформатора, нейронній мережі, призначеній для фіксації довгострокових залежностей у тексті. Такий підхід може аналізувати великі обсяги даних та виявляти закономірності та тенденції.

Завдяки індивідуальній та інтерактивній допомозі, технології штучного інтелекту дозволяють покращити навчальний процес та збільшити кількість відповідей які можуть отримати люди в різних сферах. При цьому даний підхід є дешевим в порівнянні з використанням вчителів або тренерів. Завдяки наданню персоналізованої та інтерактивної підтримки програмні помічники мають потенціал для розвитку автономності та звичок самостійного навчання та самовдосконалення у людей. Сучасні помічники можуть генерувати текст, відповідати на запитання та вести природні розмови. Вони навчаються з використанням великих обсягів текстових даних і можуть бути точно налаштований для виконання конкретних завдань.

Проте використання засобів штучного інтелекту має ряд проблем. Серед основних проблем це достовірність навчальних даних і коректність при формуванні запитів до мовної моделі. Якщо в першому випадку на сьогодні існують багато відкритих баз для проведення навчання, інформація в яких перевірена та верифікована експертами, то друга проблема є набагато складнішою. Чим точніше буде сформовано запит під структуру відповідної моделі, тим адекватнішу відповідь зможе згенерувати система.

Отже актуальною задачею є розроблення програмного модуля формування тестових запитів для мовних моделей генеративних систем на аналізу запитів природньою мовою та додавання ключових слів.

Об’єкт дослідження – системи автоматизованого аналізу та генерування повідомлень. Предмет дослідження – алгоритми обробки та формування повідомлень для генеративних моделей на основі природніх мов. Метою кваліфікаційної роботи є розроблення програмного модуля формування тестових запитів для мовних моделей генеративних систем. Для досягнення мети такі завдання:

– дослідити сучасні підходи та технології штучного інтелекту;

– провести класифікацію технологій обробки природніх мов;

– провести дослідження програмних помічників з елементами штучного інтелекту;

– проаналізувати мовні моделі генератичних систем;

– розробити алгоритм формування тестових запитів для генеративних систем на основі ключових слів;

– спроектувати та провести тестування внутрішньої архітектури програмного модуля формування тестових запитів для генеративних систем штучного інтелекту.

За результатами роботи опубліковано тези доповіді на II Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі» [1]. Копію публікації наведено у додатку В.

1 ТЕХНОЛОГІЇ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ГЕНЕРАТИВНИХ ПОМІЧНИКІВ

1.1 Сучасні підходи та технології штучного інтелекту

Штучний інтелект (ШІ) – це технологія, яка дозволяє комп'ютерам та машинам імітувати людське навчання, розуміння, вирішення проблем, прийняття рішень, креативність та автономію.

Програми та пристрої, оснащені штучним інтелектом, можуть бачити та ідентифікувати об'єкти. Вони можуть розуміти людську мову та реагувати на неї. Також можуть навчатися на основі нової інформації та досвіду. Вони можуть надавати детальні рекомендації користувачам та експертам. Програми з елементами штучного інтелекту можуть діяти самостійно, замінюючи необхідність людського інтелекту чи втручання (класичним прикладом є безпілотний автомобіль). Але у 2024 році більшість досліджень у галузі штучного інтелекту, зосереджені на проривах у генеративному штучному інтелекті (genAI) – технології, яка може створювати оригінальний текст, зображення, відео та інший контент. Технології генеративного штучного інтелекту побудовані на машинному навчанні(ML) та глибокому навчанні.

В компанії IBM для розуміння взаємодії стеку технологій, що використовуються при створенні програмних додатків з елементами штучного інтелекту наводять наступну архітектуру (рисунок 1.1).

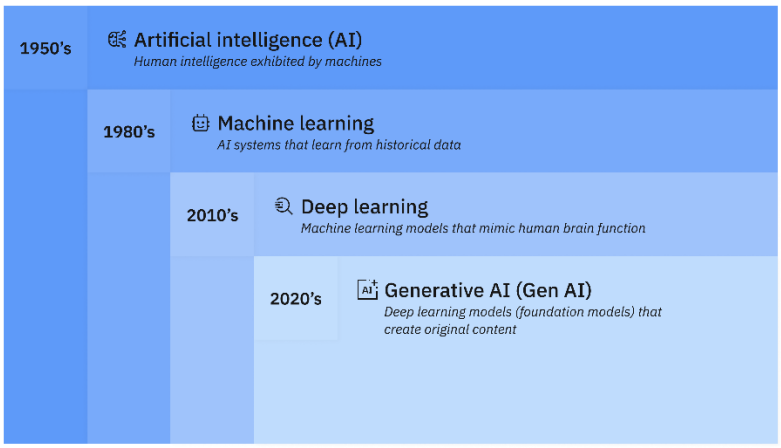

Рисунок 1.1 – Взаємодія технологій штучного інтелекту

Безпосередньо під штучним інтелектом знаходиться машинне навчання, яке передбачає створення моделей шляхом навчання алгоритму для прийняття прогнозів або рішень на основі даних. Воно охоплює широкий спектр методів, що дозволяють комп'ютерам навчатися та робити висновки на основі даних без необхідності їхнього явного програмування для виконання конкретних завдань. Існує багато типів методів або алгоритмів машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень, випадковий ліс, методи опорних векторів (SVM) , k-найближчих сусідів (KNN), кластеризацію та інші. Кожен із цих підходів підходить для різних типів проблем і даних.

Але один з найпопулярніших типів алгоритмів машинного навчання називається нейронною мережею (або штучною нейронною мережею). Нейронні мережі моделюються за структурою та функціями людського мозку. Вони складається з взаємопов'язаних шарів вузлів (аналогічних нейронам), які працюють разом для обробки та аналізу складних даних. Нейронні мережі добре підходять для завдань, що передбачають виявлення складних закономірностей та взаємозв'язків у великих обсягах даних. Найпростіша форма машинного навчання називається навчанням з учителем , яке передбачає використання маркованих наборів даних для навчання алгоритмів для класифікації даних або точного прогнозування результатів. У навчанні з учителем люди поєднують кожен навчальний приклад з міткою виходу. Мета полягає в тому, щоб модель вивчила відповідність між входами та виходами в навчальних даних, щоб вона могла передбачати мітки нових, невидимих даних.

Глибоке навчання – це підмножина машинного навчання, яка використовує багатошарові нейронні мережі, що називаються глибокими нейронними мережами, що точніше імітують складну здатність людського мозку приймати рішення. Глибокі нейронні мережі включають вхідний шар, щонайменше три, але зазвичай сотні прихованих шарів і вихідний шар. Це відбувається на відміну від нейронних мереж, що використовуються в класичних моделях машинного навчання, які зазвичай мають лише один або два приховані шари. Ці численні шари дозволяють здійснювати навчання без учителя. Вони можуть

автоматизувати вилучення ознак з великих, немаркованих та неструктурованих наборів даних і робити власні прогнози щодо того, що представляють ці дані.

Оскільки глибоке навчання не потребує втручання людини, воно дозволяє використовувати машинне навчання у величезних масштабах. Воно добре підходить для обробки природної мови (NLP), комп'ютерного зору та інших завдань, що передбачають швидку та точну ідентифікацію складних закономірностей та зв'язків у великих обсягах даних. Певна форма глибокого навчання забезпечує роботу більшості застосувань штучного інтелекту (ШІ) у нашому сучасному житті. Сучасні технології навчання можна сформувати у наступні групи, що проілюстровано на рисунку 1.2:

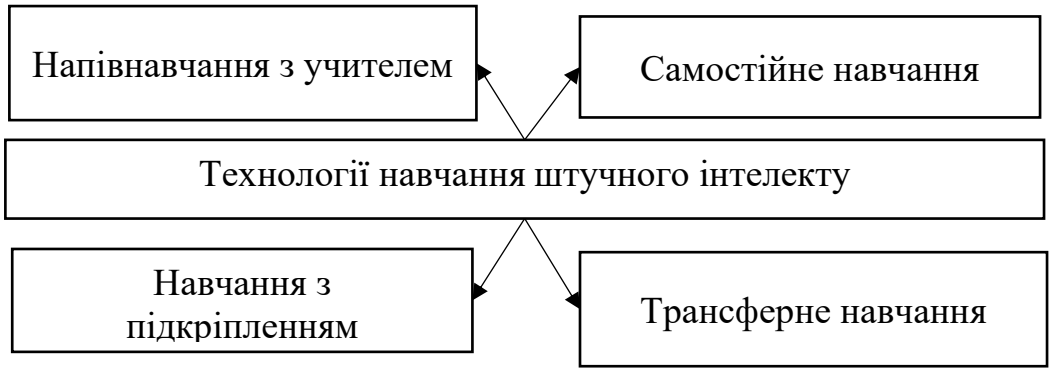

Рисунок 1.2 – Технології навчання штучного інтелекту

Напівнавчання з учителем, яке поєднує навчання з учителем та без учителя, використовуючи як марковані, так і немарковані дані для навчання моделей ШІ для завдань класифікації та регресії. Самостійне навчання генерує неявні мітки з неструктурованих даних, а не покладається на позначені набори даних для контрольних сигналів. Навчання з підкріпленням – даний підхід передбачає навчання методом спроб і помилок та винагород, а не шляхом вилучення інформації з прихованих закономірностей. Трансферне навчання, в якому знання, отримані в результаті виконання одного завдання або набору даних, використовуються для покращення продуктивності моделі в іншому пов'язаному завданні або іншому наборі даних.

Генеративний штучний інтелект, який іноді називають «генеративним штучним інтелектом», стосується моделей глибокого навчання, які можуть

створювати складний оригінальний контент, такий як довгий текст, високоякісні зображення, реалістичне відео чи аудіо тощо, у відповідь на запит або запит користувача. На високому рівні генеративні моделі кодують спрощене представлення своїх навчальних даних, а потім використовують це представлення для створення нових моделей, подібних, але не ідентичних, до вихідних даних. Генеративні моделі роками використовуються в статистиці для аналізу числових даних. Але протягом останнього десятиліття вони еволюціонували для аналізу та генерації складніших типів даних. Ця еволюція збіглася з появою трьох складних типів моделей глибокого навчання:

Варіаційні автоенкодери або VAE, які були представлені у 2013 році, дозволили моделям генерувати кілька варіантів контенту у відповідь на підказку чи інструкцію.

Моделі дифузії, вперше представлені у 2014 році, які додають «шум» до зображень, доки вони не стануть невпізнанними, а потім видаляють шум для створення оригінальних зображень у відповідь на підказки.

Трансформери (трансформаторні моделі), які навчаються на послідовних даних для генерації розширених послідовностей контенту (таких як слова в реченнях, фігури на зображенні, кадри відео або команди в програмному коді). Трансформери є основою більшості сучасних генеративних інструментів штучного інтелекту, що створюють заголовки, включаючи ChatGPT та GPT-4, Copilot, BERT, Bard та Midjourney.

Використання технологій з елементами штучного інтелекту має різні переваги в різних галузях та застосуваннях (рисунок 1.3).

Штучний інтелект може автоматизувати рутинні, повторювані та часто виснажливі завдання, включаючи цифрові завдання, такі як збір даних, введення та попередня обробка, а також фізичні завдання, такі як комплектування товарів на складі та виробничі процеси. Така автоматизація звільняє можливості для роботи над більш цінною та креативною роботою.

Покращене прийняття рішень. Незалежно від того, чи використовується він для підтримки рішень, чи для повністю автоматизованого прийняття рішень, ШІ забезпечує швидші та точніші прогнози та надійні рішення на основі даних.

У поєднанні з автоматизацією ШІ дозволяє компаніям використовувати можливості та реагувати на кризи в міру їх виникнення, в режимі реального часу та без втручання людини.

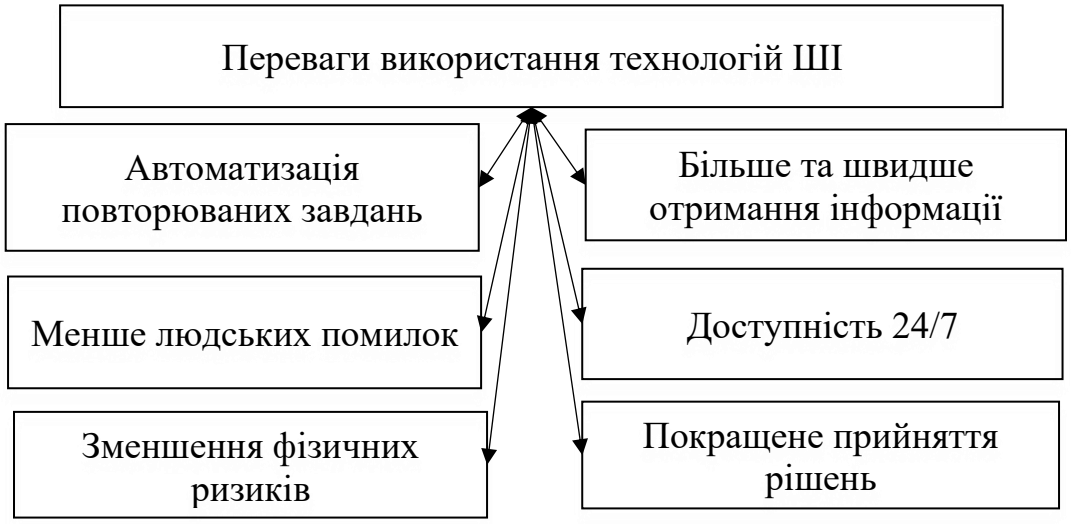

Рисунок 1.3 – Переваги використання штучного інтелекту

Штучний інтелект може зменшити кількість людських помилок різними способами, від керівництва людьми через правильні етапи процесу до виявлення потенційних помилок до їх виникнення та повної автоматизації процесів без втручання людини. Це особливо важливо в таких галузях, як охорона здоров'я, де, наприклад, хірургічна робототехніка під керуванням штучного інтелекту забезпечує стабільну точність. Алгоритми машинного навчання можуть постійно покращувати свою точність та ще більше зменшувати кількість помилок, оскільки вони отримують більше даних та «навчаються» на досвіді.

Штучний інтелект завжди увімкнений, доступний цілодобово та забезпечує стабільну продуктивність щоразу. Такі інструменти, як чат-боти зі штучним інтелектом або віртуальні помічники, можуть зменшити потреби в персоналі для обслуговування клієнтів або підтримки. В інших сферах застосування, таких як обробка матеріалів або виробничі лінії, ШІ може допомогти підтримувати стабільну якість роботи та рівень продуктивності, коли використовується для виконання повторюваних або виснажливих завдань.

Автоматизуючи небезпечну роботу, таку як контроль за тваринами, поводження з вибуховими речовинами, виконання завдань у глибоких океанських водах, на великих висотах або в космосі, штучний інтелект може усунути необхідність наражати працівників на ризик травмування або ще гірше. Хоча вони ще не вдосконалені, безпілотні автомобілі та інші транспортні засоби пропонують потенціал для зниження ризику травмування пасажирів.

1.2 Технологій та сфери застосування обробки природніх мов

Обробка природної мови (NLP) – це підгалузь інформатики та штучного інтелекту (ШІ), яка використовує машинне навчання, щоб дозволити комп'ютерам розуміти людську мову та спілкуватися нею.

Обробка природної мови дозволяє комп'ютерам і цифровим пристроям розпізнавати, розуміти та генерувати текст і мовлення, поєднуючи обчислювальну лінгвістику, моделювання людської мови на основі правил разом зі статистичним моделюванням, машинним навчанням та глибоким навчанням.

Дослідження NLP допомогли розпочати еру генеративного штучного інтелекту , починаючи від комунікативних навичок моделей великих мов (LLM) і закінчуючи здатністю моделей генерації зображень розуміти запити. НЛП вже є частиною повсякденного життя багатьох людей, живлячи пошукові системи, спонукаючи чат-ботів до обслуговування клієнтів за допомогою голосових команд, голосових GPS-систем та цифрових помічників, що відповідають на запитання, на смартфонах, таких як Alexa від Amazon, Siri від Apple та Cortana від Microsoft. НЛП також відіграє зростаючу роль у корпоративних рішеннях, які допомагають оптимізувати та автоматизувати бізнес-операції, підвищити продуктивність співробітників та спростити бізнес-процеси.

Використання технологій обробки природніх мов спрощує для людей спілкування та співпрацю з машинами, дозволяючи їм робити це природною

людською мовою, яку вони використовують щодня. Використання даної технології створює переваги в багатьох галузях та застосуваннях (рисунок 1.4).

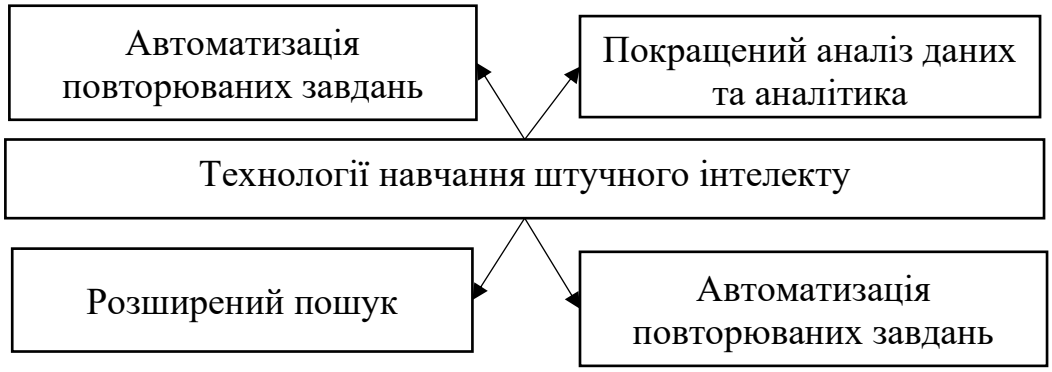

Рисунок 1.4 – Переваги використання технології обробки природніх мов

NLP особливо корисний для повної або часткової автоматизації таких завдань, як підтримка клієнтів, введення даних та обробка документів. Наприклад, чат-боти на базі NLP можуть обробляти рутинні запити клієнтів, звільняючи людських агентів для вирішення складніших питань. В обробці документів інструменти NLP можуть автоматично класифікувати, витягувати ключову інформацію та узагальнювати вміст, зменшуючи час та кількість помилок, пов'язаних з ручною обробкою даних. NLP сприяє перекладу, конвертуючи текст з однієї мови на іншу, зберігаючи при цьому значення, контекст та нюанси.

NLP покращує аналіз даних, дозволяючи витягувати корисну інформацію з неструктурованих текстових даних, таких як відгуки клієнтів, публікації в соціальних мережах та новинні статті. Використовуючи методи текстового інтелектуального аналізу, NLP може виявляти закономірності, тенденції та настрої, які не одразу очевидні у великих наборах даних. Аналіз настроїв дозволяє витягувати з тексту суб'єктивні якості, ставлення, емоції, сарказм, плутанину чи підозру. Це часто використовується для спрямування комунікацій до системи або особи, яка, найімовірніше, зробить наступну відповідь.

Це дозволяє краще розуміти вподобання користувачів, ринкові умови та громадську думку. Інструменти NLP також можуть виконувати категоризацію та

узагальнення величезних обсягів тексту, що полегшує аналітикам визначення ключової інформації та ефективніше прийняття рішень на основі даних.

NLP надає переваги пошуку, дозволяючи системам розуміти намір, що стоїть за запитами користувачів, надаючи точніші та контекстуально релевантні результати. Замість того, щоб покладатися виключно на зіставлення ключових слів, пошукові системи на базі NLP аналізують значення слів і фраз, що спрощує пошук інформації, навіть коли запити розпливчасті або складні. Це покращує взаємодію з користувачем, чи то в веб-пошуку, пошуку документів чи корпоративних системах даних.

NLP використовує передові мовні моделі для створення тексту, подібного до людського, для різних цілей. Попередньо навчені моделі, такі як GPT-4, можуть генерувати статті, звіти, маркетингові тексти, описи продуктів і навіть творче письмо на основі підказок, наданих користувачами. Інструменти на базі НЛП також можуть допомогти в автоматизації таких завдань, як написання електронних листів, публікацій у соціальних мережах або юридичної документації. Розуміння контексту, тону та стилю в NLP забезпечує зв'язність, релевантність та відповідність згенерованого контенту запланованому повідомленню, заощаджуючи час і зусилля на створення контенту, зберігаючи при цьому якість.

1.3 Аналіз програмних помічників з елементами штучного інтелекту

Розвиток штучного інтелекту докорінно змінив підхід до створення інформаційного та візуального контенту. Завдяки глибокому навчанню на великих масивах даних, новітні алгоритми здатні імітувати людське мислення й творчість на такому рівні, що результати їхньої роботи дедалі складніше відрізнити від продукту, створеного людиною. Це відкрило широкі можливості для впровадження ШІ у редакторську діяльність, цифровий маркетинг, дизайн,

освітні проєкти та навіть програмування. У цьому контексті доцільним є аналіз найбільш відомих інструментів, які стали доступними широкому загалу.

Одним із найпоширеніших ШІ-інструментів є мовна модель ChatGPT від компанії OpenAI. Завдяки гнучкому архітектурному рішенню GPT-4, ця система здатна підтримувати осмислений діалог, генерувати логічно побудовані тексти, перекладати мовні одиниці, резюмувати документи, а також надавати допомогу у вирішенні програмних завдань. ChatGPT ефективно застосовується у сфері освіти для створення адаптивних навчальних матеріалів, а також у бізнесі – зокрема, для написання рекламних описів, відповідей клієнтам і підготовки презентацій. У сфері програмування модель здатна виявляти синтаксичні помилки у коді, пояснювати алгоритми та навіть генерувати фрагменти програмного забезпечення відповідно до технічного запиту.

Bing AI, розроблений Microsoft, інтегрується безпосередньо в браузер Edge і поєднує функції пошукової системи, чат-бота й генератора зображень. Його особливістю є здатність взаємодіяти з вебвмістом у реальному часі, що дозволяє підсумовувати статті, відповідати на контекстні запити та доповнювати відповіді графічними матеріалами. Це дає змогу використовувати інструмент у журналістиці – для швидкої обробки інформації з кількох джерел, в освітній діяльності – для пояснення термінів і понять безпосередньо у браузері, а також у діловому спілкуванні – для генерації комунікаційних шаблонів.

Midjourney функціонує як генератор зображень, який приймає текстові підказки та інтерпретує їх у вигляді візуального контенту. Завдяки високому рівню деталізації, цей інструмент широко застосовується в дизайні, зокрема у розробці візуальної концепції для ігор, ілюстрації книг і створенні оригінального контенту для соціальних мереж. Midjourney особливо цінується серед фахівців із креативної індустрії, які прагнуть візуалізувати абстрактні ідеї до етапу технічної реалізації. Його використання дозволяє заощадити час на створення ескізів і значно розширює можливості художнього самовираження.

Інструмент D-ID пропонує користувачам змогу створювати віртуальні відео з анімованими обличчями, які озвучують заданий текст. Система поєднує візуальні ШІ-моделі з мовними технологіями, зокрема інтегрується з ChatGPT

для автоматичного генерування сценаріїв. Цей інструмент активно використовується в електронному навчанні для створення відеоуроків з "живими" віртуальними викладачами, у рекламній сфері – як засіб персоналізації повідомлень до клієнтів, а також у культурних проєктах – наприклад, для реконструкції мови історичних персонажів у музеях та архівах.

DALL·E 2, ще одна розробка OpenAI, є генератором зображень, здатним не лише створювати ілюстрації на основі текстових описів, а й редагувати вже наявні зображення, змінюючи або доповнюючи їхні елементи. Завдяки цим функціям DALL·E 2 активно використовується у видавничій сфері для створення обкладинок книг, в освітніх платформах – для генерації навчальних схем та ілюстрацій, а також у сфері реклами – для створення унікальних візуальних рішень, які не потребують участі дизайнерів або фотографів. Широка доступність сервісу робить його зручним інструментом для фахівців початкового рівня, а також для використання в академічному середовищі.

Stable Diffusion 2 вирізняється серед аналогічних інструментів відкритим кодом, що дає змогу не лише використовувати його як стандартний генератор зображень, але й адаптувати або вдосконалювати модель відповідно до індивідуальних потреб. Завдяки цьому Stable Diffusion часто застосовується у наукових лабораторіях для візуалізації результатів, у сфері цифрового мистецтва для створення інсталяцій, а також у навчальному процесі – як засіб вивчення алгоритмів глибокого навчання. Можливість запуску моделі локально забезпечує високий рівень контролю над даними та процесами генерації.

Для більш детального аналізу програмних продуктів, що представлені на сучасному світовому ринку програмних розробок було проведено порівняльне дослідження самих поширених представників з кожного напряму. Оскільки дані розробки виконують різні операції, проте в їх основі закладено принципи автоматичної генерації нового контенту на основі аналізу запитів користувачів та використанні закладеної моделі. Проте, можна виділити їх основні параметри, на основі яких здійснити їх порівняння. Серед таких критеріїв виділяють можливість вільного використання, вартість користування, наявність критеріїв

налаштування роботи відповідних моделей тощо. Узагальнене порівняння програмних продуктів на ведено в таблиці 1.1.

Назва (розробник)	Доступність	Виконувані завдання	Сфери застосування	Вартість
ChatGPT (OpenAI)	Web-сайт	Генерація текстів	Освіта Бізнес Програмування	Безкоштовно/ підписка
Bing AI (Microsoft)	Web-сайт	Web-браузер помічник	Журналістика Пошукова оптимізація Освітній супровід	Безкоштовно (в браузері)
Midjourney (Midjourney)	Discord	Генерування зображень	Дизайн Маркетинг Мода	Підписка
D-ID (D-ID)	Web-сайт/ мобільний додаток	Синтез звуків/анімації	Освіта Брендинг Музеї	Безкоштовно/ підписка
DALL·E 2 (OpenAI)	Web-сайт	Синтез зображення	Реклама Видавнича справа Освітній контент	Безкоштовно/ підписка
Stable Diffusion 2 (Stability AI)	Web-сайт	Синтез зображення	Наукові дослідження Культура і мистецтво Академічні проєкти	Безкоштовно (тільки через сайт)

Сучасні інструменти штучного інтелекту демонструють величезний потенціал у галузі створення контенту. Їхнє використання дозволяє значно прискорити робочі процеси, знизити витрати та розширити творчі можливості користувача. Системи на кшталт ChatGPT та DALL·E 2, вже сьогодні стали невід’ємною частиною професійної діяльності в освіті, бізнесі, культурі й ІТ.

Подальші дослідження у сфері генеративного ШІ мають на меті не лише вдосконалення точності моделей, а й розвиток етичних механізмів їх застосування, що забезпечить сталий розвиток у цифрову епоху.

1.4 Висновки до розділу та постановка завдань кваліфікаційної роботи

В першому розвоілі було здійсноне огляд та та проведено ряд класифікацій підходів які використовуються під час створення програмних додатків з елементами штучного інтелекту, розглянуто основні сфери застосування технології аналізу та обробки природніх мов та здійснено оглях функціональних можлливостей сучасних програмних помічників з елементами штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1) дослідити сучасні підходи та технології штучного інтелекту;
2) провести класифікацію технологій обробки природніх мов;
3) провести дослідження програмних помічників з елементами штучного інтелекту;
4) проаналізувати мовні моделі генеративних систем;
5) розробити алгоритм формування тестових запитів для генеративних систем на основі ключових слів;
6) спроектувати та провести тестування внутрішньої архітектури програмного модуля формування тестових запитів для генеративних систем штучного інтелекту.

2 АЛГОРИТМИ ОБРОБКИ ТА АНАЛІЗУ ПРИРОДНІХ МОВ

2.1 Алгоритми обробки текстових повідомлень на природніх мовах

Обробка природної мови (NLP) зосереджена на взаємодії між комп'ютерами та людською мовою. Вона дозволяє машинам розуміти, інтерпретувати та генерувати людську мову таким чином, щоб вона була одночасно змістовною та корисною. Ця технологія не лише підвищує ефективність та точність обробки даних, але й забезпечує глибокі аналітичні можливості, що є кроком до кращого прийняття рішень. Ці переваги досягаються завдяки різноманітним складним алгоритмам NLP.

Обробка природної мови – це розділ штучного інтелекту, який зосереджується на взаємодії між комп’ютерами та людьми за допомогою природної мови. Основна мета NLP – дати комп’ютерам змогу розуміти, інтерпретувати та генерувати людську мову цінним чином. Впровадження алгоритмів NLP може значно покращити ваші операції, обробляючи такі завдання, як обслуговування клієнтів, витягуючи значущу інформацію з великих обсягів неструктурованих даних, а також автоматизуючи значну частину рутинних завдань.

Алгоритми НЛП – це обчислювальні методи, що використовуються для аналізу, розуміння та генерації людської мови. Ці алгоритми можна розділити на три основні типи: символьні алгоритми, статистичні алгоритми та гібридні алгоритми (рисунок 2.1).

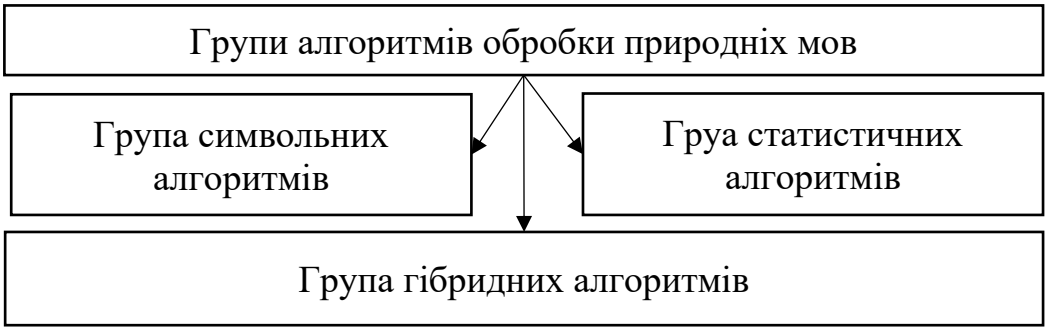

Рисунок 2.1 – Класифікація груп алгоритмів обробки природніх мов

Символічні алгоритми, також відомі як алгоритми на основі правил або знань, спираються на заздалегідь визначені лінгвістичні правила та представлення знань. Ці алгоритми використовують словники, граматики та онтології для обробки мови. Вони легко інтерпретуються та можуть обробляти складні лінгвістичні структури, але їх розробка та підтримка вимагають значних ручних зусиль. Символьні алгоритми ефективні для виконання певних завдань, де правила чітко визначені та узгоджені, таких як розбір речень та визначення частин мови.

Статистичні алгоритми використовують математичні моделі та великі набори даних для розуміння та обробки мови. Ці алгоритми спираються на ймовірності та статистичні методи для виведення закономірностей та взаємозв'язків у текстових даних. Методи машинного навчання, включаючи навчання з учителем та без учителя, зазвичай використовуються в статистичному НЛП. Приклади включають класифікацію тексту, аналіз настроїв та моделювання мови. Статистичні алгоритми є більш гнучкими та масштабованими, ніж символьні алгоритми, оскільки вони можуть автоматично навчатися на основі даних та вдосконалюватися з часом з більшою кількістю інформації.

Гібридні алгоритми поєднують елементи символьного та статистичного підходів, щоб використовувати сильні сторони кожного з них. Ці алгоритми використовують методи на основі правил для виконання певних лінгвістичних завдань та статистичні методи для інших. Інтегруючи обидва методи, гібридні алгоритми можуть досягти вищої точності та стійкості в застосунках NLP. Вони можуть ефективно керувати складністю природної мови, використовуючи символьні правила для структурованих завдань та статистичне навчання для завдань, що потребують адаптивності та розпізнавання образів. Наприклад, це включає поєднання граматичних правил з машинним навчанням для розбору або використання статистичних методів для вдосконалення систем на основі правил.

Використання даних груп алгоритмів тісно повязане з виконанням певних операцій при обробці тестових повідомлень на природніх мовах. Серед основних операцій, що виконують в процесі обробки можна виділити такі (рисунок 2.2):

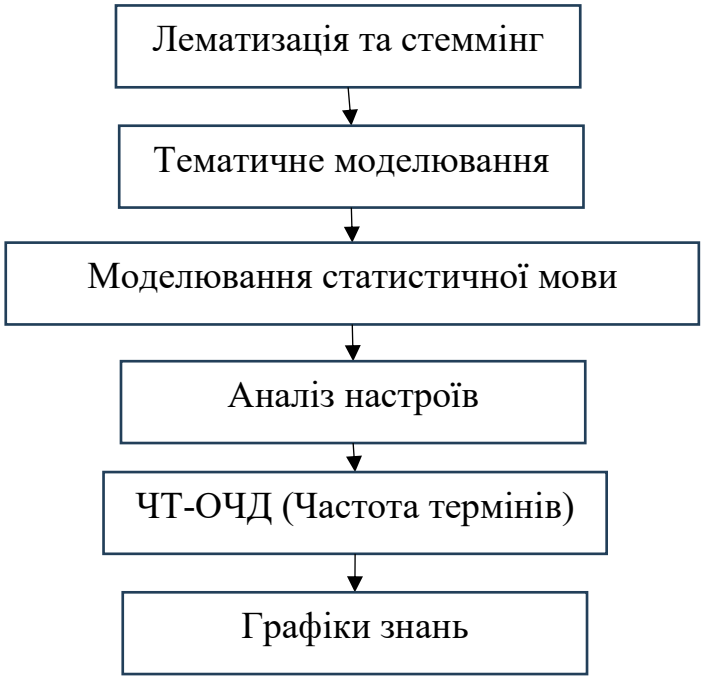

Рисунок 2.2 – Етапи обробки природніх мов

Лематизація та стеммінг – це методи, що використовуються для редукції слів до їхньої основної або кореневої форми, що допомагає нормалізувати текстові дані. Обидва методи спрямовані на нормалізацію текстових даних, що полегшує аналіз та порівняння слів за їхньою основною формою, хоча лематизація, як правило, є точнішою завдяки врахуванню лінгвістичного контексту. Лематизація зводить слова до їхньої словникової форми, або леми, гарантуючи, що слова аналізуються в їхній базовій формі (наприклад, «бігає» стає «бігти»). Стемінг обрізає кінці слів, щоб видалити суфікси. Це простіше та швидше, але менш точно ніж лематизація, оскільки іноді «корінь» не є реальним (наприклад, «навчатись» стає «вчити»).

Тематичне моделювання – це метод, який використовується для виявлення прихованих тем або питань у колекції документів. Він допомагає виявляти абстрактні теми, що зустрічаються в наборі текстів.

Статистичне моделювання мови передбачає прогнозування ймовірності послідовності слів. Це допомагає зрозуміти структуру та ймовірність послідовностей слів у мові.

Аналіз настроїв – це процес класифікації тексту на категорії позитивного, негативного або нейтрального настрою.

Це працює за допомогою кількох технік (рисунок 2.3):

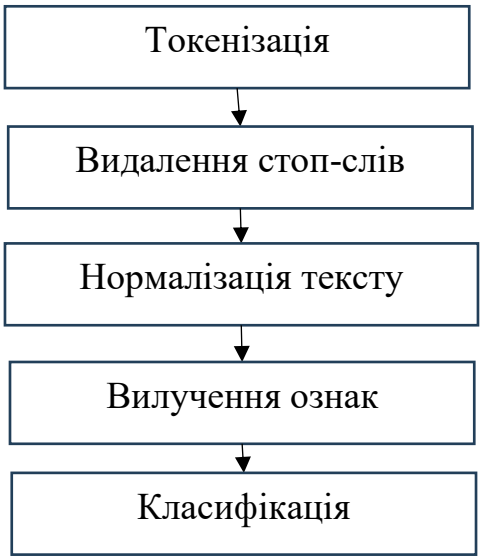

Рисунок 2.3 – Етапи аналізу настроїв в процесі обробки текстових повідомлень

Токенізація – це перший крок процесу, під час якого текст розбивається на окремі слова або «лексеми». Потім кожна лексема аналізується окремо. Видалення стоп-слів проводиться на наступному кроці. Стоп-слова, такі як «ой» чи «ага», які не несуть значного значення, видаляються, щоб зосередити увагу на важливих словах. Нормалізація тексту проводиться з метою надати слову його основу або кореневу форму. Наприклад, слово «розписати» може бути скорочено до кореневого слова «писати». Це називається стеммінгом або лематизацією.4. Вилучення ознак з тексту призводить до витягування ключових рис тексту або слів, які допоможуть визначити настрій. До них можуть належати прикметники на кшталт «хороший», «поганий», «чудовий» тощо. На останньому етапі проводиться класифікація. Настрої класифікуються за допомогою алгоритмів машинного навчання. Це може бути бінарна класифікація (позитивне/негативне), багатокласова класифікація (щасливий, сумний, злий тощо) або шкала (оцінка від 1 до 10). Однак, сарказм, іронія, сленг та інші фактори можуть ускладнювати точне визначення настроїв. Тим не менш, його часто використовують компанії для оцінки настроїв клієнтів щодо своїх продуктів чи послуг за допомогою відгуків клієнтів.

ЧТ-ОЧД (частота термінів – обернена частота документів) – це статистичний показник, що використовується в обробці природної мови та пошуку інформації для оцінки важливості слова в документі відносно колекції документів (корпусу). TF- ОЧД поєднує два компоненти: частоту термінів (ЧТ) та обернену частоту документів (ОЧД).

Частота терміна (ЧТ) – вимірює, як часто слово з'являється в документі. Вища частота свідчить про більшу важливість. Якщо термін часто з'являється в документі, він, ймовірно, релевантний до вмісту документа та обчислюється за формулою:

$$\text{ЧТ(i)} = \frac{\text{Частота зустрічання } i\text{–го терміну в тексті}}{\text{Частота зустрічання усіх термінів в тексті}}.$$

При використанні даного терміну слід враховувати настіпні обмеження:

– ЧТ не враховує глобальної важливості терміна в усьому тексті.

– Такі поширені слова, як «ой» або «і», можуть мати високі бали ЧТ, але не є значущими для розрізнення документів.

Обернена частота документів (ОЧД) зменшує вагу поширених слів у кількох документах, водночас збільшуючи вагу рідкісних слів. Якщо термін зустрічається в меншій кількості документів, він, ймовірно, буде змістовним і конкретним. Дана характеристика обчислюється за формулою:

$$\text{ОЧД(i)} = \frac{\text{Загальна кількість документів}}{\text{Кількість документів з і–тим терміном}}.$$

При використанні даного терміну слід враховувати настіпні обмеження:

– Логарифм використовується для зменшення впливу дуже великих або дуже малих значень, забезпечуючи відповідне масштабування балів ОЧД.

– Це також допомагає збалансувати вплив термінів, які з'являються в надзвичайно малій або надзвичайно великій кількості документів.

Графи знань представляють інформацію через мережу сутностей та їх зв'язків. Вони використовуються для ілюстрації пов'язаності документів.

2.2 Структури та технічні особливості мовних моделей генеративних помічників

Генеративний штучний інтелект (ШІ) — це клас моделей машинного навчання, здатних створювати нові дані, що імітують характеристики навчального набору. Він базується на вивченні розподілу ймовірностей вхідних даних і формуванні нових зразків, які відповідають цьому розподілу. У контексті обробки природних мов генеративні моделі призначені для генерації тексту, що є лінгвістично коректним і семантично змістовним. Вони використовують статистичні або нейронні підходи для моделювання послідовностей слів, забезпечуючи здатність створювати контекстуально узгоджені та когерентні мовні структури. Такі моделі знаходять застосування у різних задачах, включаючи автоматичний переклад, діалогові системи, узагальнення тексту та генерацію відповіді, що робить їх важливим інструментом сучасної лінгвістичної інженерії. Загалом, генеративний ШІ працює у три фази (рисунок 2.4):

Рисунок 2.4 – Етапи роботи генеративного штучного інтелекту

Навчання генеративний ШІ починає з «базової моделі», моделі глибокого навчання, яка служить основою для кількох різних типів застосувань генеративного ШІ. Найпоширенішими базовими моделями сьогодні є великі мовні моделі (LLM), створені для програм генерації тексту. Але також існують базові моделі для генерації зображень, відео, звуку або музики, а також мультимодальні базові моделі, які підтримують кілька видів контенту. Щоб

створити базову модель, фахівці навчають алгоритм глибокого навчання на величезних обсягах відповідних неструктурованих, немаркованих даних, таких як терабайти або петабайти тексту, зображень чи відео з Інтернету. В результаті навчання створюється нейронна мережа з мільярдів параметрів, закодованих у представленнях сутностей, шаблонів та зв'язків у даних, які можуть автономно генерувати контент у відповідь на підказки. Це базова модель. Цей процес навчання є ресурсоємним, трудомістким та дорогим. Він вимагає тисяч кластерних графічних процесорів (GPU) та тижнів обробки, що зазвичай коштує мільйони доларів. Проєкти з відкритим кодом, такі як Llama-2 від Meta, дозволяють розробникам штучного інтелекту уникнути цього кроку та пов'язаних з ним витрат.

Далі модель необхідно налаштувати на конкретне завдання генерації контенту. Це можна зробити різними способами, зокрема:

– Точне налаштування, яке включає передачу моделі специфічних для програми позначених даних, запитань або підказок, які програма, ймовірно, отримає, та відповідних правильних відповідей у потрібному форматі.

– Навчання з підкріпленням та людським зворотним зв'язком (RLHF), за якого користувачі оцінюють точність або релевантність результатів моделі, щоб модель могла самовдосконалюватися. Це може бути так само просто, як введення тексту людьми або відповідь на виправлення чат-боту чи віртуальному помічнику.

На етапі генерації розробники та користувачі регулярно оцінюють результати своїх генеративних ШІ-додатків та додатково налаштовують модель навіть раз на тиждень для більшої точності або релевантності. Натомість сама базова модель оновлюється набагато рідше, можливо, раз на рік або 18 місяців. Ще одним варіантом покращення продуктивності застосунку на основі штучного інтелекту є доповнена генерація пошуку (RAG), метод розширення базової моделі для використання відповідних джерел поза навчальними даними для уточнення параметрів для більшої точності або релевантності.

На сьогоднішній день існують велика кількість генеративних моделей, що модуть бути вільно використані для створення програмних додатків різної

складності. Серед них одна з найпоширеніших Chat Generative Pre-trained Transformer (ChatGPT). Зі 175 мільярдами параметрів Chat Generative Pre-trained Transformer є однією з найбільших і найпотужніших моделей обробки даних штучним інтелектом, що зараз представлені на ринку, що пояснює його розширення застосування в різних галузях. Відповіді ChatGPT значно перевершують відповіді попередніх систем штучного інтелекту, тому що вони більше схожі на людські. ChatGPT набув широкого розповсюдження у комерційному секторі. Експерти вважають, що програми та технології попередньо навченого трансформатора Chat Generative слугуватимуть трампліном для більш складних систем штучного інтелекту.

Одним із ключових досягнень Chat GPT є використання архітектури трансформатора. Архітектура трансформатора особливо добре підходить для завдань обробки природної мови, таких як переклад мови та генерація тексту, завдяки своїй здатності ефективно обробляти довгі послідовності даних. Архітектура трансформатора складається з шарів самоуваги, які дозволяють моделі зважувати важливість різних слів або фраз на кожному вході. Це дозволяє моделі краще зрозуміти контекст і значення вхідних даних, а також генерувати більш узгоджені та узгоджені відповіді. Окрім шарів самоуваги, архітектура трансформатора також включає рівні прямого зв'язку та залишкові з'єднання. Ці компоненти дозволяють моделі вивчати складніші шаблони в даних і краще фіксувати зв'язки між різними словами чи фразами (рисунок 2.5).

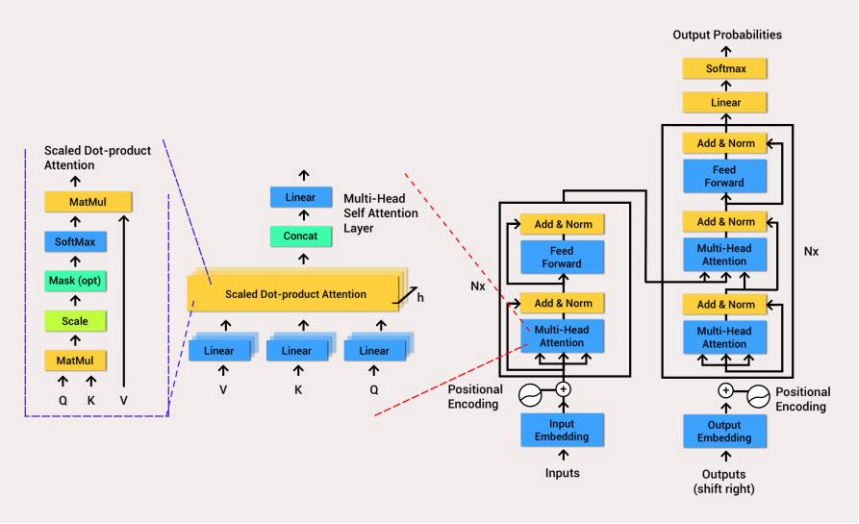

Рисунок 2.5 – Узагальнена структура моделі ChatGPre Transformer

Ще однією ключовою перевагою ChatGPT є його здатність вчитися на великих обсягах даних. Він попередньо навчений на масивному наборі даних тексту, що дозволяє йому зрозуміти закономірності та структуру природної мови. Це попереднє навчання дозволяє ChatGPT генерувати відповіді, які є більш схожими на людину та менш роботизованими. Процес попереднього навчання передбачає подачу моделі великого набору даних тексту та навчання його передбаченню наступного слова в кожній послідовності. Це дозволяє моделі вивчати закономірності та структуру мови, а також зв'язки між різними словами та фразами. Ще однією перевагою Chat GPT є його здатність адаптуватися до різних контекстів і ситуацій. Він може зрозуміти контекст розмови та генерувати відповідні відповіді на основі цього контексту. Це дозволяє йому вести більш природні та різноманітні розмови з користувачами. Наприклад, якщо користувач запитує чат-бота про погоду, чат-бот може відповісти поточними погодними умовами або прогнозом на найближчі дні. Якщо потім користувач запитує про погоду в іншому місці, чат-бот може зрозуміти зміну контексту та надати відповідну інформацію.

2.3 Алгоритм формування тестових запитів для генеративних систем на основі ключових слів

В процесі дослідження запиті користувачів, що надсилаються до моделей генеративного штучного інтелекту було виділено ряд факторів, які впливають на кіневий результат отриманої відповіді. Серед основних причин отримання неякісних відповідей можна вибілити такі:

– запити є дуже короткі або надзвичайно довгі;

– наявність великої кількісті орфографічних помилок;

– відсутність ключових слів-характеристи для формування відповіді;

– запити сформовані надто загально і можуть охоплювати декілька галузей знань.

Тому врахувавши даний набір помилок які користувачі допускають в процесі використання генеративних моделей, а також дослідовши особливості функціонування моделі ChatGPT було запропоновано наступний алгоритм підвищення якості формування запитів:

1) Початок процесу обробки запиту.

2) Ввід запиту користувача.

3) Здійснення аналізу запиту з метою визначення, чи потребує він обробки природної мови (Natural Language Processing, NLP). Якщо обробка потрібна перехід на пунк 4, інакше 8.

4) Розбиття повідомлення на окремі слова та словосполучення (сегментація тексту на лексичні одиниці).

5) Виділення набору наказових дій (ідентифікація інструкцій або команд, що містяться в запиті).

6) Перевірка потреби у додатковій обробці. Якщо так, то перехід на пунк 7, інакше 8.

7) Класифікація повідомлення за типом (наприклад, інформаційне, запитальне, інструктивне тощо).

8) Формування запиту з доданими метаданими (контекстною інформацією).

9) Відправка запиту до моделі GPT.

10) Отримання результату обробки від мовної моделі.

11) Завершення процесу формування запиту та виводу відповіді.

Даний алгоритм використовує алгоритми обробки природніх мов для виділення основної суті запиту та дозволяє його доповнити додатковою технічною інформацією для спрощення функціонування генеративної моделі та отримання більш професійних відповідей на запити користувача. Використання апарату обробки природніх мов дозволяє виділяти ключові терміни запиту, відкидати слова-паразити та зменшувати навантаження на генеративну модель, завдаки більш чіткому та стандартизованому запиту. Узагальнену блок схему запропованого алгоритму формування та обробки запитів/відповідей до генеративної моделі ChartGPT наведено на рисунку 2.6.

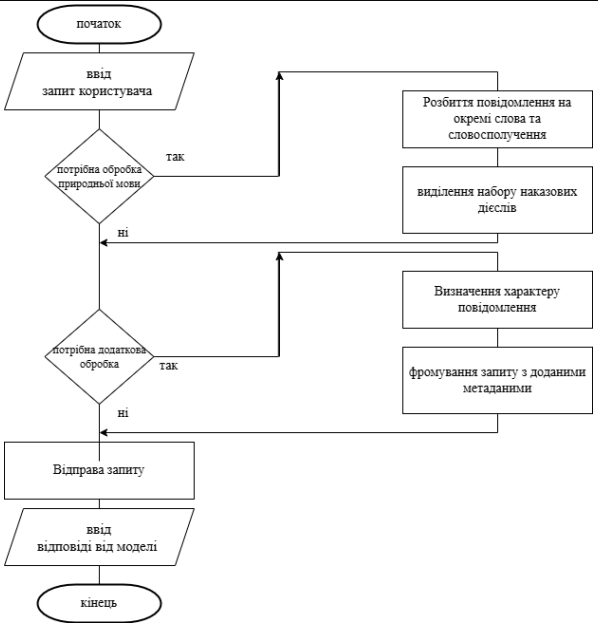

Рисунок 2.6 – Блок-схема алгоритму формування та обробки запитів до генеративної моделі ChartGPT

Проведений аналіз запропонованого алгоритму дозволив зробити ряд висновків про його функціональні можливості та провести його оцінку:

– Даний алгоритм має просту послідовність кроків, що дозволяє його легко реалізувати та інтегрувати в різні системи;

– використання алгоритмів обробки природніх мов дозволяє значно спростити процес формування відповіді завдяки видаленню малоінфоматичних даних;

– алгоритм можна модифікувати в залежності відпоставлених задач чи апаратних обмежень виконавця.

Проте було відмічено і ряд недоліків:

– для коректної роботи алгоритму необхідна якісна обробка вхідного запиту, зокрема залежність від обраного набору слів-паразитів;

– можливість проблемної обробки запитів, що містять велику кількість орфографічних помилок, описок, маловідомих скорочень тощо.

3 ПРОГРАМНИЙ ДОДАТОК ФОРМУВАННЯ ТА ОБРОБКИ ЗАПИТІВ ДО ГЕНЕРАТИВНОЇ МОДЕЛІ CHATGPT

3.1 Структура та графічний інтерфейс програми-помічника

Сьогодні програми з інтеграцією ChatGPT є актуальними через зростаючий попит на інтелектуальні помічники, здатні автоматизувати рутинні завдання, генерувати контент, надавати консультації та покращувати користувацький досвід завдяки природній мовній взаємодії. Вони забезпечують швидкий доступ до потужних моделей штучного інтелекту у зручній, персоналізованій формі, що особливо цінується у сферах освіти, бізнесу, обслуговування та розробки. Вибір мови програмування Java та графічної бібліотеки Swing для реалізації застосунку JavaGPT є виваженим рішенням, що ґрунтується на низці технічних, практичних та архітектурних міркувань. По-перше, мова Java є однією з найпоширеніших та найстабільніших мов програмування, яка зарекомендувала себе як кросплатформне середовище з високим рівнем переносності. Це дозволяє розгортати застосунок на різних операційних системах (Windows, macOS, Linux) без необхідності зміни вихідного коду. У випадку JavaGPT така незалежність є важливою, оскільки застосунок розрахований на широку аудиторію користувачів, які можуть мати різне програмне забезпечення. Крім того, Java має високий рівень підтримки мережевих операцій і роботи з API, що є критичним для інтеграції з OpenAI GPT. Завдяки потужним бібліотекам Java для обробки HTTP-запитів, обробки JSON та асинхронного програмування, забезпечується стабільна й безпечна взаємодія з віддаленим сервером моделей. Важливою перевагою є також наявність перевіреної часом бібліотеки Swing, яка входить до складу стандартної бібліотеки Java. Swing надає розробникам повний контроль над виглядом і поведінкою інтерфейсу, що дозволяє створити адаптивне та налаштовуване графічне середовище. Для застосунку типу JavaGPT, що включає декілька вікон (головне вікно, вікно перегляду чатів, інформаційне вікно), можливість швидко створювати ізольовані та водночас інтегровані інтерфейси має вирішальне значення. Крім того, використання Swing було виправдано з

огляду на продуктивність і низькі системні вимоги. На відміну від сучасніших бібліотек на основі JavaFX або сторонніх фреймворків, Swing не потребує додаткових залежностей або модулів, що спрощує поширення та підтримку застосунку. Зокрема, для невеликих застосунків, що не вимагають складної анімації або мультимедійного контенту, Swing залишається оптимальним вибором за критеріямипростоти реалізації. Не менш важливою є наявність великої кількості навчальних матеріалів, документації та прикладів коду, що дозволяє забезпечити швидке розгортання проєкту, його тестування та підтримку. Завдяки цьому JavaGPT може легко інтегрувати нові модулі або адаптуватись до змін у зовнішніх API. З точки зору архітектури застосунку, Java і Swing ідеально підтримують принципи об'єктно-орієнтованого програмування, забезпечуючи розмежування логіки інтерфейсу, бізнес-логіки та зовнішніх залежностей.

Для програмної реалізації запропонованого алгоритму було спроектовано структуру майбутньої програми-помічника. Принципи проектування структури базують на використанні вже готових елементів для реалізації простих частин програми (інтерфейси, роботу з мережею тощо) та можливості простої інтеграції запропонованого алгоритму. Приклад архітектури наведено на рисунку 3.1.

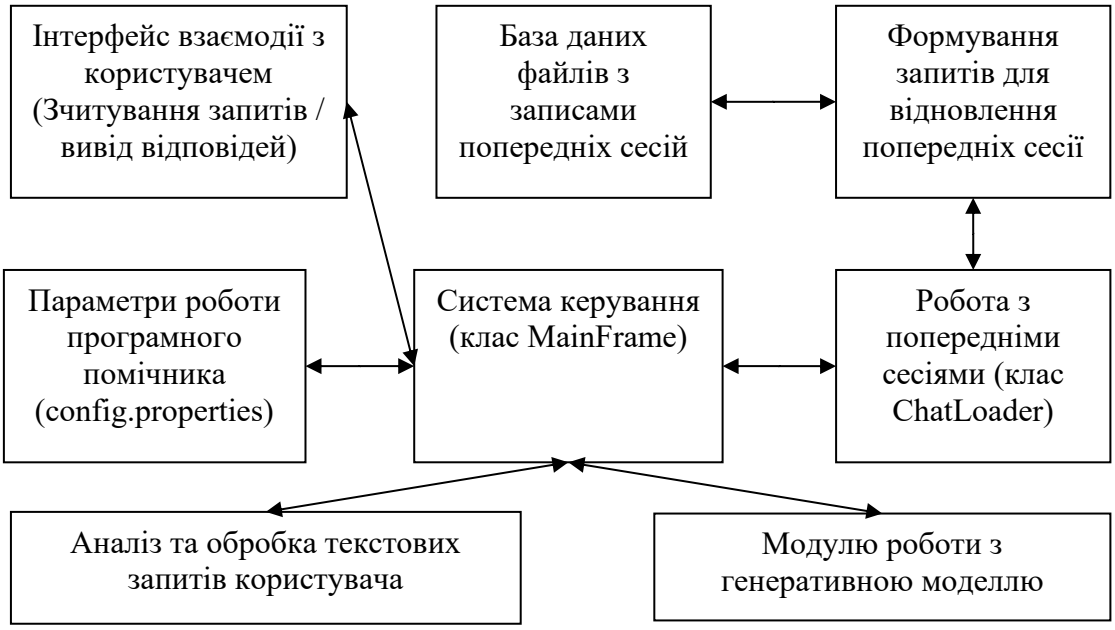

Рисунок 3.1 – Структурна схема програмного формування та обробки запитів

Запропонована структура для програмного додатку із графічним інтерфейсом користувача, реалізована на мові програмування Java. Його основна функціональна мета полягає у забезпеченні інтерактивної комунікації користувача з моделями штучного інтелекту OpenAI GPT через діалоговий інтерфейс. Архітектура цього застосунку сформована з урахуванням принципів модульності, конфігурованості, масштабованості та чіткого розділення обов'язків між структурними елементами, що значною мірою полегшує супровід, модифікацію та подальший розвиток системи.

Центральним компонентом застосунку виступає клас MainFrame, який реалізує основне вікно інтерфейсу користувача. Цей модуль виконує функції ініціації та обробки чат-сесій, інтеграції з API OpenAI, організації обробки конфігураційних файлів, а також підтримки повноцінного управління історією чатів, включаючи збереження, завантаження та видалення діалогів. Крім того, даний компонент забезпечує динамічне налаштування параметрів інтерфейсу, таких як розмір шрифтів, тема оформлення та розміри вікна, відповідно до уподобань користувача або системних умов.

Допоміжним інструментом для роботи з архівами чатів є компонент ChatLoader, який реалізує окреме вікно для перегляду, сортування, перейменування та видалення збережених чат-сесій. Його функціональність орієнтована на ефективне управління файловою системою, де зберігаються дані користувача.

Функціональність взаємодії з API моделі GPT інкапсульована в межах класу MainFrame, де реалізовано логіку формування запиту, надсилання його до серверу та отримання відповіді. Окрім цього, у межах того ж модуля реалізовані методи обробки конфігураційного файлу config.properties, який відіграє ключову роль у зовнішньому налаштуванні поведінки системи без потреби зміни програмного коду. Зокрема, через цей файл задаються параметри доступу до API (наприклад, API-ключ), налаштування проксі-сервера, візуальні теми інтерфейсу та інші системні опції.

Одним із фундаментальних принципів використаної архітектури є чітке дотримання розділення обов'язків між модулями. Кожен з них реалізує окремий

аспект функціонування системи: інтерфейс користувача, обробку чатів, взаємодію з API або управління конфігурацією. Така структурна організація дозволяє мінімізувати міжмодульні залежності, що, своєю чергою, полегшує тестування, рефакторинг та інтеграцію нових функцій. Архітектура також демонструє ізоляцію від зовнішніх систем, за винятком прямої взаємодії з API OpenAI, яка є єдиним зовнішнім інтерфейсом системи.

Інтеграція з файловою системою здійснюється переважно через компоненти MainFrame та ChatLoader, які взаємодіють з локальними файлами користувача з метою збереження або відтворення історії чатів. Межа між логікою застосунку і файловою системою чітко окреслена. Файли зберігаються у форматі, визначеному внутрішніми правилами іменування, що базуються на вмісті чатів та поточному сеансі.

Завдяки логічному розподілу функцій, гнучкій системі налаштувань, а також обмеженій залежності від зовнішніх компонентів, дана система забезпечує високу надійність, зручність у користуванні та технічну адаптивність. Подальше розширення функціоналу можливе без значного втручання у вже реалізовану структуру, що свідчить про стабільність та ефективність обраного архітектурного підходу.

Для отримання підтвердження, що запропонована архітектура дозволоить у повній мірі реалізувати попередьо описаний алгоритм необхідно провести її моделювання для отримання підсумкової оцінки. Для початку необхідно переконатись, що запропонований підхід до побудови внутрішньої архутектури програмного додатку дозволить користувачам різного ранку отримувати доступ до усіх необхідних для виконання завдання функцій. Для цього було обрато проведення тестування на основі use case моделі що входить до UML. Використання даного типу моделявання дозволяє проілюструвати блоки системи з якими може взаємодіяти окрема рупа користувачів, що називаються акторами. Це дозволить вже на початкових етапах внести корекції в структуру програми та виявити її вразливі місця. Моделювання проводилось для двох груп акторів «Користувач» та «Адміністратор». Результати, які були отримані в процесі моделювання наведено на рисунку 3.2.

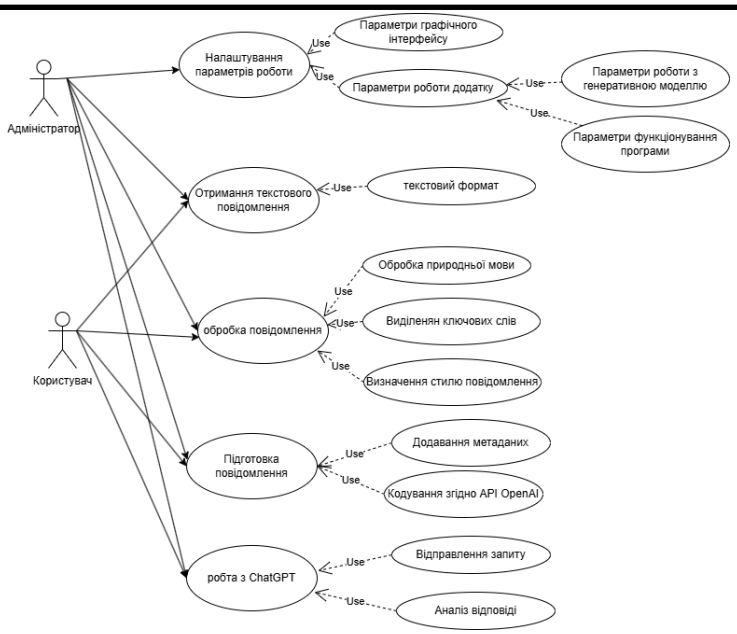

Рисунок 3.2 – Діаграма прецедентів для моделювання доступу різних груп користувачів

Подана use case-діаграма моделює функціональні сценарії взаємодії між користувачами та системою, яка забезпечує інтерфейс для роботи з генеративною мовною моделлю. У діаграмі беруть участь два актори: адміністратор та користувач. Кожен з яких виконує специфічні ролі в межах роботи з програмним забезпеченням.

Адміністратор є відповідальним за налаштування параметрів роботи системи. Цей процес охоплює конфігурацію графічного інтерфейсу, визначення загальних параметрів функціонування програми, а також встановлення режимів взаємодії з генеративною мовною моделлю. Через цей механізм досягається адаптивність системи до різноманітних сценаріїв використання та можливість централізованого контролю.

Користувач взаємодіє з системою у кількох послідовних етапах. Спочатку здійснюється введення текстового повідомлення, що надалі обробляється за допомогою вбудованого мовного процесора. На цьому етапі виконується розпізнавання ключових слів, обробка природної мови та визначення стилістичних характеристик повідомлення. Таке багаторівневе аналізування дозволяє інтерпретувати запит користувача максимально точно, враховуючи лексичний контекст і намір.

Після обробки повідомлення система переходить до етапу підготовки запиту. У процесі формування запиту здійснюється додавання метаданих, які розширюють семантичний простір вхідного повідомлення, а також кодування відповідно до специфікації API OpenAI. Це дозволяє гарантувати відповідність запиту технічним вимогам генеративної моделі. Фінальна стадія полягає у безпосередній взаємодії з ChatGPT. Застосунок надсилає підготовлений запит до серверної частини OpenAI, а після отримання відповіді виконує її семантичний аналіз та подальше відображення для користувача. Цикл завершується презентацією згенерованого тексту, що може слугувати відповіддю, порадою або наступним кроком у діалозі.

Таким чином, діаграма use case демонструє чітко структуровану модель взаємодії, в якій обробка повідомлення користувача проходить декілька етапів трансформації, забезпечуючи точність, релевантність та відповідність комунікації із системою на основі мовної моделі. Система організована згідно з принципами розділення обов'язків між користувачем, який ініціює запит, і адміністратором, що забезпечує технічні умови для її функціонування.

Іншим важливим фактором цілісності та коректної роботи програмної системи є уникнення внутрішніх колізій переході даних між різними етапами їх обробки. Для даного дослідження використовуєть діаграма послідовності взіємодії окремиз модулів системи (рисунок 3.3).

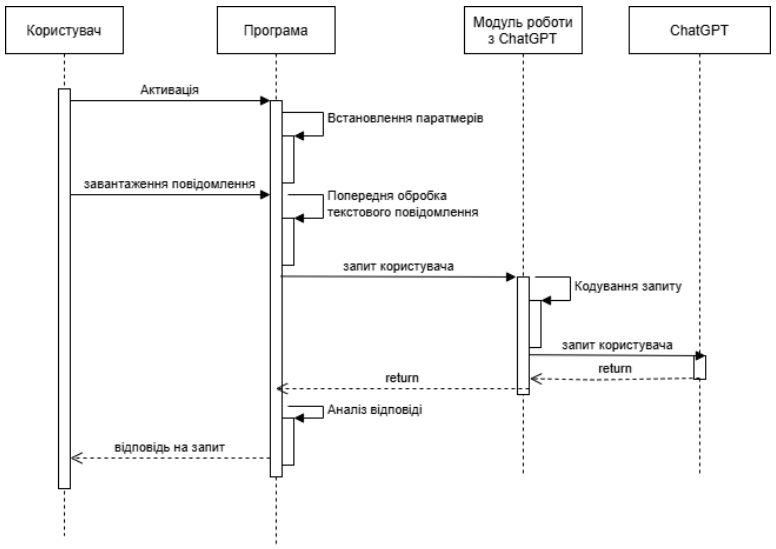

Рисунок 3.3 – Діаграма послідовності кроків взаємодії користувача та програми

Отримані резуьтати моделювання проілюстрували коректність та повноту запропонованої внутріщньої арітектури програмного помічника. Було показано, що в процесі робти користувачі різних груп мають повних доступ до необхідного набору функцій, а внутрішня структура є цілісною та не містить елементи, що можуть конфліктувати один з одним.

Оскільки даний помічник буде активно використовуватись користувачами в процесі пошуку інформації або виконання поставлених завдань ще одним важливим етапом є розробка графічного інтерфейсу користувача. Хоча спідкування відувається в текстовому режимі, вокористання візуальних елементів для управління роботою програмизначно підвищить рівень зручності при роботі з нею. При розробці дизайну програмного додатку було враховано переваги та недоліки програм аналогів. Результати проектування графічного інтерфейсу наведено на рисунку 3.4.

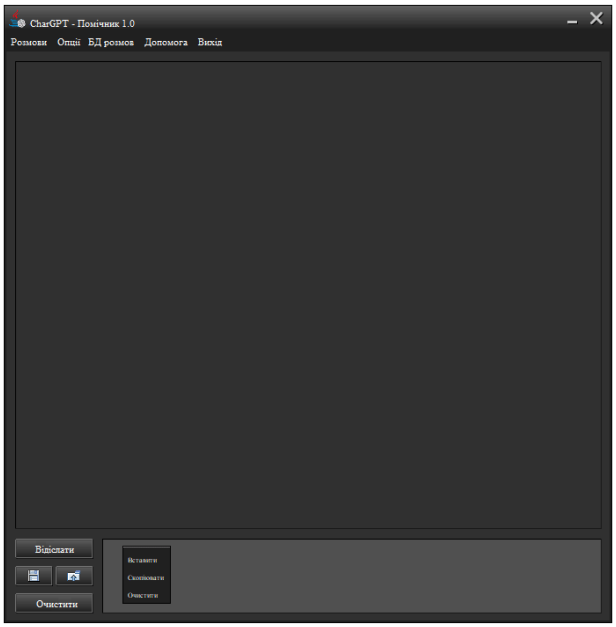

Рисунок 3.4 – Дизайн головного вікна програмного додатку

На рисунку 3.4 представлено головне вікно користувацького інтерфейсу застосунку, який реалізовано за допомогою бібліотеки Swing на мові Java. Інтерфейс побудовано відповідно до класичної структурної моделі десктопного застосунку, що передбачає вертикальне розміщення основних елементів: меню, робочої області та панелі керування. У верхній частині вікна розташовано рядок

заголовка, який містить назву застосунку: «ChatGPT – Помічник 1.0», а також стандартні кнопки керування вікном (згорнути, розгорнути, закрити). Безпосередньо під заголовком розміщено головне горизонтальне меню, яке включає пункти:

- «Розмови»;
- «Опції»;
- «БД розмов»;
- «Допомога»;
- «Вихід».

Це меню слугує для навігації між ключовими функціями системи: керування чатами, доступ до налаштувань, роботи з базою даних чатів, довідкових матеріалів і завершення сеансу.

Центральну частину інтерфейсу займає велика темна текстова панель, яка виконує роль області виводу діалогу. Вона призначена для відображення історії спілкування з мовною моделлю та містить простір для обробки повідомлень користувача та відповідей від GPT.

Нижня панель інтерфейсу виконує функцію панелі введення та керування. У лівій частині розміщено основну кнопку «Відправити», а також блок допоміжних інструментів з трьома піктограмами для збереження, завантаження або вставки сеансів.

У правій частині нижньої панелі передбачено випадаюче меню або набір дій, серед яких: «Вставити», «Скопіювати» та «Очистити». Ці функції орієнтовані на керування вмістом текстового поля або буфера обміну.

Інтерфейс витримано у темному кольоровому оформленні, що знижує навантаження на зір і відповідає сучасним UI-трендам. Усі елементи розміщені логічно та забезпечують інтуїтивну взаємодію користувача з системою. Представлений графічний інтерфейс забезпечує чітку, зручну та функціонально завершену форму взаємодії користувача з генеративною моделлю GPT.

Процес проектування та моделювання структури програмного помічника продемонстрував цілісність запропонованої розробки та можливість її програмної реалізації засобами мови програмування Java.

3.2 Особливості реалізацій функцій програми-помічника

Клас MainFrame виконує роль центрального елемента архітектури програмного забезпечення JavaGPT, реалізуючи основні механізми взаємодії з користувачем, керування інтерфейсом та інтеграції з зовнішніми сервісами. У рамках реалізації на платформі Java Swing даний клас наслідується від JFrame, що дозволяє йому виступати в якості головного вікна застосунку з відповідною графічною оболонкою. В таблиці 3.1 наведено основні розроблені класи та структури даних програми-помічника.

Таблиця 3.1 – Структурні елементи програми

Компонент	Основні обов'язки
MainFrame	Управління графічним інтерфейсом, координація API, обробка подій
config.properties	Централізоване управління конфігурацією
OpenAiService	Зв'язок та потокове передавання даних через API ChatGPT
ArrayList<ChatMessage>	Керування вмістом БД
ChatLoader	Завантаження та керування історією чату
Елементи графічного інтерфейсу користувача	Елементи інтерфейсу користувача та взаємодії

Під час створення екземпляра класу MainFrame здійснюється повна ініціалізація елементів інтерфейсу та налаштування візуальних і структурних параметрів головного вікна. Визначаються розміри вікна, встановлюється заголовок програми, активуються системні елементи керування, такі як кнопки мінімізації та завершення, а також формується структура панелі меню з подальшим прив'язуванням до неї необхідної логіки.

Усі функціональні компоненти інтерфейсу розміщуються на головній панелі contentPane, яка слугує базовим контейнером. Через неї організовується виведення відповідей мовної моделі у форматах RTF та HTML, а також

забезпечується введення повідомлень користувачем. Для цього використовуються відповідні графічні компоненти, що візуалізують як вхідну, так і вихідну інформацію, зокрема текстові області для діалогу та елементи керування у вигляді кнопок.

Функціональність взаємодії з користувачем реалізовано за допомогою системи обробки подій. Основні сценарії взаємодії, такі як натискання кнопок чи введення тексту супроводжуються активацією відповідних обробників подій, що викликають методи логіки застосунку. Наприклад, при ініціації відправлення повідомлення запускається процедура формування запиту до API OpenAI через метод submit.

Ключовим елементом зв'язку із зовнішнім генеративним сервісом є об'єкт service, який реалізує взаємодію з OpenAI через інтерфейс OpenAiService. Цей об'єкт формує HTTP-запити на основі введеного тексту, надсилає їх до API моделі ChatGPT, а отриману відповідь відображає у відповідному текстовому компоненті інтерфейсу. Така інтеграція забезпечує повний цикл обробки запиту від моменту введення до отримання відповіді.

Крім механізмів взаємодії та відображення, клас MainFrame реалізує систему конфігурації застосунку. Зчитування налаштувань здійснюється з конфігураційного файлу за допомогою об'єкта prop, що дозволяє динамічно змінювати параметри, такі як візуальне оформлення, розміри шрифтів, проксі-сервери та інші критично важливі параметри роботи програми.

Таким чином, MainFrame формує ядро інтерфейсної частини системи JavaGPT, виступаючи посередником між користувачем і сервісом штучного інтелекту, забезпечуючи не лише технічну комунікацію із зовнішнім API, але й повноцінне управління конфігураційними характеристиками програми.

Для налаштування параметрів роботи програми використовуються ряд змінних. Дані змінні були згруповані в залежності від сфери застосування та функціонального навантаження. Окремо було виділено групи для зберігання параметрів графічного інтерфейсу користувача, параметрів встановлення взаємодії між програмою та сервером генеративної моделі тощо. Сформовані групи параметрів та самі параметри наведено у таблиці 3.2.

Таблиця 3.2 – Групи параметрів для налаштування роботи програми

Категорія	Властивості	Площа впливу
Конфігурація API	apikey,timeout, model, maxTokens	Ініціалізація сервісу OpenAI
Налаштування мережі	proxyip, proxyport,proxytype	Конфігурація проксі-сервера HTTP
Поведінка інтерфейсу користувача	autoscroll, EnterToSubmit,Theme	Поведінка інтерфейсу користувача
Керування чатом	autotitle, chat_history, chatLocationOverride	Файлові операції та автоматизація
Налаштування дисплея	WindowSize,FontSize	Візуальна презентація

Програма працює послідовно активуючи ряд методів. Одним з перших активується метод перевірки можливості роботи в мережі internet. Позамовчуванню програма розраховує, що виконавець має прямий доступ до мережі. Проте, якщо це не так, то активується спроба підєднання до мережі через проксі сервер. Програмний код даної функції наведено нижче:

```
if(prop.getProperty("proxyip") != null &&
!prop.getProperty("proxyip").isEmpty() && prop.getProperty("proxyport")
!= null && !prop.getProperty("proxyport").isEmpty()) {
 if(prop.getProperty("proxytype").toLowerCase().equals("http"))
      System.setProperty("http.proxyHost",prop.getProperty("proxyip"));
      System.setProperty("http.proxyPort",prop.getProperty("proxyport"));
 }else
      if(prop.getProperty("proxytype").toLowerCase().equals("https")){
      System.setProperty("https.proxyHost", prop.getProperty("proxyip"));
      System.setProperty("https.proxyPort",prop.getProperty("proxyprt"));
      }else {
      System.getProperties().put( "proxySet", "true" );
      System.getProperties().put( "socksProxyHost",
prop.getProperty("proxyip") );
      System.getProperties().put( "socksProxyPort",
prop.getProperty("proxyport") );
      }
}
```

Даний фрагмент коду відповідає за налаштування проксі-серверу для програми, на основі конфігурацій, зчитаних з об’єкта *prop*, який містить параметри із зовнішнього файлу налаштувань (*config.properties*). Його основна мета – це встановити проксі-параметри залежно від типу обраного проксі-

серверу (HTTP, HTTPS або SOCKS). На початку відбувається перевірка, чи значення властивостей *proxyip* та *proxyport* не є порожніми та не дорівнюють *null*. Якщо обидві умови виконуються, тобто IP-адреса проксі-серверу та порт були вказані, виконується наступна перевірка типу проксі. Далі, в залежності від значення властивості *proxytype*, відбувається розгалуження:

– Якщо тип дорівнює "*http*", встановлюються системні властивості *http.proxyHost* та *http.proxyPort*, що застосовуються до HTTP-з'єднань.

– Якщо тип дорівнює "*https*", відповідно, конфігуруються *https.proxyHost* і *https.proxyPort*, що використовуються при роботі з HTTPS-з'єднаннями.

– Якщо ж тип не є ні HTTP, ні HTTPS, система переходить до налаштування SOCKS-проксі. У такому разі активується загальна проксі-конфігурація через параметр *proxySet*, а також задаються *socksProxyHost* і *socksProxyPort*.

Тобто дана частина коду є гнучким механізмом для динамічного налаштування параметрів мережевого підключення програми, що дозволяє використовувати проксі-сервери різних типів без необхідності змінювати сам код, а лише конфігураційний файл. Це особливо корисно в умовах корпоративних мереж, ізоляції середовищ або забезпечення додаткового рівня безпеки через маршрутизацію трафіку.

3.3 Тестування програмного додатку

Процес тестування проводився за наступним алгоритмом. Для отримання початкового варіанту відповіді користувач задавав деякий запит. Даний запит без додаткової обробки пересилався на сервер генеративної моделі, після чого користувачу демонструвалась його відповідь. Після чого, програмі повторно задавався то же запит, проте програма-помічник намагалась удосконалити запит в автоматичному режимі. Отриманий результат передавався користувачу. Третій

варіант запиту користувач формував самостійно на основі параметрів які пропонувала задати програма. Серед можливих параметрів удосконалення запиту були такі:

– стиль написання відповіді (художній, віршований тощо);

– рівень точності згенерованого тексту від 1 до 10 (1-повна відповідність джерелам, 10-максимальна переробка);

– розмір відповіді (кількість слів);

– тип джерел (книги, журнали, сайти, блоги тощо);

– мова джерела (тільки мова запиту, іншомовні джерела);

– давнина інформації (від якого року аналізувати джерела).

Даний перелік параметрів дозволяє більш точно описати необхідний результат відповіді. Для прикладу проаналізуємо три відповіді системи на питання: «Що таке Java?».

На рисунку 3.5 наведено результат відповіді програми без додаткової корекції.

Java — це об'єктно-орієнтована мова програмування, яка дозволяє створювати платформа-незалежні застосунки, що виконуються у віртуальній машині Java (JVM).

Рисунок 3.5 – Відповідь на питання без корекції

На рисунку 3.6 наведено результат відповіді програми на те саме запитання коли в програмі була актовована опція автоматичної корекції запиту:

Java — це об'єктно-орієнтована, високорівнева мова програмування загального призначення, розроблена компанією *Sun Microsystems* у 1995 році (згодом придбана *Oracle Corporation*). Вона призначена для написання програм, що є незалежними від конкретної платформи завдяки концепції віртуальної машини Java (*Java Virtual Machine*, JVM), яка дозволяє запускати байт-код на будь-якому пристрої, де встановлено JVM.

Java відома своєю стабільністю, безпекою та масштабованістю, завдяки чому широко використовується у веб-розробці, корпоративних системах, мобільних додатках (особливо для Android), хмарних рішеннях та великомасштабних серверних застосунках.

Ключовими характеристиками Java є: принцип "один раз написано — скрізь виконується" (*write once, run anywhere*), автоматичне управління пам'яттю (через збирач сміття), сильна типізація та підтримка багатопотоковості.

Рисунок 3.6 – Відповідь на питання з автоматичною корекцією

На рисунку 3.7 наведено результат відповіді програми на те саме запитання при умові, що користувач сам обирає параметри за якийми буде свормована відповідь на запитання. Серед параметрів були обрані наступні:

Стиль відповіді – художній;

Кількість слів - 100;

Рівень точності – 10.

Java — це мова, що народилась у спробі змусити машини говорити з людьми. Вона схожа на всесвіт, де кожен об'єкт живе за своїми законами, але слухається однієї невидимої логіки. Її рядки коду — як ноти, що оживляють програми, здатні працювати на будь-якому пристрої, де б той не був. Java — це інструмент інженера й чарівника водночас: суха й точна, але з душею, що мріє про порядок у цифровому хаосі. Вона витривала, універсальна, мов міст, який з'єднує ідею з реальністю, перетворюючи думки на дії, а логіку — на живий рух.

Рисунок 3.7 – Відповідь на питання з ручною корекцією корекцією

На рисунку 3.8 наведено результат відповіді програми на те саме запитання при умові, що користувач сам обирає параметри за якийми буде свормована відповідь на запитання. Для відповіді було обрано віршований стиль.

Java — мова, що в коді живе,
У світі машин вона все оживе.
Напишеш рядок — і лунає команда,
Працює скрізь, немов та пропаганда.

Об'єкти — її вірні друзі й суть,
Усе в ній по плану — нічого не буть.
Вона не про жарти, вона — про стабільність,
Про строгість, потоки й масштабну мобільність.

В браузері, серці, у хмарі, в руці —
Java керує в цифровім вінці.
І хоч не нова, та втоми не знає —
Кодом живе, і майбутнє будує, й чекає.

Рисунок 3.8 – Відповідь на питання з ручною корекцією корекцією

Використання програмних помічників з елементами штучного інтелекту значно розширює можливості користувачів. Проте для отримання максимального ефекту необхідно чітко формулювати запити, що не завжди виходить в недосвідчених користувачів.

ВИСНОВКИ

На основі здійсненого в процесі виконання індивідуального завдання створення програмного додатку оптимізації запитів до генеративних моделей на основі використання алгоритмів обробки природніх мов, засобів штучного інтелекта та аналізу функціональних можливостей програмних меседжерів можна зробити настіпні висновки:

1) Досліджено сучасні підходи та технології штучного інтелекту, на основі функціональних можливостей його використання при обробці природніх мов, що дозволило виділити основні підходи та алгоритми які використовуються в даній галізу;

2) Проведено класифікацію технологій обробки природніх мов на основі аналізу можливостей їх використання для аналізу текстів природньою мовою в реальному часі, що дозволило виділити основні етапи при обробці текстових запитів;

3) Проведено дослідження програмних помічників з елементами штучного інтелекту з метою виділення їх основних функціональних можливостей, що дозволило виділити їх основні архітектурні модулі;

4) Проаналізовано мовні моделі генератичних систем на основі дослідження параметрів їх архітектур, що дозволило виділити основні характерні слова які мають вплив на кінцевий результат роботи моделей;

5) Розроблено алгоритм формування тестових запитів для генеративних систем на основі ключових слів, що дозволило розробити узагальнену архітектуру програмного помічника для роботи з генеративною моделлю;

6) Спроектовано та здійснено тестування внутрішньої архітектури програмного помічника роботиз генеративною моделлю, що надало можливість програмно його реалізувати та порівняти його функціональні можливості з програмами-аналогами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Моцал В.Р., Протаковський Р.А.. Програмний засіб тестування генеративних моделей для виявлення об'єктів у відеопотоці. Збірник тез II Всеукраїнської науково-практичної конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі»., Тернопіль, 20 травня 2025 р. с. 125-126.

2. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Бакалавр” спеціальності 123 «Комп’ютерна інженерія» галузі знань 12 Інформаційні технології / О.М. Березький, Л.О.Дубчак, Г.М. Мельник, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2020. 60с.

3. Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп’ютерних систем”/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

4. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

5. Stokel-Walker C. Chat Generative Pre-trained Transformer listed as author on research papers: many scientists disapprove. Nature. 2023;613(7945):620-621.

6. Open artificial intelligence. Chat Generative Pre-trained Transformer general FAQ. 2023. Accessed February 10, 2023.

7. Gao C, Howard F, Markov N, et al. Comparing scientific abstracts generated by Chat Generative Pre-trained Transformer to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers. 2022.

8. Thorp HH. Chat Generative Pre-trained Transformer is fun, but not an author. Science. 2023;379(6630):313.

9. Communications of the ACM. ICML bans papers written by Chat Generative Pre-trained Transformer and artificial intelligence language tools. 2023. Accessed February 10, 2023.

10. Man Dis E, Bollen J, Zuidema W, van Rooij R, Bockting C. Chat-Generative Pre-trained Transformer: five priorities for research. Nature. 2023;614(7947):224-226.

11. Stokel-Walker C, VanNoorden R. What Chat Generative Pre-trained Transformer and generative artificial intelligence mean for science. Nature. 2023;614(7947):214-216.

12. Baidoo-Anu, D., & Ansah, L. O. Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. Journal of AI, 2023. 7(1), 52-62.

13. Braun, V., & Clarke, V. Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 2006. 77–101.

14. Chen, L. Chen, P., & Lin, Z. Artificial intelligence in education: A review. Ieee Access, 8, 2020.75264– 75278. Dağdeler,

15. K. O., & Demiröz, H. EFL instructors' perceptions of utilizing mobile-assisted language learning in Higher Education. Acta Educationis Generalis, 12(2), 2022.22–40.

16. Haluza, D., & Jungwirth, D. Artificial Intelligence and Ten Societal Megatrends: An Exploratory Study Using GPT-3. Systems, 11(3), 2023.120.

17. Hill-Yardin, E. L., Hutchinson, M. R., Laycock, R., and Spencer, S. A Chat(GPT) about the future of scientific publishing. Brain Behav. Immunity 110, 2023. 152–154. doi: 10.1016/j.bbi.2023.02.022

18. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A. "Language models are few-shot learners". Adv. Neural Inf. Process. Syst. 2020, 33, pp. 1877–1901.

19. Jin, L.; Tan, F.; Jiang, S.; Köker, R. "Generative Adversarial Network Technologies and Applications in Computer Vision". Intell. Neurosci. 2020, 2020, 1459107.

20. Jabbar, A.; Li, X.; Omar, B. "A survey on generative adversarial networks: Variants, applications, and training". ACM Comput. Surv. (CSUR) 2021, 54, 157.Kikalishvili, S. Unlocking the potential of GPT-3 in education: opportunities, limitations, and recommendations for effective integration. Interactive Learning Environments, 2023. 1–13.