Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Системотехніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
____Розробка та дослідження інтелектуальної системи семантичного_____
пошуку програмних продуктів за відкритими даними_____
(тема)

Виконав:
здобувач ___2__ року навчання,
групи _ІТПм-24-1_____
_____Артем Семенов_____
(власне ім'я, прізвище)

Спеціальність __122 Комп'ютерні науки____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Інформаційні технології проектування_
(повна назва освітньої програми)

Керівник _проф. каф. СТ. Вадим Саваневич___
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _____ __Ігор ГРЕБЕННІК_
(підпис) (власне ім'я, прізвище)

2025 р.

Я, як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

15.12.2025  *Семенов А. Г.*

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено 15 грудня 2025

Керівник кваліфікаційної роботи  *проф. Саваневич В.Є.*

Харківський національний університет радіоелектроніки

Факультет__Комп'ютерних наук_____

Кафедра__Системотехніки_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_____Інформаційні технології проєктування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Семенову Артему Георгійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи __Розробка та дослідження інтелектуальної системи семантичного пошуку програмних продуктів за відкритими даними
затверджена наказом університету від _24_листопада__2025 р. № 1058Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії _20_грудня_ 2025 р.
3. Вихідні дані до роботи: _текстові описи програмного забезпечення, метадані продуктів, анотації, приклади існуючих IT-систем, матеріали щодо сучасних методів семантичного пошуку на основі трансформерів, приклади відкритого програмного забезпечення, інструменти для створення моделей глибинного навчання Python, TensorFlow, PyTorch, OpenCV, метрики оцінювання точність, повнота, precision, F1-міра, інструменти інтеграції мовних моделей через API_____
_____
4. Перелік питань, що потрібно опрацювати в роботі:__4.1 Аналіз предметної області. 4.2 Методи класифікації та оцінки релевантності ПЗ. 4.3 Алгоритми семантичного пошуку. 4.4 Огляд існуючих інформаційних систем. 4.5 Порівняння IT-систем. 4.6 Глибинні нейронні мережі для аналізу тексту. 4.7 Оптимізація та оцінка моделей. 4.8 інтеграція великої мовної моделі. 4.9 Розробка гібридної моделі. 4.10 Вибір і опис вхідних даних. 4.11 Інструменти реалізації моделі. 4.12 Експериментальне дослідження.
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій:__5.1 Об'єкт, предмет та мета дослідження. 5.2 Актуальність дослідження. 5.3 Проблематика та обмеження традиційних підходів. 5.4 Можливості LLM та AI у семантичному пошуку ПЗ. 5.5 Постановка задачі дослідження. 5.6 Принцип роботи трансформера. 5.7 Підхід до навчання, вибір моделей. 5.8 Функціональні вимоги до системи. 5.9 Алгоритм роботи та архітектура. 5.10 Взаємодія користувача та системи. 5.11

Архітектура підсистеми інтелектуальної обробки запитів. 5.12 Метрики оцінки LLM. 5.13 Метрики оцінки LLM, точність та повнота. 5.14 Метрики оцінки LLM, MRR та nDCG. 5.15 Аналіз експерименту, точність моделей. 5.16 Аналіз експерименту, швидкодія моделей. 5.17 Аналіз експерименту, продуктивність моделей. 5.18 Аналіз експерименту, якість семантичного пошуку. 5.19 Методи оптимізації продуктивності. 5.20 Технології розробки застосунку. 5.21 Логіка роботи застосунку. 5.22 Розроблений застосунок, головна сторінка. 5.23 Розроблений застосунок, сторінка користувача. 5.24 Висновки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1.	*Отримання завдання на виконання кваліфікаційної роботи*	*24.11.2025*	*Виконано*
2.	*Аналіз завдання та предметної області*	*25.11.2025 – 26.11.2025*	*Виконано*
3.	*Опрацювання літератури та аналіз об'єкту дослідження*	*27.11.2025 – 29.11.2025*	*Виконано*
4.	*Розробка моделі семантичного пошуку програмного забезпечення*	*30.11.2025 – 03.12.2025*	*Виконано*
5.	*Розробка та тестування нейронної мережі для аналізу текстових описів ПЗ*	*04.12.2025 – 06.12.2025*	*Виконано*
6.	*Інтеграція моделей у гібридну систему та її навчання*	*07.12.2025 – 09.12.2025*	*Виконано*
7.	*Проведення експериментального дослідження роботи великої мовної моделі*	*10.12.2025 – 11.12.2025*	*Виконано*
8.	*Аналіз результатів, формулювання висновків та рекомендацій*	*12.12.2025 – 13.12.2025*	*Виконано*
9.	*Оформлення пояснювальної записки та презентаційних матеріалів комп'ютерного захисту*	*14.12.2025 – 16.12.2025*	*Виконано*
10.	*Представлення роботи на рецензування*	*17.12.2025*	*Виконано*

Дата видачі завдання 24 листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. каф. СТ Вадим САВАНЕВИЧ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 100 с., 14 рис., 1 табл., 2 дод., 47 джерела.

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ПОШУКУ, ВІДКРИТЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, AI-ПІДТРИМКА, СЕМАНТИЧНИЙ ПОШУК, ВЕЛИКІ МОВНІ МОДЕЛІ, КЛАСИФІКАЦІЯ ПРОДУКТІВ, АВТОМАТИЗАЦІЯ ПОШУКОВИХ ПРОЦЕСІВ

Об’єкт дослідження – процес автоматизованого пошуку, класифікації та аналізу відкритих програмних продуктів за допомогою інтелектуальної системи.

Предмет дослідження – методи та алгоритми семантичного пошуку, класифікації програмних продуктів, оцінки їх релевантності та рекомендацій для користувачів із використанням великих мовних моделей та AI-технологій.

Мета дослідження – підвищення ефективності пошуку та вибору відкритих програмних продуктів шляхом розробки інтелектуальної системи семантичного пошуку, яка інтегрує AI-модулі для аналізу контенту, класифікації за функціональністю та контекстом, автоматичної генерації рекомендацій та оптимізації процесу доступу до релевантних рішень.

Методи дослідження – системний та структурний аналіз предметної області; моделювання процесів інтелектуального пошуку; застосування алгоритмів машинного та глибинного навчання для семантичного аналізу та класифікації програмних продуктів; алгоритми рекомендацій та ранжування результатів; інтеграція великих мовних моделей для обробки природної мови.

Наукова новизна роботи полягає у створенні комплексної інтелектуальної системи семантичного пошуку для відкритого програмного забезпечення, яка поєднує методи автоматичної класифікації продуктів, семантичного аналізу запитів користувачів та генерації рекомендацій на основі AI.

ABSTRACT

Explanatory Note to the Qualification Thesis: 100 pages, 14 figures, 1 tables, 2 appendices, 47 references.

INTELLIGENT SEARCH SYSTEMS, OPEN-SOURCE SOFTWARE, AI SUPPORT, SEMANTIC SEARCH, LARGE LANGUAGE MODELS, PRODUCT CLASSIFICATION, AUTOMATION OF SEARCH PROCESSES

Object of the research – the process of automated search, classification, and analysis of open-source software products using an intelligent system.

Subject of the research – methods and algorithms for semantic search, classification of software products, assessment of their relevance, and generation of user-oriented recommendations using large language models and AI technologies.

Purpose of the research – to increase the efficiency of search and selection of open-source software products by developing an intelligent semantic search system that integrates AI modules for content analysis, functionality- and context-based classification, automatic recommendation generation, and optimization of access to relevant solutions.

Research methods – systemic and structural analysis of the domain; modelling of intelligent search processes; application of machine learning and deep learning algorithms for semantic analysis and classification of software products; recommendation and ranking algorithms; integration of large language models for natural language processing.

Scientific novelty of the work lies in the development of a comprehensive intelligent semantic search system for open-source software, which combines methods of automatic product classification, semantic analysis of user queries, and AI-based recommendation generation.

ЗМІСТ

ВСТУП

Актуальність даного дослідження зумовлена стрімким і постійно зростаючим обсягом відкритого програмного забезпечення, що суттєво ускладнює процес швидкого, обґрунтованого та якісного вибору інструментів, які дійсно відповідають конкретним потребам користувачів, організацій або команд розробників. Сучасні репозиторії відкритого ПЗ містять сотні тисяч проєктів, які відрізняються за призначенням, рівнем зрілості, якістю реалізації, активністю спільноти та використаними технологіями. Традиційні засоби пошуку та навігації в таких середовищах функціонують переважно на основі ключових слів, тегів або заздалегідь визначених статичних категорій, що істотно обмежує їхню здатність глибоко розуміти зміст програмного продукту, коректно інтерпретувати його функціональне призначення, архітектурні та технологічні особливості, а також оцінювати реальну практичну користь у конкретному контексті застосування. У результаті користувачі часто стикаються з великою кількістю нерелевантних або малопридатних результатів, змушені витрачати значні часові та інтелектуальні ресурси на ручний аналіз документації, читання вихідного коду, порівняння аналогів і пошук оптимальних рішень, що негативно впливає на ефективність прийняття рішень і загальну продуктивність роботи.

У цьому контексті великі мовні моделі та сучасні AI-технології відкривають принципово нові можливості для автоматизації процесів аналізу, порівняння та інтелектуальної класифікації програмних продуктів. LLM здатні обробляти значні обсяги різнорідної текстової інформації, включно з описами проєктів, технічною документацією, коментарями розробників і відгуками користувачів, виявляти приховані семантичні зв’язки між характеристиками програмного забезпечення, інтерпретувати технічні описи різного рівня деталізації та узгоджувати отримані знання з вимогами й очікуваннями користувачів [1]. Завдяки таким можливостям з’являється перспектива створення інтегрованої інтелектуальної платформи, яка не обмежується функціями пошуку, а виконує глибокий змістовний аналіз програмних

продуктів, оцінює їхню релевантність у заданому контексті, автоматично класифікує за функціональними та технологічними ознаками, а також формує персоналізовані рекомендації з урахуванням профілю користувача, його попереднього досвіду та конкретних завдань. Такий підхід сприяє підвищенню якості вибору відкритого ПЗ, зменшенню часових витрат і більш ефективному використанню сучасних програмних ресурсів.

Предмет дослідження – методи та алгоритми семантичного пошуку, класифікації програмних продуктів, оцінки їх релевантності та генерації рекомендацій користувачам із використанням великих мовних моделей та AI-технологій.

Мета дослідження – розробити інтегровану AI-систему для автоматизації процесів пошуку, аналізу та класифікації відкритих програмних продуктів, яка забезпечує підтримку користувача через інтелектуальні алгоритми семантичного пошуку, рекомендацій та ранжування результатів.

Практичне значення роботи полягає у створенні прототипу програмного комплексу, який можна застосовувати для підвищення ефективності пошуку та вибору відкритого ПЗ. Система дозволяє автоматично аналізувати зміст програмних продуктів, визначати їх функціональні категорії, порівнювати отримані характеристики з потребами користувача та генерувати релевантні рекомендації, що сприяє підвищенню продуктивності і якості роботи в умовах великого обсягу доступних даних, а також зменшує часові витрати на прийняття обґрунтованих рішень щодо використання програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні тенденції розвитку пошукових систем для програмного забезпечення

1.1.1 Ринок відкритого програмного забезпечення та роль інтелектуальних систем пошуку

Ринок відкритого програмного забезпечення (Open Source Software, OSS) в Україні за останні роки демонструє стабільне та динамічне зростання. Це обумовлено як глобальними тенденціями у сфері цифрових технологій, так і локальними економічними та організаційними факторами. В умовах швидкого розвитку IT-інфраструктури та збільшення числа цифрових сервісів підприємства та розробники стикаються з необхідністю ефективного пошуку, оцінки та інтеграції відкритих програмних продуктів. Одночасно, зростає кількість проєктів із відкритим кодом, доступних на різних репозиторіях, що ускладнює традиційний підхід до пошуку та вибору релевантного ПЗ.

Інтелектуальні системи семантичного пошуку виконують критично важливу роль у цьому середовищі. На відміну від класичних пошукових систем, які використовують переважно ключові слова для пошуку, інтелектуальні системи аналізують семантичний зміст запитів і контенту програмного забезпечення [2]. Це дозволяє не лише знайти релевантні продукти за функціональністю, а й оцінити їхню придатність для конкретних завдань та контекстів використання. Застосування методів обробки природної мови (NLP), алгоритмів машинного навчання та великих мовних моделей (LLM) дає змогу здійснювати контекстний аналіз, ранжування результатів та генерацію рекомендацій щодо сумісності та актуальності продуктів.

Аналітичні дослідження ринку OSS в Україні показують, що середньорічний темп зростання цього сегменту становить близько 25–30%. Основними факторами, що сприяють цьому зростанню, є: збільшення числа

цифрових стартапів, розширення використання відкритих рішень у великих та середніх підприємствах, підвищення ролі автоматизації бізнес-процесів та зростання потреби у масштабованих ІТ-рішеннях. Крім того, використання відкритого ПЗ дозволяє знижувати витрати на ліцензійне програмне забезпечення, прискорювати впровадження інноваційних рішень та сприяти розвитку ІТ-спільнот.

Інтелектуальні системи семантичного пошуку у цьому контексті виконують комплекс функцій, які підвищують ефективність використання відкритого ПЗ. До них належать: автоматичне індексування та класифікація великих обсягів репозиторіїв відкритого ПЗ, аналіз функціональних характеристик продуктів, адаптивне ранжування результатів пошуку з урахуванням контексту користувацького запиту, а також надання рекомендацій щодо ліцензійних обмежень і сумісності продуктів [3]. Завдяки цьому користувачі можуть більш швидко та точно знаходити необхідні рішення, зменшуючи час та ресурси на ручний аналіз.

Разом із потенційними перевагами, ринок відкритого програмного забезпечення та інтелектуальних систем семантичного пошуку в Україні стикається з низкою значущих викликів. Перш за все, обмежена доступність високоякісних лінгвістичних ресурсів для української мови створює серйозні перепони для побудови ефективних моделей обробки природної мови, що негативно впливає на точність семантичного аналізу і здатність систем правильно інтерпретувати запити користувачів. По-друге, структура та формат даних у відкритих репозиторіях демонструють значну різноманітність, що потребує від систем пошуку високої адаптивності, здатності інтегрувати різнорідні джерела інформації та забезпечувати узгодженість результатів незалежно від специфіки представлених даних. По-третє, критично важливим є підтримання балансу між автоматизованою обробкою інформації та можливістю користувача отримувати персоналізовані результати, які враховують унікальні вимоги конкретного проекту, організації або домену, що забезпечує

ефективність прийняття рішень і підвищує цінність системи для кінцевого користувача.

Розвиток технологій штучного інтелекту та NLP створює сприятливі умови для подальшого розвитку ринку інтелектуальних систем семантичного пошуку в Україні. Підвищення точності семантичного аналізу, зростання обсягів структурованих відкритих даних та вдосконалення алгоритмів машинного навчання дозволяють очікувати, що такі системи забезпечать швидкий і ефективний доступ до програмних продуктів, підвищуючи продуктивність користувачів і сприяючи прискоренню впровадження інновацій у національному IT-сегменті. Крім того, інтеграція адаптивних моделей, здатних враховувати динаміку запитів і змінювані потреби користувачів, відкриває перспективи для створення інтелектуальних платформ, які підтримують комплексний аналіз, класифікацію та рекомендації, формуючи основу для розвитку конкурентоспроможного українського програм.

1.1.2 Приклади успішних реалізацій систем семантичного пошуку

Інтелектуальні системи семантичного пошуку застосовуються у різних галузях IT та бізнесу для підвищення ефективності доступу до інформації та автоматизації процесів пошуку. У контексті відкритого програмного забезпечення такі системи дозволяють користувачам швидко знаходити релевантні програмні продукти, оцінювати їх функціональні можливості та сумісність, а також отримувати рекомендації щодо оптимального вибору для конкретних завдань. У науковій та практичній літературі можна виділити кілька ключових прикладів успішних реалізацій семантичних систем, що демонструють різні підходи та технології.

GitHub Copilot, розроблений у партнерстві з OpenAI, є прикладом системи, яка інтегрує великі мовні моделі для контекстного розуміння коду та автоматичної генерації рекомендацій щодо реалізації функцій [4]. Приклад інтерфейсу асистента наведено на рисунку 1.1.

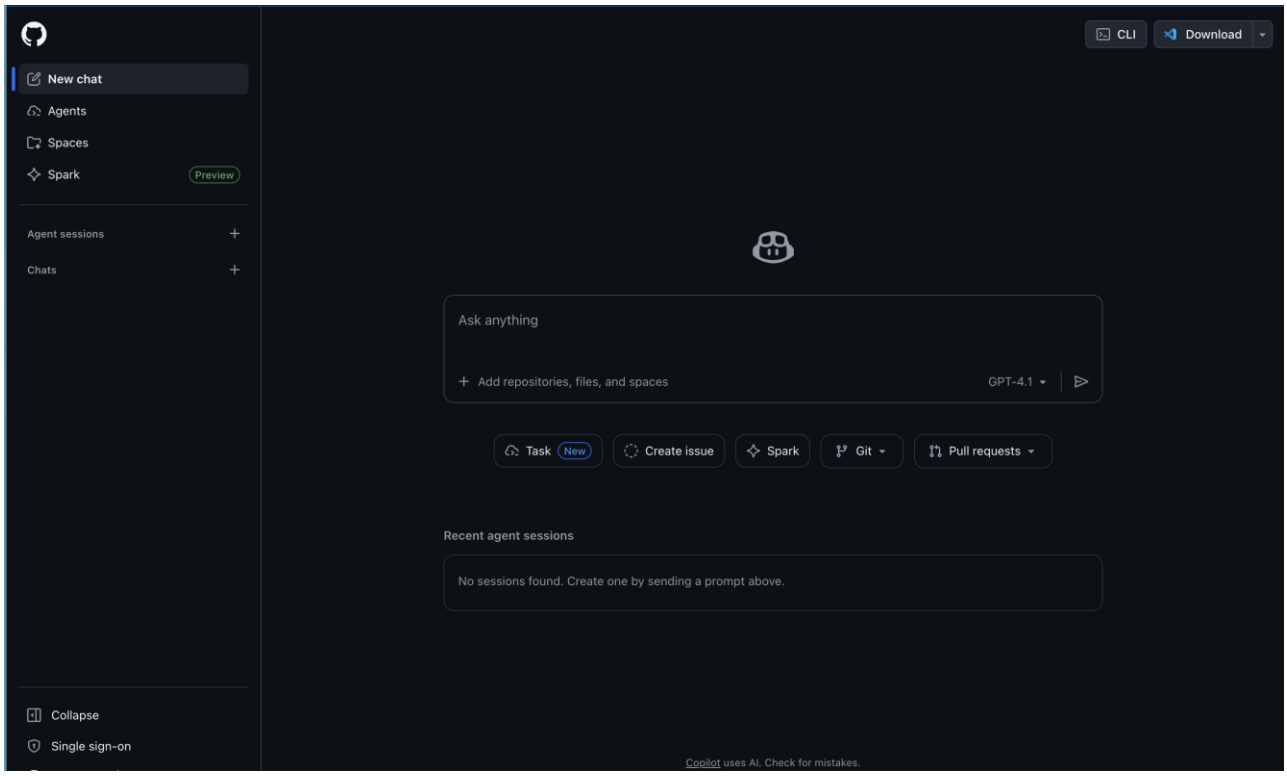

Рисунок 1.1 – Інтерфейс інтелектуального асистенту GitHub Copilot

Copilot аналізує контекст проєкту та запити користувача, пропонуючи синтаксично та логічно коректні фрагменти коду. Система використовує семантичний аналіз запитів та коду, що дозволяє розробникам швидко знаходити потрібні функціональні рішення, скорочуючи час розробки і знижуючи ризики помилок. Цей приклад ілюструє, як семантичний пошук у поєднанні з AI-моделями підвищує ефективність процесів у сфері програмної інженерії.

Open Hub – це аналітична платформа, яка здійснює збір, класифікацію та аналіз відкритих проєктів програмного забезпечення [5]. Головна сторінка та структура представлення даних платформи зображена на рисунку 1.2.

На рисунку представлено інтерфейс результатів пошуку платформи Open Hub, де для кожного програмного продукту генерується детальна аналітична картка. У ній відображаються ключові метрики активності, такі як загальна кількість рядків коду, число поточних контриб’юторів та час з моменту останнього оновлення, що дозволяє миттєво оцінити рівень підтримки та життєздатність проєкту. Крім того, для кожного рішення зазначено технічні характеристики, зокрема домінуючу мову програмування та тип ліцензії, а в

нижній частині картки розміщено набір тематичних тегів, які використовуються для точної класифікації та фільтрації програмного забезпечення за функціональними ознаками.

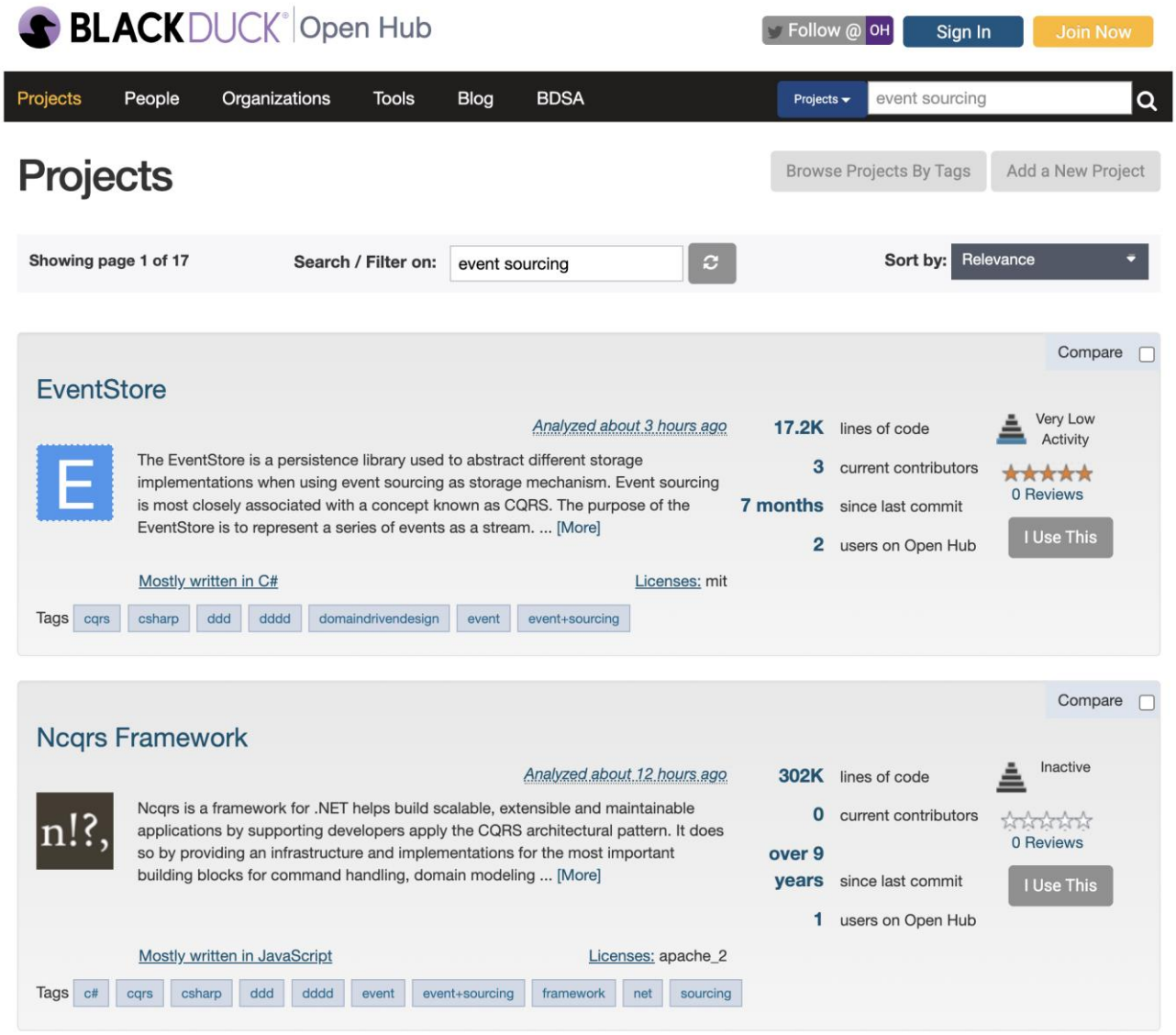

Рисунок 1.2 – Сторінка пошуку та аналітичні метрики платформи Open Hub

Система використовує алгоритми семантичного пошуку для індексування репозиторіїв, аналізу функціональних характеристик проектів та побудови рейтингових таблиць на основі активності, популярності та ліцензійних обмежень. Користувачі можуть виконувати запити з урахуванням технологій, мови програмування, функціональних модулів або специфічних ключових слів, а система семантично зіставляє запити з описами проектів, забезпечуючи високий рівень релевантності результатів.

Хоча Semantic Scholar орієнтований на академічні публікації, його підхід демонструє універсальні принципи побудови семантичних систем пошуку (рис 1.3).

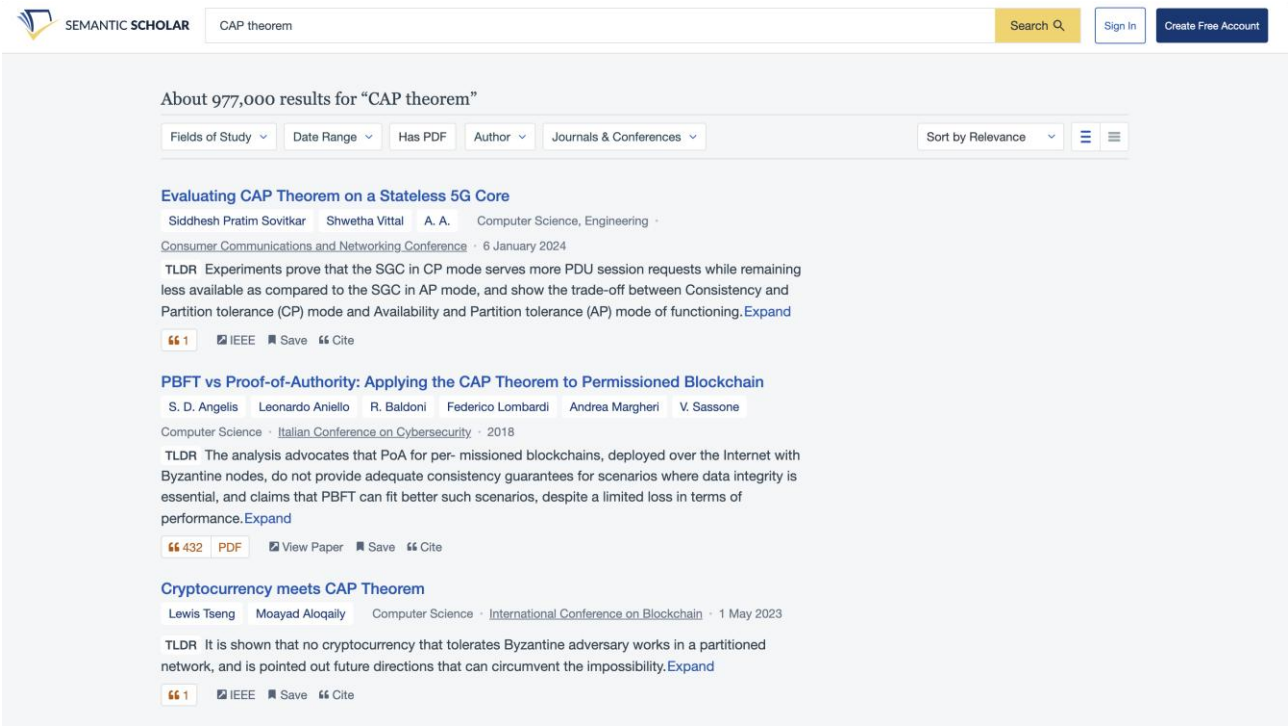

Рисунок 1.3 – Інтерфейс семантичного пошуку в системі Semantic Scholar

На рисунку зображено інтерфейс Semantic Scholar, де ключовою особливістю є блоки «TLDR» – автоматично згенеровані штучним інтелектом стислі анотації, що дозволяють миттєво зрозуміти суть публікації. Також показано панель розумних фільтрів та структуровані метадані (цитування, посилання на PDF), що демонструє ефективність семантичного аналізу для швидкого відбору релевантних наукових джерел.

Система використовує алгоритми NLP та глибокого навчання для аналізу текстового змісту, виявлення ключових концепцій та контекстних зв'язків між документами [6]. Подібні підходи можуть бути адаптовані для інтелектуальних систем пошуку програмного забезпечення, де необхідно обробляти технічну документацію, опис функціональності та метадані проектів.

Деякі великі корпорації та платформні рішення активно впроваджують власні інтелектуальні пошукові системи для відкритих репозиторіїв, спрямовані

на підвищення ефективності пошуку та аналізу проектів. Наприклад, SourceForge і GitLab реалізують пошук із семантичним ранжуванням проектів на основі поєднання ключових слів, тегів, історії комітів, активності розробників та популярності проектів (рис. 1.4).

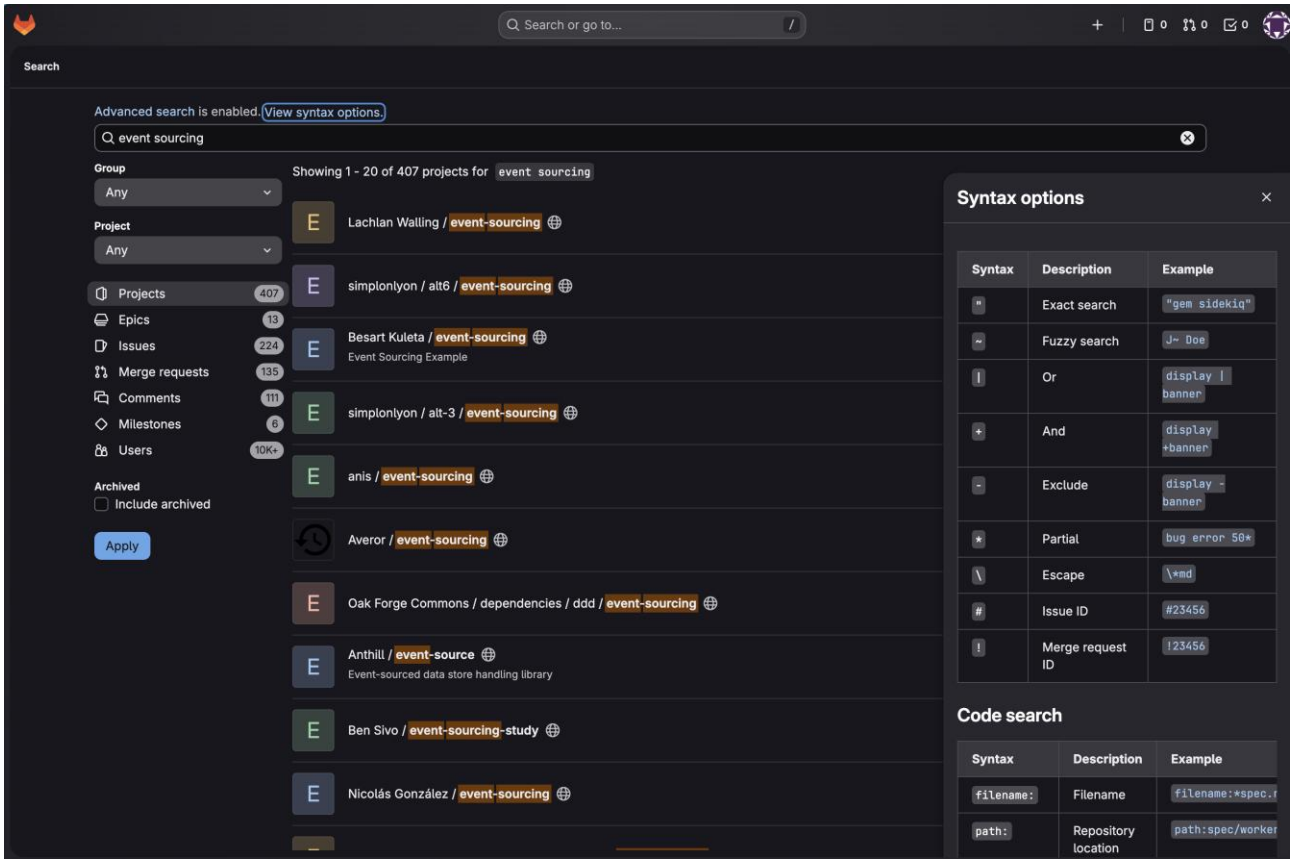

Рисунок 1.4 – Приклад результатів пошуку та фільтрації проєктів у GitLab

На рисунку зображено інтерфейс розширеного пошуку в середовищі GitLab, де результати за ключовими словами структуруються за типами об’єктів, такими як проєкти, завдання та епіки. Центральним елементом є відкрита панель «Syntax options», яка демонструє перелік спеціальних операторів (логічні AND/OR, виключення, нечіткий пошук), необхідних для точного налаштування видачі. Це наочно ілюструє бар’єр традиційного підходу: на відміну від інтуїтивного семантичного пошуку природною мовою, такий метод вимагає від користувача вивчення складного синтаксису та ручного конструювання запитів для отримання точних результатів.

Незважаючи на цю складність інтерфейсу, подібні системи представляють собою значний крок уперед, оскільки інтегрують класичні методи інформаційного пошуку з елементами семантичного аналізу метаданих. Це дозволяє користувачам знаходити релевантні продукти серед десятків тисяч проєктів, враховуючи не лише точне збіг ключових слів, а й контекстуальні та функціональні характеристики [7].

Сучасні реалізації семантичного пошуку активно інтегрують великі мовні моделі, такі як GPT і Codex, що дозволяє системам здійснювати більш глибоке розуміння контексту запитів користувачів, генерувати рекомендації та здійснювати класифікацію результатів за функціональними ознаками. Завдяки цьому алгоритми здатні оцінювати семантичну близькість концепцій, навіть якщо запит не збігається точно з назвами або описами проектів. Наприклад, запит «інструменти для обробки зображень Python» може коректно зіставлятися з проектами, які не містять усіх ключових слів у назві, але відповідають функціонально поставленим критеріям. Це суттєво підвищує точність пошуку, зменшує кількість нерелевантних результатів і дозволяє користувачеві швидко отримувати інформацію, необхідну для прийняття рішень щодо вибору або інтеграції програмного забезпечення.

Усі наведені приклади демонструють, що застосування семантичних систем пошуку істотно підвищує ефективність користувачів у роботі з відкритим ПЗ, скорочує час доступу до релевантної інформації та покращує якість автоматично генерованих рекомендацій [8]. Крім того, інтеграція алгоритмів глибинного навчання та великих мовних моделей забезпечує високу адаптивність системи до різних типів запитів, підвищує точність класифікації проектів та відкриває можливості для автоматизації аналітичних процесів у сфері ІТ. Використання таких підходів дозволяє оцінювати проекти не лише за назвою чи ключовими словами, а й за функціональністю, ліцензійними характеристиками, технологічними параметрами та взаємозв’язками між компонентами, формуючи комплексне семантичне представлення ресурсів.

Розглянуті реалізації підтверджують, що поєднання семантичного пошуку з алгоритмами штучного інтелекту та великими мовними моделями є ефективним інструментом управління великими масивами даних відкритого програмного забезпечення. Такі системи не лише забезпечують швидкий доступ до релевантних продуктів, а й створюють умови для їх детальної оцінки, класифікації та рекомендаційного аналізу.

1.2 Загальні концепції інтелектуальних пошукових систем

1.2.1 Класифікація систем пошуку та їх особливості

Сучасні інформаційні екосистеми характеризуються величезними обсягами даних, що зберігаються у різноманітних джерелах – від наукових публікацій та документації до відкритих репозиторіїв програмного забезпечення. Ефективний доступ до цієї інформації неможливий без застосування систем пошуку, здатних не лише швидко знаходити дані, а й оцінювати їхню релевантність та відповідність контексту запиту користувача. Розвиток технологій обробки природної мови, семантичного аналізу та машинного навчання призвів до появи інтелектуальних систем, які суттєво перевищують традиційні підходи за точністю та адаптивністю.

Системи пошуку інформації можна умовно класифікувати на чотири основні категорії [9]:

– системи пошуку за ключовими словами;

– булеві та логічні системи пошуку;

– системи інформаційного пошуку з ранжуванням;

– семантичні та інтелектуальні системи пошуку.

Класичні системи пошуку за ключовими словами забезпечують прямий доступ до документів або об’єктів, які містять задані терміни. Вони функціонують на основі порівняння тексту запиту з текстом документа і повертають результати за точним збігом слів. Такий підхід є простим та

ефективним для невеликих баз даних або специфічних завдань, проте його точність значно знижується у випадках, коли необхідно враховувати контекст або синонімічні зв’язки між поняттями. Цей тип систем не здатний розпізнавати омоніми та багатозначні слова, що обмежує його застосування у складних інформаційних екосистемах.

Булеві та логічні системи пошуку дають можливість комбінувати ключові слова за допомогою операторів AND, OR, NOT, формуючи більш складні запити [10]. Вони підвищують гнучкість пошуку та дозволяють точніше відбирати документи відповідно до заданих умов. Водночас для ефективного використання цих систем необхідні знання логіки побудови запитів, а також чітке визначення структурних умов пошуку. Ці системи не враховують семантичні взаємозв’язки та контекст, що обмежує їхню ефективність при роботі з великими, динамічними або неоднорідними масивами даних.

Системи інформаційного пошуку з ранжуванням застосовують алгоритми оцінки релевантності, такі як TF-IDF, BM25 або PageRank, що дозволяють враховувати вагу термінів у документі, їхню частоту та авторитетність джерел. Це забезпечує більш точне відображення результатів та можливість обробки великих масивів інформації. Незважаючи на підвищену точність порівняно з класичними методами, такі системи обмежені поверхневим аналізом тексту та не здатні повною мірою враховувати смислові зв’язки між поняттями або контекст запиту [11].

Семантичні та інтелектуальні системи пошуку є сучасними рішеннями, які поєднують методи обробки природної мови, семантичного аналізу та машинного навчання для глибинного розуміння запиту користувача [12]. Вони здатні розпізнавати контекст, синоніми, омоніми, класифікувати документи за тематикою та формувати персоналізовані рекомендації. Такі системи інтегрують різнорідні джерела даних, включно з текстовими документами, метаданими та репозиторіями коду, що робить їх особливо ефективними у сфері відкритого програмного забезпечення. Вони здатні адаптуватися до динаміки запитів,

зберігати історію взаємодії користувача та підвищувати точність результатів завдяки постійному навчання на реальних даних.

Особливістю семантичних систем є здатність забезпечувати високий рівень релевантності навіть при неповному або неточному формулюванні запиту. Алгоритми семантичного пошуку враховують контекст, тематичну близькість і функціональну відповідність об’єктів пошуку, що дозволяє ефективно знаходити проєкти або документи, які не містять точних ключових слів у назві чи описі [13]. Завдяки цьому скорочується час доступу до релевантної інформації, зменшується навантаження на користувача та створюються умови для автоматизації аналітичних процесів у сфері IT.

Тип даних, з якими працюють системи, також впливає на їхню класифікацію. Одні системи оптимізовані для текстових документів, інші – для структурованих або напівструктурованих даних, включаючи метадані, таблиці та кодові репозиторії. Семантичні системи поєднують ці підходи, забезпечуючи можливість одночасного аналізу різнорідної інформації та підвищення точності оцінки релевантності.

1.2.2 Використання великих мовних моделей і штучного інтелекту для семантичного пошуку

Сучасний розвиток інформаційних технологій та значне зростання обсягів даних у відкритих репозиторіях програмного забезпечення потребує ефективних інструментів пошуку, здатних не лише ідентифікувати точні збіги ключових слів, а й аналізувати смислові, контекстуальні та функціональні взаємозв’язки між інформаційними об’єктами. У цьому контексті великі мовні моделі (Large Language Models, LLM) та алгоритми штучного інтелекту виступають ключовими компонентами інтелектуальних систем семантичного пошуку. Вони забезпечують можливість обробки природної мови на рівні, який значно перевищує традиційні підходи, дозволяючи розуміти складні запити

користувача, формувати релевантні відповіді та проводити глибоку класифікацію об’єктів пошуку [14].

Великі мовні моделі, які базуються на архітектурі трансформерів, реалізують механізми багаторівневого аналізу тексту. Трансформери забезпечують здатність одночасно враховувати глобальні й локальні контексти, використовуючи багатоголову систему уваги (multi-head attention), що дозволяє моделі фокусуватися на різних частинах запиту або документа для визначення релевантності. Це дає змогу LLM не просто співставляти ключові слова, а й розпізнавати семантичні зв’язки між термінами, аналізувати синоніми, омоніми, контекстуальні залежності та багаторівневі логічні взаємозв’язки. Такі можливості є критично важливими для роботи з відкритим програмним забезпеченням, де описи проектів, метадані та документація можуть містити неоднорідні, технічно складні формулювання.

Інтеграція великих мовних моделей у семантичні системи пошуку здійснюється за допомогою кількох взаємопов’язаних напрямів, кожен із яких підсилює функціональність і точність системи. Перший напрямок – контекстний аналіз запиту користувача, який реалізується через здатність LLM обробляти запити природною мовою, навіть коли вони сформульовані неповно, нечітко або містять граматичні й стилістичні неточності. Моделі інтерпретують смислову структуру запиту, виявляють приховану семантичну близькість між словами та концептами, що дозволяє точно зіставляти запит із релевантними документами та проєктами відкритого програмного забезпечення [15]. Це підвищує загальний рівень точності видачі, зменшує кількість малозначущих результатів та створює умови для більш ефективної і зручної взаємодії користувача з системою. Крім того, завдяки глибинному контекстному аналізу система здатна розпізнавати наміри користувача і пропонувати уточнюючі варіанти запитів, що істотно покращує якість семантичного пошуку.

Другий напрям інтеграції полягає у класифікації та семантичному тегуванні документів і проектів відкритого програмного забезпечення. LLM дозволяють автоматично визначати функціональне призначення продукту, його

технологічну спрямованість, сумісність із конкретними платформами, а також взаємозв’язки та залежності між компонентами. Такий підхід створює структуровану та багаторівневу інформаційну базу, що полегшує пошук і підвищує ефективність рекомендаційних функцій системи. Поєднання цього підходу з алгоритмами ранжування, які враховують активність розробників, популярність проєктів і частоту оновлень репозиторіїв, забезпечує комплексну оцінку релевантності результатів і формує цілісну систему підтримки прийняття рішень у середовищі відкритого ПЗ.

Третій напрямок – генерація рекомендацій на основі семантичного порівняння об’єктів. LLM здатні аналізувати схожі проекти, оцінювати їхню функціональність, сумісність, рівень документації та технічну якість, пропонуючи користувачеві оптимальні варіанти для використання або інтеграції. Це забезпечує активну підтримку користувача, коли система не обмежується пасивним пошуком, а виступає в ролі інтелектуального асистента, спрямовуючи користувача до найбільш придатних і ефективних рішень. Така функціональність значно підвищує продуктивність процесу прийняття рішень, особливо в умовах неповної інформації або великого обсягу доступних проектів.

Особливу увагу слід приділити інтерактивним режимам пошуку, які стають можливими завдяки використанню LLM. У таких режимах користувач формує запит природною мовою, а модель на основі глибинного семантичного аналізу контексту формує розгорнуту відповідь або підбірку релевантних ресурсів, що може включати технічні пояснення, інструкції щодо використання, інтеграційні рекомендації та додаткові аналітичні коментарі [16]. Це створює інтерфейс природної взаємодії, який наближає роботу користувача до мислення експерта, скорочує час аналізу інформації і підвищує точність вибору програмних продуктів. Такі інтерактивні механізми також дозволяють проводити уточнення запитів у реальному часі, що підвищує адаптивність системи та забезпечує більш глибоку персоналізацію процесу пошуку.

Важливою особливістю застосування LLM є здатність системи до адаптивного навчання та самостійного вдосконалення на основі історії запитів

користувача і поведінкових патернів у системі. Це дозволяє прогнозувати майбутні потреби, персоналізувати результати та покращувати точність рекомендацій у динамічних інформаційних середовищах. Крім того, поєднання LLM із традиційними алгоритмами інформаційного пошуку, такими як TF-IDF або BM25, формує гібридні системи, які враховують як статистичні характеристики даних, так і семантичні зв'язки, забезпечуючи високий рівень релевантності та стабільності роботи. Такий комбінований підхід дозволяє ефективно обробляти великі обсяги інформації, зменшувати навантаження на користувача та створювати інтегровані системи семантичного пошуку, здатні адаптуватися до специфічних доменів і потреб різних груп користувачів [17].

1.3 Основи розробки інтелектуальних пошукових систем

1.3.1 Сучасні фреймворки та технології для розробки пошукових систем

Розробка сучасних інтелектуальних систем семантичного пошуку ґрунтується на поєднанні передових методів обробки природної мови, алгоритмів класифікації та індексації даних, а також сучасних програмних технологій, які забезпечують масштабованість, гнучкість і ефективність роботи системи. Фундаментальною вимогою до таких систем є здатність не лише знаходити точні збіги ключових слів, а й розуміти контекст, смислові взаємозв'язки та функціональне призначення об'єктів інформаційного простору, що особливо актуально у сфері відкритого програмного забезпечення.

Сучасні пошукові платформи будуються на базі систем індексації, які дозволяють швидко організовувати та обробляти великі обсяги текстових і метаданих. Архітектури, що використовуються для таких систем, забезпечують ефективне зберігання документів у структурованому форматі, наприклад JSON, а також підтримують повнотекстовий пошук, фільтрацію та агрегацію даних у реальному часі. Це дозволяє реалізовувати високопродуктивні механізми пошуку, здатні обробляти тисячі запитів одночасно, що є критично важливим у

динамічному середовищі відкритого ПЗ, де обсяг даних постійно зростає, а структура і зміст репозиторіїв значно різняться.

Для забезпечення глибокого семантичного аналізу інтегруються спеціалізовані бібліотеки для обробки природної мови, що реалізують токенізацію, морфологічний і синтаксичний аналіз тексту, виявлення сутностей та тематичне моделювання. Використання таких компонентів дозволяє системі ідентифікувати значення слів у конкретному контексті, розпізнавати синоніми та омоніми, а також здійснювати багаторівневу інтерпретацію складних запитів користувачів. Ці можливості особливо важливі для семантичного пошуку, оскільки вони забезпечують не просто співпадіння символів, а розуміння змісту документа і запиту.

Великі мовні моделі відіграють у цьому процесі ключову роль, надаючи можливість обробляти природну мову на рівні, який значно перевищує можливості традиційних алгоритмів пошуку. Завдяки механізмам уваги, багатошаровим трансформерам та глибинному навчанню, LLM здатні аналізувати контекст на глобальному та локальному рівнях, оцінювати семантичну близькість термінів та формувати рекомендації щодо найбільш релевантних об’єктів [18]. Такі моделі дозволяють здійснювати як класифікацію та тегування документів, так і побудову рекомендаційних систем, які автоматично відбирають найбільш відповідні проекти або компоненти програмного забезпечення на основі аналізу їх функціональної та технологічної сумісності.

Інтеграція великих мовних моделей у системи семантичного пошуку реалізується через хмарні сервіси та API, що дозволяють виконувати обчислення без необхідності локального розгортання складних моделей, а також інтегрувати їх із традиційними системами індексації та базами даних. Це створює гнучку архітектуру, у якій LLM відповідають за семантичний аналіз і генерацію рекомендацій, тоді як індексаційні системи забезпечують швидкий доступ до великих обсягів структурованих і неструктурованих даних.

Для збереження та обробки інформації використовуються бази даних різного типу, включно з документно-орієнтованими, колонковими та графовими базами даних. Документно-орієнтовані сховища дозволяють ефективно зберігати складні структури метаданих, графові бази створюють можливість моделювання складних взаємозв’язків між об’єктами, а колонкові рішення оптимізують аналітичні запити за великими обсягами даних. У комплексі з алгоритмами семантичного ранжування та адаптивного навчання це забезпечує високий рівень точності та релевантності результатів.

Особливо важливим елементом сучасних систем є використання контейнеризації та оркестрації, що дозволяє розгортати модульну архітектуру, інтегрувати різні компоненти, масштабувати систему у відповідності до навантаження та забезпечувати безперебійну роботу навіть при обробці великих масивів даних. Контейнеризація також сприяє повторному використанню компонентів, стандартизації середовища розробки та забезпеченню стабільності при впровадженні оновлень.

1.3.2 Методи програмування та алгоритми семантичного пошуку

Розробка інтелектуальних систем семантичного пошуку ґрунтується на поєднанні класичних алгоритмів інформаційного пошуку та сучасних методів обробки природної мови з використанням штучного інтелекту. Ключовим завданням таких систем є забезпечення здатності розуміти смислові зв’язки між об’єктами інформаційного простору, оцінювати релевантність даних запитів та адаптувати результати пошуку під контекст і цілі користувача. Для цього застосовуються як програмні методи традиційного пошуку, так і алгоритми семантичного аналізу, які дозволяють автоматизувати процес інтерпретації запитів та документів.

На рівні програмування семантичні системи зазвичай реалізуються як багаторівневі архітектури, де кожен компонент відповідає за певний етап

обробки інформації [19]. Узагальнену схему такої архітектури та взаємодію її компонентів наведено на рисунку 1.5.

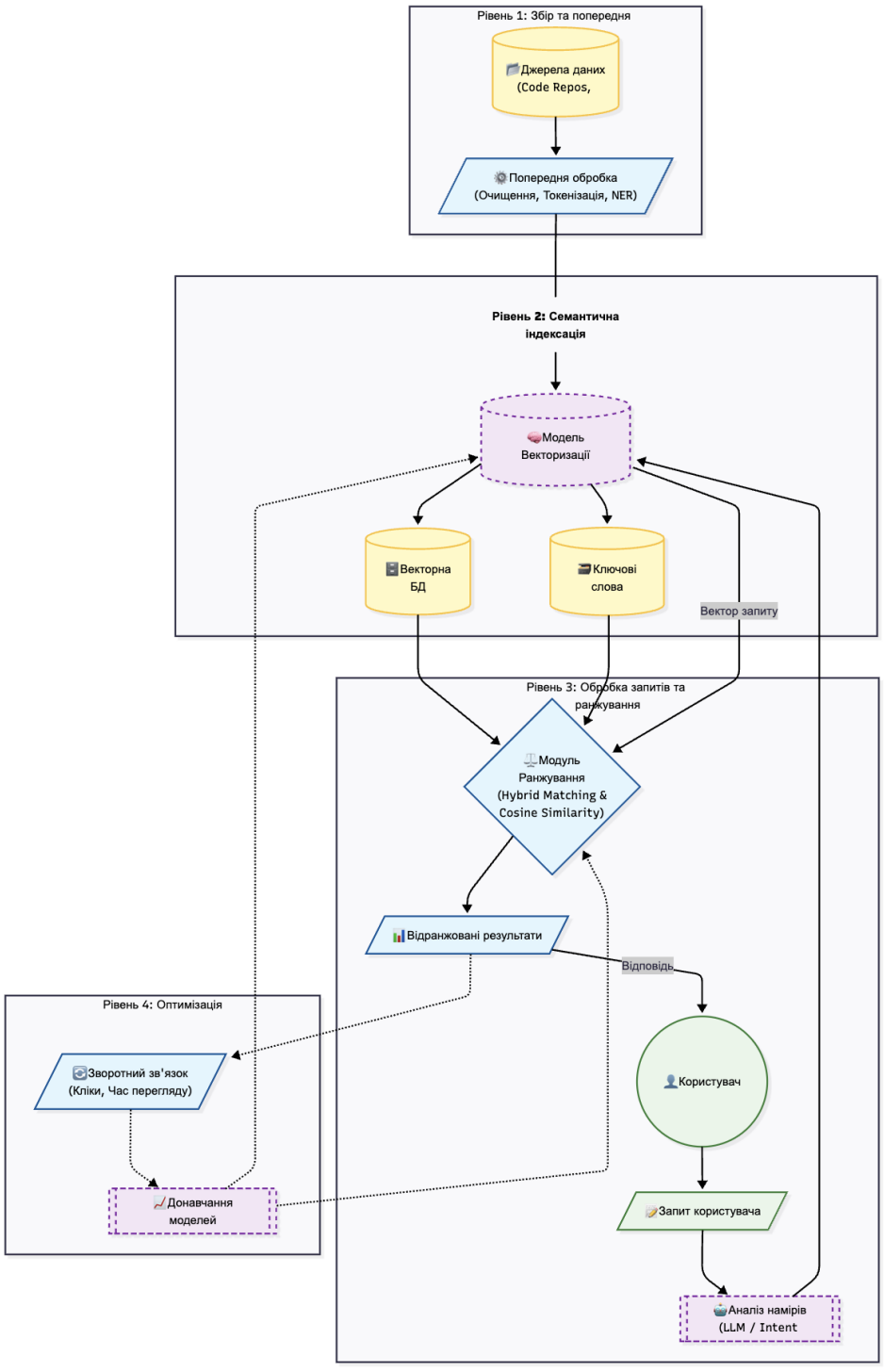

Рисунок 1.5 – Узагальнена схема багаторівневої архітектури системи семантичного пошуку

На першому рівні, що відповідає за збір та попередню обробку даних, відбувається підготовка інформаційної бази системи. Цей етап включає

агрегацію даних із різнорідних джерел (репозиторіїв коду, технічної документації, форумів) та їх глибоке очищення. Окрім стандартних процедур токенізації, нормалізації словоформ та видалення стоп-слів, критично важливим процесом на цьому етапі є інтелектуальна сегментація тексту (chunking). Оскільки сучасні моделі мають обмеження на довжину вхідного контексту, великі документи розбиваються на логічні фрагменти зі збереженням перекриття (overlap), що дозволяє не втрачати контекст на межах сегментів. Паралельно здійснюється виділення ключових сутностей (Named Entity Recognition), таких як назви бібліотек, функції або класи помилок, що забезпечує уніфіковану структуру вхідних даних – критичну передумову для ефективної індексації.

На рівні семантичної індексації система перетворює підготовлені дані у машиночитаний формат. Центральним елементом тут виступає модель векторизації, яка використовує архітектуру трансформерів (наприклад, BERT-подібні моделі) для побудови щільних векторних представлень (dense embeddings) документів і запитів. Векторизація дозволяє зіставляти об’єкти за семантичною близькістю, розпізнаючи синоніми (наприклад, «image recognition» і «OCR») та полісемантичні терміни [20]. Архітектура реалізує гібридний підхід до зберігання: векторна база даних відповідає за пошук за змістом, тоді як інвертований індекс зберігає розріджені вектори (sparse vectors) для точного пошуку за ключовими словами (BM25). Така комбінація дозволяє нівелювати недоліки нейронних мереж у пошуку специфічних абревіатур або точних назв змінних у коді.

Процес безпосередньої обробки запитів та ранжування відображено на третьому рівні. Тут відбувається взаємодія з користувачем: вхідний запит проходить через модуль аналізу намірів, який, завдяки інтеграції великих мовних моделей, здійснює логічний аналіз формулювань та виявляє приховані зв’язки між сутностями. Це дозволяє генерувати уточнюючі рекомендації або альтернативні запити природною мовою. Ключову роль відіграє алгоритм ранжування, який об’єднує результати векторного пошуку (на основі косинусної подібності) та традиційного текстового аналізу. Такий підхід дозволяє оцінювати

глибинну подібність між змістовими характеристиками документів та запитом, забезпечуючи високу релевантність видачі.

Завершальним елементом архітектури, представленим на четвертому рівні, є контур оптимізації та адаптації. Важливим компонентом побудови подібних систем є застосування алгоритмів адаптивного навчання, які дозволяють підтримувати постійне вдосконалення механізмів обробки інформації на основі зворотного зв'язку (кліків, часу перегляду). Динамічна адаптація вагових коефіцієнтів у формулі ранжування та корекція моделей ембедингів (fine-tuning) забезпечують стійкість системи до еволюції контенту та лінгвістичних змін. Крім того, можливість інкрементального оновлення індексів запобігає необхідності повної перегенерації пошукових структур, що є критично важливим для роботи з великими сховищами даних [21].

Програмна реалізація таких алгоритмів базується на модульному, компонентно орієнтованому підході, що забезпечує високу масштабованість і можливість інтеграції нових моделей без порушення цілісності системи. Це створює основу для довгострокового розвитку архітектури семантичного пошуку та дозволяє адаптувати її до нових викликів, пов'язаних зі збільшенням обсягів даних, ускладненням запитів та підвищенням вимог до персоналізації пошукових результатів.

1.4 Постановка задачі дослідження

Метою дослідження є підвищення ефективності пошуку та оцінки відкритих програмних продуктів шляхом створення інтелектуальної системи семантичного пошуку, здатної забезпечувати точне зіставлення запитів, адаптивне ранжування результатів та генерацію релевантних рекомендацій з урахуванням функціональної сумісності, технологічних характеристик та контексту запитів користувачів.

Для досягнення цієї мети необхідно виконати наступні задачі:

– обробити та структурувати дані, включно з описами програмних продуктів, документацією, метаданими та коментарями до коду, для створення єдиного стандартизованого формату для семантичного аналізу;

– розробити алгоритми семантичного аналізу для оцінки контексту запитів, виявлення тематичних і функціональних зв’язків між проектами та формування векторних представлень інформації для точного зіставлення запиту з релевантними результатами;

– інтегрувати великі мовні моделі та методи штучного інтелекту для підвищення точності пошуку, генерації рекомендацій та класифікації результатів за функціональністю, технологією реалізації та сумісністю;

– реалізувати адаптивне ранжування результатів з урахуванням популярності проектів, оцінок користувачів, історії запитів та семантичної близькості до запиту;

– побудувати модульну та масштабовану архітектуру системи для інтеграції нових технологій, адаптивного навчання моделей та підтримки оновлення даних у режимі реального часу;

– забезпечити збір зворотного зв’язку від користувачів та автоматичне оновлення моделей для підвищення точності результатів і покращення користувацького досвіду.

2 ВЕЛИКІ МОВНІ МОДЕЛІ У СЕМАНТИЧНОМУ ПОШУКУ

2.1 Технологія трансформерів

2.1.1 Принцип роботи трансформерів

Трансформери стали проривною архітектурою для обробки послідовних даних, зокрема в області обробки природної мови. Впроваджені дослідниками Google у 2017 році, вони запропонували новий підхід до аналізу послідовностей, який спирається виключно на механізми уваги замість рекурентних або згорткових структур, що використовувалися в традиційних нейронних мережах [22]. Такий підхід дозволяє ефективно фіксувати довгострокові залежності та контекстну інформацію в межах текстових послідовностей, таких як речення чи абзаци.

Архітектура трансформера поділяється на два ключові блоки: кодер і декодер (рис. 2.1).

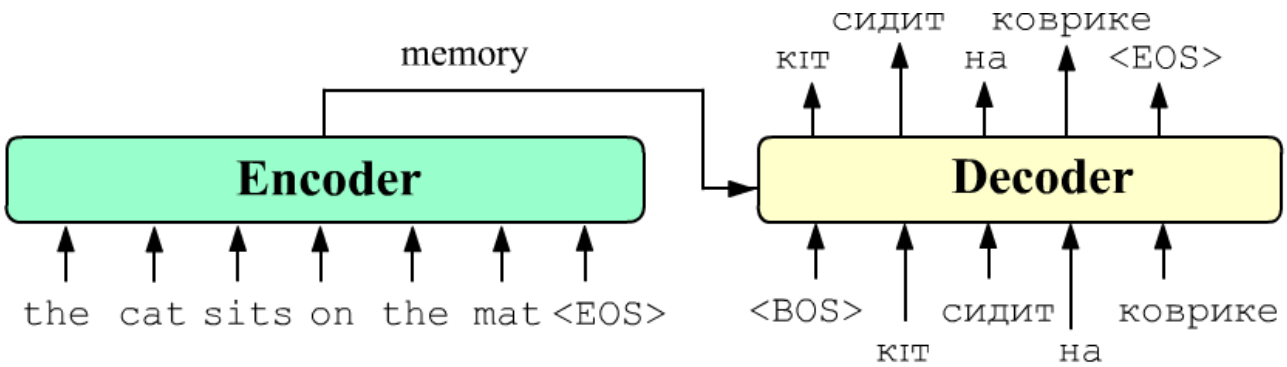

Рисунок 2.1 – Порівняння підходів до формування рекомендацій

Основою трансформерів є механізм самоуваги, що забезпечує можливість моделі зважувати значимість кожного елемента послідовності при обробці конкретного токена. Для цього використовуються вектори запиту, ключа та значення, які дозволяють обчислювати релевантність окремих елементів до поточного контексту. Ваги, отримані в результаті функції softmax,

застосовуються для формування зважених представлень елементів, що дозволяє моделі враховувати семантичні та синтаксичні зв’язки в довгих текстах. На відміну від рекурентних мереж, які обробляють дані послідовно і часто страждають від проблеми зникаючого градієнта, трансформери можуть одночасно аналізувати всі елементи послідовності, що значно підвищує ефективність навчання та продуктивність моделі при обробці великих обсягів інформації.

Архітектура трансформера ґрунтується на розділенні на кодер і декодер, кожен із яких виконує специфічну функцію в процесі обробки послідовностей даних. Кодер перетворює вхідну послідовність у багатовимірні векторні представлення, що зберігають інформацію про контекст та взаємозв’язки між елементами послідовності. Декодер, у свою чергу, використовує ці вектори для генерації вихідної послідовності, покладаючись на вже згенеровані елементи для забезпечення узгодженості та логічної цілісності результату. Основною перевагою трансформерів є паралельна обробка даних та можливість розподіленого навчання на сучасних обчислювальних платформах, таких як GPU та TPU, що значно скорочує час навчання порівняно з традиційними рекурентними або згортковими нейронними мережами [23].

Використання трансформерів стало основою для створення великих мовних моделей, таких як GPT, BERT та їхніх модифікацій, які забезпечують високоточне розуміння тексту та генерацію природної мови. В українському контексті архітектура трансформерів застосовується у різноманітних проєктах обробки природної мови. Наприклад, модель UkrBERT продемонструвала високі результати у завданнях класифікації тексту, розпізнавання іменованих сутностей, семантичного пошуку та відповіді на запитання українською мовою. Адаптація таких моделей до національної мови та культурного контексту підвищує релевантність результатів, покращує точність класифікаційних алгоритмів та сприяє більш ефективному використанню технологій великих мовних моделей у локальному IT-сегменті.

У сфері розмовного штучного інтелекту трансформери застосовуються для створення чат-ботів, віртуальних асистентів і систем семантичного пошуку. Їхня здатність враховувати контекст, проводити логічний аналіз запитів і генерувати природну мову забезпечує більш точну, адаптивну та персоналізовану взаємодію з користувачем. Водночас широке впровадження трансформерів в Україні стикається з низкою викликів, серед яких обмежена кількість високоякісних текстових корпусів українською мовою, що ускладнює ефективне навчання моделей, а також значні обчислювальні та пам’яттєві вимоги великих трансформерів, які можуть бути недоступними для малих і середніх підприємств, обмежуючи масштабування та практичне застосування технологій LLM у локальному контексті.ного ринку та інтеграції у глобальні відкриті репозиторії.

Крім технічних викликів, існують і суттєві етичні аспекти застосування трансформерів та великих мовних моделей, які набувають особливої актуальності в умовах стрімкого розширення їх використання у критично важливих інформаційних системах. Оскільки такі моделі навчаються на масштабних вибірках текстових даних, отриманих з різнорідних і не завжди контрольованих джерел, вони здатні неусвідомлено відтворювати системні упередження, культурні чи соціальні стереотипи, а також моделі дискримінаційної поведінки [24]. Це створює ризики помилкових рішень у сферах, де автоматизовані рекомендації можуть впливати на безпеку чи права користувачів. Додатковою проблемою залишається можливість ненавмисного відтворення фрагментів конфіденційних даних, що можуть міститися у тренувальних вибірках.

Незважаючи на зазначені обмеження, трансформери та великі мовні моделі демонструють значний потенціал у підвищенні ефективності сучасних автоматизованих сервісів, що пов’язано з їх здатністю до глибокого контекстного аналізу та адаптивної генерації результатів. Розвиток таких систем сприяє побудові інтелектуальних сервісів нового покоління, орієнтованих на природну комунікацію, персоналізацію взаємодії та адаптацію до індивідуальних потреб користувача. У міру того, як українська екосистема NLP

посилюється завдяки появі нових корпусів, розширенню доступних мовних ресурсів і вдосконаленню моделей, можна очікувати поступового поширення трансформерних підходів у різних галузях – від сервісів підтримки клієнтів та електронної комерції до медичних інформаційних систем, освітніх платформ і державних цифрових сервісів. Така тенденція сприятиме формуванню більш ефективних, гнучких і користувацько-орієнтованих цифрових рішень.

Трансформери суттєво трансформували сам принцип роботи з послідовними даними, запропонувавши механізм уваги як альтернативу традиційним рекурентним моделям та створивши підґрунтя для розвитку високопродуктивних мовних моделей, здатних до глибокого семантичного аналізу та генерації природної мови. Подальше розв’язання проблем, пов’язаних з доступністю якісних даних, забезпеченням достатніх обчислювальних ресурсів і чітким регулюванням етичних аспектів, відкриє можливості для повноцінної реалізації потенціалу трансформерів у широкому спектрі застосувань. Це сприятиме підвищенню інноваційності та ефективності українського інформаційного середовища, а також створить передумови для появи нових технологічних рішень, здатних забезпечити стійке та безпечне функціонування сучасних цифрових систем.

2.1.2 Особливості навчання та застосування трансформерів у пошукових системах

Навчання трансформерних моделей є обчислювально інтенсивним процесом, що вимагає значних обчислювальних потужностей і великих обсягів даних. Особливості української мови та обмежена доступність високоякісних ресурсів створюють специфічні виклики для підготовки моделей трансформерів для українських задач NLP, які потребують ретельного планування та застосування стратегічних методик.

Процес підготовки таких моделей зазвичай включає два основних етапи: попереднє навчання та доопрацювання для конкретних задач. Попереднє

навчання передбачає тренування трансформера на великому корпусі немаркованих текстів, що дозволяє моделі сформувати узагальнене розуміння мовних закономірностей, синтаксичних структур та семантичних залежностей. Цей етап є ключовим для побудови надійних моделей, здатних ефективно відображати контекстну та семантичну інформацію в текстах.

Для української мови проблема полягає у відносно невеликому обсязі та обмеженій різноманітності доступних корпусів, таких як Український національний лінгвістичний корпус та Український веб-корпус. Для подолання цієї проблеми дослідники використовують стратегії трансферного навчання.

Трансферне навчання визначається як метод, за якого знання, здобуті моделлю під час вирішення однієї задачі на великих масивах даних, переносяться та адаптуються для іншої, спорідненої цільової задачі. Цей підхід дозволяє використовувати вже сформовані параметри моделі як стартову точку, значно підвищуючи ефективність навчання в умовах дефіциту даних. Зокрема, це реалізується через адаптацію моделей, попередньо навчених на споріднених слов'янських мовах, таких як польська, для українських даних. Додатково застосовуються методи розширення корпусу, зокрема зворотний переклад та генерація синтетичних текстів, що дозволяє штучно збільшувати обсяг і різноманітність навчальних даних.

Після етапу попереднього навчання трансформер піддається тонкому налаштуванню під конкретні задачі, наприклад, класифікацію текстів, розпізнавання іменованих сутностей або генерацію тексту. Це дозволяє моделі адаптуватися до специфіки предметної області та вивчати закономірності, релевантні для поставлених завдань. У випадку української мови тонке налаштування ускладняється браком достатньо великих і якісно анотованих наборів даних, що змушує використовувати альтернативні підходи, такі як краудсорсинг анотацій, трансферне навчання між суміжними доменами та синтетичні методи генерації даних.

Обчислювальна складність та вимоги до ресурсів є ще одним значним викликом. Для навчання великих трансформерів необхідні потужні GPU або TPU

та великі обсяги оперативної пам’яті, що може бути недоступним для багатьох українських організацій, особливо малих та середніх підприємств. Для пом’якшення цих обмежень застосовуються методи стиснення моделей, розподілене навчання, використання ефективних фреймворків з відкритим кодом та спільне використання обчислювальної інфраструктури між академічними установами, дослідницькими центрами та промисловими партнерами [26].

Особливу увагу приділяють вибору алгоритмів оптимізації та налаштуванню гіперпараметрів, таких як швидкість навчання, розмір пакетів та методи регуляризації, оскільки вони істотно впливають на продуктивність та збіжність моделі. Під час виведення трансформерні моделі також вимагають оптимізованих стратегій, здатних забезпечити відповіді в реальному часі при мінімальних обчислювальних витратах, що досягається через квантування, обрізку параметрів та динамічну пакетну обробку.

Інтеграція трансформерів у виробничі системи, такі як чат-боти, віртуальні асистенти або конвеєри генерації контенту, потребує дотримання принципів модульного дизайну, масштабованості та ремонтопридатності для забезпечення безперебійної роботи та довгострокової стійкості рішень. Важливими є також заходи безпеки та захисту конфіденційних даних, включно з шифруванням, контролем доступу та безпечними конвеєрами розгортання, що відповідають законодавчим нормам щодо захисту персональних даних.

2.2 Огляд сучасних великих мовних моделей для інтелектуального пошуку

Сучасні системи інтелектуального семантичного пошуку значною мірою базуються на потужностях великих мовних моделей, здатних ефективно обробляти, інтерпретувати та структурувати величезні масиви текстової інформації. Розвиток архітектур трансформерів відкрив можливості для створення моделей різного масштабу, які відрізняються гнучкістю застосування та адаптацією до специфічних завдань. Такі завдання включають класифікацію

тексту, семантичний аналіз метаданих, генерацію коду, обробку довгих контекстів, інтерактивну взаємодію з користувачем та формування рекомендацій на основі аналізу запитів. Ефективність великих мовних моделей визначається не лише кількістю параметрів, а й вдосконаленням механізмів уваги, оптимізацією обробки контекстів, а також балансом між продуктивністю і ресурсною ефективністю під час навчання та виведення результатів. У цьому контексті моделі Mistral 7B, LLaMA 3, Claude 3 та Falcon 180B демонструють різні стратегії балансування продуктивності, точності та обчислювальної ефективності, що робить їх ключовими інструментами для побудови сучасних семантичних систем пошуку та аналітики інформаційних ресурсів.

Модель Mistral 7B, що містить приблизно 7 мільярдів параметрів, створена з акцентом на високу продуктивність та енергоефективність. Вона демонструє значні результати у завданнях розуміння природної мови, генерації коду, логічного мислення та вирішення комплексних запитів, перевершуючи більші моделі в окремих сценаріях завдяки оптимізованим механізмам уваги та архітектурним рішенням [27]. На практиці це означає, що модель може успішно використовуватися в системах семантичного пошуку, де важлива висока якість обробки запитів при обмежених обчислювальних ресурсах, наприклад, на локальних серверах або при автономному розгортанні без доступу до хмарних платформ. Для прискореного виведення результатів Mistral 7B застосовує оптимізовані алгоритми уваги, такі як grouped-query attention (GQA) та sliding window attention (SWA), що дозволяє обробляти великі текстові контексти без істотного збільшення обчислювальних витрат та зменшує затримку у генерації відповідей у режимі діалогу. Поєднання високої швидкодії та енергоефективності робить модель придатною для інтеграції у масштабовані або автономні рішення, включно з корпоративними системами аналітики, платформами підтримки розробників та інструментами автоматизованого семантичного пошуку, де важливі одночасно точність, швидкість та ресурсна оптимізація.

LLaMA 3, розроблена компанією Meta AI, є черговим поколінням відкритої лінійки моделей LLaMA і базується на декодер-трансформері з низкою інноваційних рішень. Використання великого словника токенів і механізму Grouped Query Attention забезпечує ефективну обробку багатомовних текстів і одночасне зменшення обчислювальних витрат [28]. Завдяки цьому модель може якісно працювати з даними різного походження, що робить її корисною для глобальних систем, де необхідно охоплювати тексти різними мовами. Підтримка великих контекстних вікон до 8 192 токенів дозволяє зберігати релевантну контекстну інформацію для складних або довгих запитів, що важливо при аналізі технічної документації, порівнянні специфікацій програмних продуктів або моніторингу змін у репозиторіях відкритого ПЗ. Наявність різних масштабів моделі, від 8B до 70B параметрів, забезпечує гнучкість у виборі моделі відповідно до наявних обчислювальних ресурсів і поставлених завдань, підтримуючи високу продуктивність у генерації коду, перекладі текстів та логічному мисленні, що робить LLaMA 3 ефективним інструментом для прикладних рішень із поєднанням аналітики, асистування та генерації текстів.

Claude 3, розроблена Anthropic, вирізняється підходом «Constitutional AI», що підвищує контроль над безпекою та стабільністю відповідей [29]. Модель демонструє стабільність і зрозумілість у відповідях на складні логічні запити, глибокий понятійний аналіз та здатність формувати детальні пояснення. Це робить Claude 3 особливо цінною для систем семантичного пошуку, де окрім релевантності результатів важливо надавати користувачеві зрозумілі рекомендації, синтезовану інформацію та підтримку прийняття рішень на основі контексту запиту. Модель здатна ефективно поєднувати семантичне розуміння, генерацію тексту та логічний аналіз, що робить її придатною для інтеграції у системи корпоративного та дослідницького рівня, включаючи інструменти аналітики і консультування на основі текстових даних.

Falcon 180B, розроблена Technology Innovation Institute, є однією з найбільших відкритих моделей із 180 мільярдами параметрів і була натренована на надвеликих корпусах даних із застосуванням багатомодульної уваги, що

дозволяє ефективно обробляти величезні обсяги інформації та забезпечує високу точність генерації тексту, структурування знань і логічного аналізу [30]. Ліцензійні умови моделі дозволяють її використання як у наукових дослідженнях, так і в комерційних проєктах, що робить Falcon 180B привабливим інструментом для масштабних систем семантичного пошуку, де критично важливі обсяг оброблюваних даних, продуктивність і точність результатів. Завдяки масштабованій обчислювальній архітектурі та підтримці роботи з довгими контекстами, модель забезпечує комплексну підтримку аналізу великих масивів даних, автоматизацію аналітичних процесів та генерацію рекомендацій у реальному часі. Це робить Falcon 180B однією з найбільш перспективних моделей для інтеграції в корпоративні та дослідницькі платформи, орієнтовані на семантичний пошук і глибинний аналіз інформації.

У комплексі використання моделей різного масштабу та архітектурних особливостей дозволяє системам семантичного пошуку ефективно адаптуватися до різних умов і ресурсних обмежень. Mistral 7B забезпечує високу продуктивність і швидкість обробки запитів на обмеженому апаратному забезпеченні, LLaMA 3 надає глибоке розуміння контексту та можливість роботи з довгими текстами, Claude 3 додає функціональність формування пояснень і контекстних рекомендацій, а Falcon 180B гарантує високу пропускну здатність та точність при обробці великих обсягів даних. Така інтеграція дозволяє системам не лише підвищити точність і швидкість видачі результатів, а й забезпечує глибину аналітики, інтелектуальну підтримку прийняття рішень та персоналізацію результатів. В результаті поєднання цих моделей створює комплексні рішення для інтелектуального семантичного пошуку, здатні ефективно працювати у різноманітних прикладних сценаріях, від локальних корпоративних систем до глобальних відкритих платформ, забезпечуючи оптимальний баланс між точністю, продуктивністю та ресурсною ефективністю.

3 КОНЦЕПЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ СЕМАНТИЧНОГО ПОШУКУ

3.1 Функціональні вимоги

3.1.1 Типи даних та метадані, що обробляються системою

Інтелектуальна система семантичного пошуку програмних продуктів функціонує у середовищі відкритих даних, які характеризуються високою варіативністю структури, неоднорідністю джерел та відмінностями у ступені формалізованості. Тому підсистема збору та інтерпретації інформації повинна забезпечувати перетворення неструктурованих, напівструктурованих та структурованих даних у єдиний уніфікований інформаційний простір, придатний для подальшої обробки методами глибинного семантичного аналізу. У межах цього процесу система оперує не лише текстовим вмістом, а й числовими, категоріальними, теговими, графовими та поведінковими метаданими, що відображають багатовимірну природу опису програмних продуктів в екосистемах відкритого програмного забезпечення.

Основу даних становлять текстові описи проєктів, отримані з відкритих репозиторіїв, офіційної документації та супровідних сторінок програмних продуктів. Це README-файли, мануали, технічні специфікації, журнали змін та інші текстові структури, що містять семантично насичені відомості про функціональність, архітектуру, призначення та сумісність програмного забезпечення [31]. Їхня контентна складність зумовлює необхідність застосування трансформерних моделей для формування векторизованих представлень, які здатні зберігати контекстні залежності між ключовими поняттями та дозволяють здійснювати пошук на семантичному рівні, а не за ключовими словами.

Другу групу становлять метадані, які забезпечують формалізований опис структурних та експлуатаційних характеристик програмного продукту. До них

належать назва проєкту, версія, дата останнього оновлення, перевірений статус підтримки, використані технології, мова програмування, ліцензійна модель, числові індикатори активності (кількість зірок, форків, контриб'юторів), рівень стабільності, частота виходу релізів і показники динаміки розвитку. Ці параметри є ключовими для моделювання рейтингу релевантності, формування рекомендаційних сигналів та побудови профілів програмних продуктів, які відображають ступінь їхнього розвитку та актуальності у часовому вимірі.

Особливу роль відіграють взаємозв'язки між проєктами, представлені як графові метадані. До них входять залежності між бібліотеками, взаємні імпорти, крос-посилання, належність до певних технологічних екосистем та спільнот розробників. Завдяки цим зв'язкам система здатна будувати граф програмних продуктів та використовувати методи графових нейронних мереж для підвищення точності семантичного зіставлення, адже контекст проєкту визначається не лише його внутрішнім описом, а й позицією у технологічному середовищі.

Система також використовує поведінкові та статистичні метадані, отримані з історій активності користувачів відкритих репозиторіїв і динаміки змін у кодовій базі програмних проєктів. До таких метаданих належать частота та регулярність комітів, співвідношення між додаванням нового функціоналу й виправленням помилок, характер і масштаб оновлень, інтенсивність участі різних розробників, рівень централізації або децентралізації внесків, поява критичних виправлень безпеки, а також реакція спільноти на релізи у вигляді обговорень, запитів на доопрацювання та швидкості прийняття змін. Аналіз цих показників дає змогу оцінювати поточний стан проєкту, стадію його життєвого циклу, рівень підтримки та активності спільноти, а також здатність програмного продукту адаптуватися до технологічних змін і нових вимог, що є вагомим чинником у процесі пошуку та вибору оптимального програмного забезпечення для практичного використання.

Окрему роль у системі відіграють контекстні дані, пов’язані з категоризацією та тематичною класифікацією програмних проєктів. Вони представлені як явними атрибутами у вигляді тегів і тематичних міток, заданих авторами або спільнотою, так і неявними ознаками, зокрема автоматично згенерованими класами, сформованими на основі семантичного аналізу описів, README-файлів і супровідної документації. Додатково враховуються зовнішні категорії та таксономії, що визначаються платформами на кшталт GitHub чи GitLab [32], які відображають загальноприйняті у спільноті уявлення про домен застосування та функціональне призначення програмного продукту. Використання такого типу метаданих забезпечує ефективну попередню фільтрацію результатів і дозволяє гнучко адаптувати процес пошуку до конкретних предметних областей, зокрема веброзробки, кібербезпеки, глибинного навчання, аналізу даних або системного програмування.

Усі зазначені типи даних у сукупності формують комплексну й багаторівневу базу знань, у межах якої кожен програмний продукт описується багатовимірним набором характеристик, що охоплює функціональні, технологічні, структурні, поведінкові та соціальні аспекти. В інтелектуальній системі ці дані проходять послідовні етапи стандартизації, очищення від шуму, нормалізації та контекстуального структурування з подальшим перетворенням у векторні та графові представлення. Такий підхід дає змогу поєднувати переваги розподілених семантичних просторів і графів знань, що забезпечує високоточний семантичний пошук і порівняння програмних продуктів. У цьому випадку релевантність результатів визначається не лише поверхневою текстовою подібністю, а й глибинним аналізом взаємозв’язків між функціональними можливостями, архітектурними рішеннями, технологічним стеком і реальними моделями використання програмного забезпечення в екосистемі відкритого коду.

3.1.2 Основні послуги та можливості системи

Інтелектуальна система семантичного пошуку програмних продуктів орієнтована на забезпечення комплексної підтримки користувача під час навігації у великомасштабних наборах відкритих даних. Її функціональні можливості охоплюють як традиційний пошук, так і глибинний аналіз зв'язків, вмісту та контексту програмних продуктів, що дозволяє значно підвищити точність відбору релевантних проєктів у порівнянні з класичними методами інформаційного пошуку.

Основні сервіси системи формують інтегровану інформаційно-аналітичну платформу, яка здатна інтерпретувати семантику описів, структуру залежностей між проєктами, динаміку їхньої еволюції та відповідність користувацьким потребам. У межах цієї платформи реалізовано такі ключові можливості [33]:

– семантичний пошук програмних продуктів (система формує контекстуальні векторні представлення описів програмного забезпечення та здійснює пошук не за ключовими словами, а за глибинною семантичною подібністю);

– автоматична класифікація та категоризація проєктів (на основі лінгвістичних моделей та метаданих система визначає доменну належність програмного продукту, здатна структурувати його за тематикою, функціональністю, технологічним стеком та рівнем складності);

– аналіз залежностей і графових зв'язків (платформа виявляє структурні взаємозв'язки між проєктами, бібліотеками та інфраструктурними компонентами, формує граф екосистеми відкритого коду та застосовує графові нейронні моделі для оцінювання близькості та сумісності програмних рішень);

– оцінювання якості та життєвого циклу програмного продукту (система аналізує показники активності розробників, частоту оновлень, регулярність релізів, наявність критичних виправлень та інші поведінкові параметри, що дозволяє формувати індикатори стабільності та перспективності проєкту);

– персоналізовані рекомендації (на основі профілю користувача, історії пошукових запитів та поведінкових патернів система генерує рекомендаційні пропозиції, ураховуючи технологічні потреби, доменні інтереси та рівень компетентності користувача);

– фільтрація та ранжування результатів (система забезпечує багатовимірне ранжування, у якому враховується семантична близькість, якість метаданих, динаміка розвитку проєкту та інші релевантні характеристики, що дозволяє формувати точний та інформативний рейтинг);

– візуалізація аналітичних даних (платформа надає засоби графічного відображення структури залежностей, часових змін активності, тематичних кластерів та інших параметрів, що полегшує інтерпретацію складних взаємозв’язків у середовищі відкритого програмного забезпечення).

У сукупності ці можливості забезпечують не лише ефективну навігацію у великих масивах відкритих даних, а й створюють підґрунтя для комплексного аналізу програмних продуктів, дозволяючи користувачеві приймати обґрунтовані рішення щодо вибору, порівняння та подальшого використання програмного забезпечення.

3.2 Алгоритм роботи системи

3.2.1 Сценарії взаємодії користувача

Інтелектуальна система семантичного пошуку програмних продуктів, побудована на базі великої мовної моделі, забезпечує природну та контекстно обґрунтовану взаємодію з користувачем, підтримуючи повний цикл запиту від первинного звернення до формування релевантних рекомендацій. Особливістю системи є здатність динамічно адаптуватися до намірів користувача, аналізувати відкриті дані про програмні продукти та формувати змістовні відповіді з урахуванням індивідуальних інформаційних потреб.

Після ініціювання взаємодії система вітає користувача та формує уточнювальний запит, спрямований на визначення наміру: пошук програмного продукту, аналіз наявних рішень, порівняння характеристик або отримання рекомендацій. Мовна модель використовує українські лінгвістичні норми та ввічливі мовленнєві формули, забезпечуючи природність комунікації. Приклад схеми промпту із запитанням та контекстом відповіді представлено на рис. 3.1.

Рисунок 3.1 – Приклад схеми промту

Приклад взаємодії для семантичного пошуку програмного забезпечення:

Сценарій 1: Базовий пошук та контекстне уточнення

Цей сценарій демонструє здатність системи розуміти природну мову та уточнювати результати без необхідності повторного введення повного запиту.

- Дія користувача: Вводить у пошуковий рядок запит: *«Система керування проєктами з відкритим кодом»*.
- Реакція системи: Виконує семантичний аналіз, ідентифікує намір пошуку та формує список карток релевантних продуктів (наприклад, Taiga, OpenProject). Для кожного продукту відображається згенерований LLM опис, що підкреслює їхнє призначення.
- Дія користувача: Вводить уточнення в той самий рядок (не змінюючи весь текст): *«...на мові Python з підтримкою API»*.
- Реакція системи: Модуль *Context Tracking* враховує контекст попереднього запиту. Система фільтрує попередню видачу, виключаючи продукти на інших

мовах, і залишає лише відповідні (наприклад, Taiga). Опис у картках автоматично оновлюється: тепер він акцентує увагу саме на технологічному стеку Python та можливостях API.

- Дія користувача: Натискає на картку проєкту.
- Реакція системи: Відкриває сторінку деталізації, де надає розширену інформацію: інструкцію з розгортання, статистику активності репозиторію та аналіз ліцензії.

Сценарій 2: Робота з динамічними фільтрами

Система допомагає користувачеві звузити коло пошуку, пропонуючи релевантні критерії на основі аналізу знайдених результатів.

- Стан системи: Після виконання запиту *«системи керування проєктами на Python»*, система відображає панель рекомендованих фільтрів, згенерованих на основі метаданих знайдених проєктів (наприклад, «Тип ліцензії», «Рівень активності», «Останнє оновлення»).
- Дія користувача: Обирає запропонований фільтр: *«Ліцензія MIT»*.
- Реакція системи: Миттєво оновлює список результатів, залишаючи лише проєкти з ліцензією MIT. LLM додає до опису коротку примітку про переваги цієї ліцензії для обраних проєктів.

Сценарій 3: Персоналізовані рекомендації

Система враховує історію пошукових сесій для ранжування результатів, адаптуючи видачу під професійні інтереси користувача.

- Контекст: Користувач раніше цікавився інструментами для аналітики даних на Python.
- Дія користувача: Відкриває головну сторінку або вводить загальний запит *«інструменти візуалізації»*.
- Реакція системи: У блоці «Рекомендоване» або на початку списку видачі система пріоритезує проєкт Redash.
- Обґрунтування системи: У картці продукту відображається пояснення (Highlight): *«Рекомендовано на основі вашого інтересу до Python та Analytics. Проєкт має відкритий код та активну спільноту»*.

Сценарій 4: Аналіз ліцензій та обмежень

Система замінює необхідність читати юридичні тексти, надаючи синтезовані відповіді щодо можливостей використання ПЗ.

- Дія користувача: Вводить запит або обирає опцію в інтерфейсі: *«Показати можливості комерційного використання»*.
- Реакція системи: Для знайденого продукту (наприклад, під ліцензією Apache 2.0) система генерує та виводить у картці статус: *«Дозволено для комерційного використання»*.
- Деталізація: Система надає стислий згенерований огляд ключових положень ліцензії: вимоги до зазначення авторства, відсутність гарантій та умови розповсюдження змін.

3.2.2 Обробка запитів та видача релевантних результатів

Основні функції системи інтелектуального пошуку, побудованої на базі великої мовної моделі, полягають у перетворенні користувацьких запитів на осмислені, релевантні відповіді у режимі реального часу. Для цього задіюється комплекс взаємопов’язаних технологічних компонентів, які працюють узгоджено та забезпечують високоточну взаємодію в контексті семантичного пошуку програмного забезпечення з відкритим кодом.

Модуль NLP виконує глибокий аналіз україномовних запитів і включає кілька ключових етапів [34]:

– попереднє опрацювання тексту (перед безпосереднім аналізом вхідний текст нормалізується шляхом токенізації, очищення від нерелевантних слів, усунення пунктуаційних артефактів та приведення лексем до базових форм. Це підвищує точність подальших етапів обробки);

– визначення намірів (алгоритми машинного навчання, такі як SVM, рекурентні моделі або трансформерні архітектури, використовуються для класифікації запиту за типом інформаційної потреби користувача;

– виявлення сутностей (система розпізнає іменовані сутності, що стосуються предметної області: назви пакетів, версій, технологій, мов програмування, ліцензій, репозиторіїв GitHub, категорій програмних продуктів. Вилучення таких сутностей дозволяє точно інтерпретувати запит і використовувати їх у подальшій семантичній обробці);

– аналіз тональності та стилю (виявлення емоційних та прагматичних ознак допомагає адаптувати стиль відповіді. У сфері пошуку ПЗ це дозволяє, наприклад, розпізнати запит про проблему, невизначеність або нагальність пошуку рішення).

Модуль NLP навчається на розширених та спеціалізованих корпусах україномовних текстів, що охоплюють не лише стандартні тексти, а й матеріали, пов’язані з програмним забезпеченням, технічною документацією, README-файлами, каталогами open-source проєктів та технічними оглядами. Для підвищення якості навчання та зменшення впливу обмеженої кількості вітчизняних технічних корпусів застосовуються техніки донавчання (fine-tuning) та доповнення даних (data augmentation). Це дозволяє системі більш коректно обробляти спеціалізовані терміни, абревіатури, специфічну лексику та структуровану інформацію, притаманну українським технічним текстам.

Після визначення наміру користувача та ключових сутностей активується модуль генерації відповідей, який забезпечує створення повноцінного контенту у відповідь на запит. Система використовує сучасні трансформерні моделі (GPT-подібні), адаптовані для роботи з українською технічною термінологією та специфічними контекстами програмної документації. Попереднє навчання на великих корпусах англомовних і україномовних текстів дозволяє моделі ефективно опрацьовувати широкий спектр контенту, включаючи каталоги ПЗ, технічні огляди, документацію, обговорення та питання з open-source спільнот, а також форуми й технічні блоги.

Для формування узгодженої та максимально релевантної відповіді здійснюється підготовка спеціалізованого промпту. Запит до моделі містить структуровані елементи: визначений намір користувача, вилучені сутності

(наприклад, назву фреймворка, тип ліцензії, суміжні технології), а також додатковий контекст, отриманий із довідкових джерел, документації або зовнішніх баз знань.

На основі підготовленого промпту модель формує завершену відповідь, яка може включати різні типи інформації: опис програмного забезпечення, рекомендації щодо його застосування, порівняння аналогів, альтернативні рішення, а також пояснення технічних характеристик та вимог до середовища виконання. Система здатна враховувати специфіку галузі та контекст запиту, що дозволяє отримати більш релевантний та практично застосовний результат.

Згенеровані відповіді проходять етап уточнення та верифікації: видаляються потенційно нерелевантні фрагменти, перевіряється відповідність технічній термінології, корегуються неточності, а за наявності – додається структурована інформація, наприклад, версії програмного забезпечення, посилання на документацію або каталоги. Цей процес дозволяє забезпечити високу якість вихідних відповідей, їхню зрозумілість, точність та практичну користь для кінцевого користувача.

У межах семантичного пошуку роль внутрішньої бази даних виконують каталоги й API відкритого програмного забезпечення, зокрема:

– метадані з GitHub, GitLab, SourceForge;

– записи з API каталогів програмних пакетів (PyPI, npm, Maven Central тощо);

– відкриті онтології та таксономії ПЗ;

– описи проєктів із відкритих реєстрів ліцензій.

Ці джерела слугують базою знань, яку система використовує для уточнення версій, залежностей, актуальності, популярності, типів ліцензування та сумісності.

Система звертається до зовнішніх API у таких випадках [35]:

– уточнення характеристик знайденого ПЗ;

– отримання параметрів релізів;

– доступ до документації та README;

– пошук репозиторіїв за ключовими словами;

– оновлення інформації щодо активності проєкту (частота комітів, кількість контриб’юторів тощо).

Таким чином розроблена система підтримує високоточний семантичний пошук, орієнтований на актуальні та надійні відкриті джерела.

Для підтримання високої продуктивності система використовує механізми безперервного навчання:

– обробка користувацького зворотного зв’язку (аналізуються оцінки, зауваження та повідомлення про помилки, які допомагають виявляти неточності або прогалини у знаннях);

– аналіз діалогових логів (автоматично виділяються нові патерни запитів, типові технічні потреби, некоректні класифікації намірів чи сутностей. Це дозволяє своєчасно оновлювати семантичні онтології);

– перенавчання моделей NLP/LLM (моделі регулярно донавчаються на нових корпусах – описах нових бібліотек, технічних статтях, нових версіях документації);

– розширення доменної бази знань (додаються нові категорії програмного забезпечення, оновлені метадані релізів, зміни в ліцензійних політиках);

– участь фахівців технічної галузі (експерти допомагають уточнювати класифікацію ПЗ, стандартизувати терміни та формувати рекомендації, що підвищує точність семантичного пошуку).

Процес обробки користувацьких запитів у системі інтелектуального пощуку, орієнтованому на семантичний пошук програмного забезпечення з відкритим кодом, передбачає інтеграцію передових NLP-технологій, генеративних можливостей LLM, зовнішніх API проєктів open-source та інтелектуальних механізмів адаптації. Завдяки цьому користувач отримує точні, змістовні та контекстуально обґрунтовані відповіді, а система здатна постійно вдосконалюватися та реагувати на динамічні зміни у сфері програмного забезпечення.

4 ТЕХНІЧНА РЕАЛІЗАЦІЯ

4.1 Вибір та налаштування фреймворку для семантичного пошуку

Фреймворк Semantic Kernel є сучасним і гнучким інструментом для створення інтелектуальних систем обробки природної мови, зокрема діалогових пошукових інтерфейсів на базі великих мовних моделей (LLM) [36]. Його архітектура орієнтована насамперед на застосування у середовищі C#, що робить його придатним для розробки системи інтелектуального пошуку, здатної виконувати семантичний пошук програмних продуктів за відкритими даними [37].

Однією з головних переваг Semantic Kernel у цьому контексті є сумісність із широким набором мовних моделей, включаючи моделі, адаптовані для української мови. Це забезпечує коректну інтерпретацію запитів користувачів та генерацію природних, змістовних відповідей. Додатковою сильною стороною фреймворку є простота інтеграції з існуючими інформаційними системами, що дозволяє без істотних витрат адаптувати його під потреби платформ, пов’язаних із обробкою відкритих даних.

Важливим чинником вибору Semantic Kernel є його здатність уніфікувати взаємодію з різними LLM, забезпечуючи єдиний програмний інтерфейс для роботи з моделями різних постачальників. Це дає змогу безперешкодно підключати нові покоління мовних моделей – зокрема ті, що орієнтовані на українську термінологію або спеціалізовані на технічних даних – без необхідності суттєвих змін у кодовій базі програмного застосунку. Такий підхід гарантує тривалу актуальність системи та її здатність адаптуватися до технологічного розвитку у сфері LLM.

Окремою перевагою є модульна та розширювана архітектура, яка дозволяє налаштовувати функціональність відповідно до вимог семантичного пошуку ПЗ. Розробники можуть підключати нові доменні бази знань, інтегрувати зовнішні джерела відкритих даних (каталоги open-source, API репозиторіїв, технічну

документацію) або створювати спеціалізовані компоненти для класифікації запитів, аналізу намірів, фільтрації результатів чи ранжування знайдених програмних продуктів.

Процес налаштування Semantic Kernel включає кілька основних етапів:

– встановлення та конфігурація (спочатку потрібно інсталювати фреймворк разом із залежностями – через NuGet, термінал або шляхом клонування офіційного репозиторію. Після цього виконується первинне налаштування оточення, зокрема під'єднання до обраної LLM, визначення форматів відповідей та параметрів взаємодії з API відкритих джерел);

– інтеграція мовної моделі (Semantic Kernel підтримує широкий спектр LLM, що дозволяє обрати модель для конкретної задачі: генерації тексту, класифікації, узагальнення чи побудови рекомендацій. На цьому етапі налаштовуються токенізатори, параметри генерації та механізми обслуговування запитів);

– формування або підключення бази знань (для системи семантичного пошуку важливим є доступ до структурованих джерел: метаданих програмних продуктів, описів проєктів, технічної документації, даних з GitHub/GitLab, інформації про версії, ліцензії та залежності. Semantic Kernel дозволяє інтегрувати такі джерела як окремі модулі, завантажувати їх у пам'ять або використовувати на вимогу через зовнішні API);

– кастомізація та розширення функцій (фреймворк надає широкий набір API для створення власних плагінів, навичок і процесорів контексту. Це може включати розробку компонентів для аналізу намірів, модулів семантичного ранжування результатів, інтеграцію нових онтологій або інструментів для обробки технічних термінів, типових для сфери ПЗ);

– розгортання та масштабування (Semantic Kernel оптимізований для розгортання в хмарних середовищах і здатен ефективно масштабуватися відповідно до зростання навантаження. Це дає змогу обробляти значні обсяги запитів у реальному часі без зниження продуктивності).

У процесі конфігурації важливо враховувати специфіку української мови, особливості локальної термінології, а також вимоги щодо роботи з відкритими даними. Це включає правильну інтеграцію українських лінгвістичних ресурсів, адаптацію стилю відповідей та забезпечення відповідності нормативним вимогам щодо обробки інформації.

Суттєвою перевагою фреймворку є також активна міжнародна спільнота та розвинена документація, які забезпечують підтримку, оновлення та приклади застосування. Це прискорює впровадження фреймворку у проєкти, пов’язані із семантичним пошуком програмного забезпечення.

Прийняття Semantic Kernel дає змогу українським організаціям і дослідникам розгорнути потужну інтелектуальну платформу для створення LLM-орієнтованих пошукових рішень, здатних ефективно працювати з відкритими даними, адаптованими мовними моделями та великомасштабними базами знань. Модульність, масштабованість і гнучкість фреймворку роблять його оптимальним вибором для розробки сучасних діалогових систем, орієнтованих на автоматизований семантичний пошук та аналіз програмних продуктів.

4.2 Інтеграція LLM у систему та збереження контексту розмови

Підтримка зв’язного та контекстуально обґрунтованого потоку інформації є критичною для забезпечення ефективного та персоналізованого пошуку у системах семантичного аналізу програмного забезпечення з відкритим кодом [38]. У таких системах важливе точне відстеження контексту запитів користувача та історії взаємодії, що дозволяє надавати релевантні результати і рекомендації. У цьому підрозділі розглядаються методи управління контекстом і збереження інформації про запити, які забезпечують коректну роботу семантичного пошуку.

Одним із ключових підходів є використання управління сесіями та контекстних об’єктів. Сесія визначається як окремий екземпляр взаємодії

користувача із системою, що містить усю релевантну інформацію про стан пошуку.

Для системи семантичного пошуку об’єкт сесії може містити такі категорії даних:

– профіль користувача (включає інформацію про попередні запити, мовні уподобання, інтереси в галузі програмного забезпечення та інші релевантні параметри. Використання цих даних дозволяє персоналізувати результати пошуку та рекомендації на основі індивідуальних потреб користувача);

– історія запитів (хронологічний запис запитів, введених користувачем протягом поточного сеансу. Це дозволяє системі враховувати попередні запити при формуванні релевантних результатів і рекомендацій);

– відстеження намірів та контекстних цілей (система може визначати наміри користувача (наприклад, пошук бібліотеки, фреймворка або інструменту) та контекстні цілі (наприклад, сумісність із певними технологіями), що дозволяє коректно адаптувати результати пошуку);

– контекстні змінні (збереження параметрів пошуку або додаткових фільтрів дозволяє забезпечити узгодженість і точність рекомендацій);

Завдяки управлінню сесіями та контекстним об’єктам система семантичного пошуку може ефективно відстежувати стан користувацького сеансу та забезпечувати релевантні відповіді, а також підтримувати безперервність роботи при переході між сеансами або пристроями.

Крім явних методів управління контекстом, великі мовні моделі (LLM) здатні підтримувати контекст у неявній формі через внутрішні представлення та механізми уваги. Моделі на кшталт GPT-3, навчені на великих корпусах тексту, здатні розуміти семантичні зв’язки та контекстні підказки. Включення історії запитів користувача до запиту LLM дозволяє генерувати рекомендації та відповіді, які враховують попередні запити, зберігаючи зв’язність і релевантність.

Хоча LLM можуть частково фіксувати контекст, для підтримки довготривалих і узгоджених взаємодій корисним є поєднання явних і неявних

методів. Наприклад, застосування моделей із розширеною пам’яттю дозволяє зберігати та витягувати релевантну інформацію з історії запитів. Пошуково-доповнені мовні моделі, у свою чергу, дозволяють отримувати інформацію із зовнішніх джерел знань, таких як репозиторії з відкритим кодом або бази знань про проєкти [39].

Іншим ефективним підходом є використання контекстних вбудовувань та механізмів уваги в архітектурі сучасних мовних моделей. Контекстні вбудовування дозволяють представити запити у вигляді векторних репрезентацій, які враховують навколишній контекст, тобто не лише конкретне ключове слово або фразу, а й попередні запити, тему розмови та особливості користувача. Механізми уваги, у свою чергу, дозволяють моделі фокусуватися на тих частинах історії запитів або документації, які є найбільш релевантними для поточного завдання. Завдяки цьому модель може динамічно виділяти пріоритетну інформацію, підвищуючи точність генерації відповідей та зменшуючи ймовірність включення нерелевантних даних.

Поєднання контекстних вбудовувань і механізмів уваги дозволяє досягти високого рівня узгодженості та точності рекомендацій. Наприклад, у системах семантичного пошуку це означає, що результати не просто відповідають ключовим словам запиту, а враховують історію попередніх взаємодій користувача, його наміри та потенційні уточнення.

Гібридний підхід, який інтегрує управління сесіями, контекстні об’єкти, моделі з розширеною пам’яттю та контекстні вбудовування, забезпечує більш надійний та послідовний семантичний пошук. Управління сесіями дозволяє зберігати стан користувацької взаємодії між запитами, тоді як контекстні об’єкти виступають як структуровані елементи даних, що представляють конкретні аспекти інформації, наприклад, параметри продукту, часові рамки чи тематичні категорії. Моделі з розширеною пам’яттю здатні зберігати та використовувати цю інформацію протягом тривалих сесій, що особливо корисно для систем, які виконують комплексні завдання або ведуть багатоетапні діалоги.

Архітектурні підходи, характерні для систем Conversational AI (CAA), були адаптовані для забезпечення ефективного керування контекстом пошукових запитів. Використання таких патернів дозволяє відстежувати історію взаємодій користувача та генерувати релевантні відповіді, що враховують попередні ітерації пошуку. Така архітектура інтегрує контекстні вбудовування, механізми уваги, модулі управління сесіями та зовнішні бази знань, що дозволяє забезпечити природний та послідовний процес уточнення пошукових критеріїв.

На рис. 4.1 представлено високорівневу архітектуру підсистеми обробки запитів, побудовану на принципах CAA. Компонент «Розуміння природної мови» (NLU) аналізує вхідний запит користувача, виділяє наміри (Intent) та ключові сутності (Entities). Компонент управління логікою взаємодії (на схемі – DialogManagement) визначає стратегію обробки: чи потрібно виконати новий пошук, чи застосувати фільтр до існуючих результатів. Компонент Context Tracking підтримує та оновлює вектор стану сесії, зберігаючи накладені фільтри. Компонент генерації рекомендацій використовує мовну модель для формування опису знайдених продуктів із врахуванням історії запитів та інформації із зовнішніх джерел.

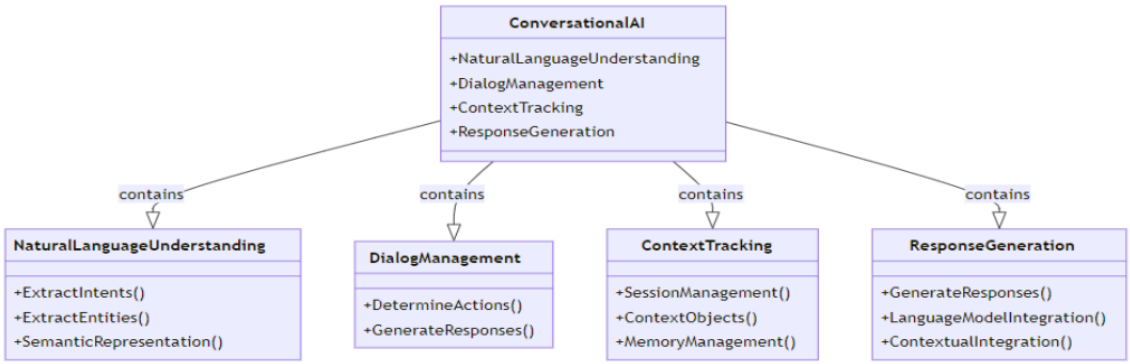

Рисунок 4.1 – Архітектура підсистем інтелектуальної обробки запитів

5 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

5.1 Тестування системи з різними LLM

5.1.1 Характеристика метрик оцінки LLM

Для оцінки ефективності системи семантичного пошуку програмного забезпечення з відкритим кодом застосовувалися класичні метрики інформаційного пошуку, що дозволяють комплексно оцінити якість виданих результатів. Однією з ключових метрик є Precision@K, яка визначає частку релевантних елементів серед перших K результатів пошуку. Формально вона визначається як [40]:

$$Precision@K = \frac{|Rel_K|}{K} \tag{5.1}$$

де $|Rel_K|$ – кількість релевантних документів у топ-K.

Ця метрика дозволяє оцінити, наскільки точними є найбільш ранні результати, що має критичне значення для користувача, оскільки більшість звернень до пошукової системи зосереджена саме на верхніх позиціях списку.

Метрика Recall@K, або повнота на K результатах, оцінює, яку частину всіх релевантних ресурсів система змогла знайти у топ-K результатах, і визначається як [41]:

$$Recall@K = \frac{|Rel_K|}{|Rel_{total}|} \tag{5.2}$$

де $|Rel_K|$ – кількість релевантних документів у топ-K;

$|Rel_{total}|$ – загальна кількість релевантних документів у базі.

Вона дозволяє оцінити здатність системи виявляти всі релевантні ресурси, що є важливим аспектом при побудові семантичного пошуку, особливо коли користувачу важливо отримати максимальний обсяг корисної інформації.

Ще однією важливою метрикою є Mean Reciprocal Rank (MRR), яка характеризує середню позицію першого релевантного результату у видачі для набору запитів. MRR визначається формулою [42]:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}, \tag{5.3}$$

де $rank_i$ – позиція першого релевантного результату i-го запиту;

N – загальна кількість запитів.

Ця метрика дозволяє оцінити, наскільки швидко користувач може отримати релевантну інформацію при взаємодії з системою.

Normalized Discounted Cumulative Gain (nDCG@K) забезпечує більш детальну оцінку якості видачі, враховуючи не лише наявність релевантних результатів, а й їхню позицію у рейтингу. Вона визначається як [43]:

$$nDCG@K = \frac{DCG@K}{IDCG@K}, \tag{5.4}$$

де:

$$DCG@K = \sum_{i=1}^{K}\frac{2^{rel_i}-1}{\log_2(i+1)}, \tag{5.5}$$

та IDCG@K – ідеальне значення DCG@K для ранжованого списку, впорядкованого за релевантністю. nDCG дозволяє оцінити не лише наявність релевантних результатів, а й їхню позицію у топ-K, що особливо важливо для

семантичного пошуку, оскільки релевантні результати, що з’являються на верхніх позиціях, значно підвищують ефективність користувацької взаємодії.

Затримка відповіді системи (Latency) відображає час, необхідний для генерації релевантних результатів пошуку. Вона вимірюється у секундах і є критичною для практичної застосовності системи, оскільки навіть високоточна система може втратити користувача, якщо час очікування відповіді є неприйнятним. У контексті семантичного пошуку, де часто застосовуються великі мовні моделі, баланс між якістю результатів і швидкодією є важливим аспектом оцінки ефективності системи.

5.1.2 Методика формування еталонної вибірки та оцінки релевантності

Для коректного розрахунку метрик, наведених у пункті 5.1.1, зокрема nDCG та Precision, критично важливою є наявність валідованої еталонної вибірки (Ground Truth). Оскільки дослідження спрямоване на специфічну предметну область (пошук програмного забезпечення з відкритим кодом за україномовними запитами), використання стандартних датасетів є неможливим через мовні та контекстуальні відмінності. Тому було розроблено власну методологію оцінки, що базується на підході «об’єднаної оцінки» (Pooled Evaluation) із залученням експертної верифікації.

Процес формування тестового набору складався з трьох етапів:

− Генерація запитів. Було сформовано набір із 50 тестових запитів різної типології, що покривають основні сценарії використання системи: навігаційні (пошук конкретного інструменту), інформаційні (пошук рішення проблеми) та транзакційні (порівняння характеристик).

− Збір пулу кандидатів. Для кожного запиту було агреговано топ-10 результатів від усіх досліджуваних моделей. Отримані списки об’єднувалися, а дублікати видалялися, що дозволило сформувати єдиний пул документів-кандидатів для "сліпого" оцінювання, уникаючи упередженості щодо конкретної моделі.

− Оцінювання релевантності. Для розрахунку метрики nDCG (формула 5.4) було застосовано чотирирівневу шкалу релевантності (rel_i):

i. 3 бали (Ідеально): Програмний продукт повністю вирішує задачу, має активну підтримку та відповідає технічним обмеженням.

ii. 2 бали (Релевантно): Продукт корисний, але вимагає доопрацювання або є менш популярним аналогом.

iii. 1 бал (Частково релевантно): Продукт належить до доменної області, але не вирішує конкретну задачу.

iv. 0 балів (Нерелевантно): Результат помилковий або не стосується запиту.

Верифікація оцінок проводилася комбінованим методом: первинна розмітка здійснювалася за підходом LLM-as-a-Judge (із використанням моделі GPT-4 як незалежного арбітра), після чого проводилася вибіркова ручна перевірка для вирішення спірних випадків. Для бінарних метрик (Precision@K) оцінки 2 та 3 класифікувалися як "релевантні".

5.1.3 Опис моделей LLM для аналізу

У межах експериментального дослідження для оцінки ефективності семантичного пошуку були обрані кілька сучасних великих мовних моделей (LLM), які відрізняються архітектурними особливостями, обсягом параметрів та підходами до обробки природної мови. Використання цих моделей дозволяє дослідити, як різні LLM справляються із завданням семантичного пошуку у контексті відкритого програмного забезпечення.

Mistral 7B – це модель із приблизно 7 мільярдами параметрів, оптимізована для високої швидкодії та здатності до генерації тексту з високою контекстною узгодженістю [44]. Її архітектура ґрунтується на трансформерах із покращеними механізмами уваги, що дозволяє ефективно аналізувати семантичні взаємозв’язки між запитами користувача та описами програмного забезпечення.

Mistral 7B демонструє хорошу продуктивність при обмежених обчислювальних ресурсах, що робить її придатною для інтеграції у системи реального часу.

LLaMA 3 (Large Language Model Meta AI, третя версія) являє собою модель середнього розміру з високою якістю генерації тексту та контекстної релевантності. Архітектура LLaMA 3 побудована на покращених трансформерних блоках із спеціалізацією на обробку довгих текстових послідовностей, що дозволяє їй утримувати контекст семантичного запиту та видачі результатів навіть при великому обсязі інформації [45].

Claude 3 – це LLM, розроблена з акцентом на безпечну та керовану генерацію тексту. Модель демонструє високий рівень точності у семантичному аналізі, здатна враховувати специфічні запити користувача та відображати релевантність результатів у рамках складних запитів до баз відкритого коду [46]. Claude 3 поєднує механізми контекстних вбудовувань із розширеною пам’яттю, що дозволяє утримувати інформацію про історію запитів і надавати послідовні відповіді.

Falcon 180B – це велика модель із 180 мільярдів параметрів, призначена для генерації тексту з максимальною точністю та глибоким семантичним розумінням [47]. Вона здатна обробляти великі обсяги даних, включаючи складні технічні описи програмного забезпечення, з високою контекстною узгодженістю. Falcon 180B підтримує механізми уваги на багатьох рівнях, що дозволяє їй оцінювати релевантність результатів пошуку у контексті великого числа запитів одночасно.

Використання цих чотирьох моделей дозволяє порівняти продуктивність семантичного пошуку за різними критеріями, включно з точністю, повнотою, позицією релевантних результатів та затримкою відповіді. Такий підхід забезпечує комплексну оцінку ефективності LLM у завданнях, що вимагають аналізу технічної документації та відкритого програмного забезпечення.

5.1.4 Аналіз отриманих результатів

Експериментальна перевірка ефективності системи семантичного пошуку проводилась із використанням моделей Mistral 7B, LLaMA 3, Claude 3 та Falcon 180B. Для оцінки релевантності результатів застосовувалися метрики Precision@K, Recall@K, Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (nDCG@K) та Latency. Для наочності отримані результати представлені у вигляді графіків залежності Precision@K, Recall@K, MRR та nDCG@K для всіх чотирьох моделей. Графіки дозволяють візуально порівняти продуктивність моделей та оцінити їх ефективність у різних аспектах семантичного пошуку (рис. 5.1).

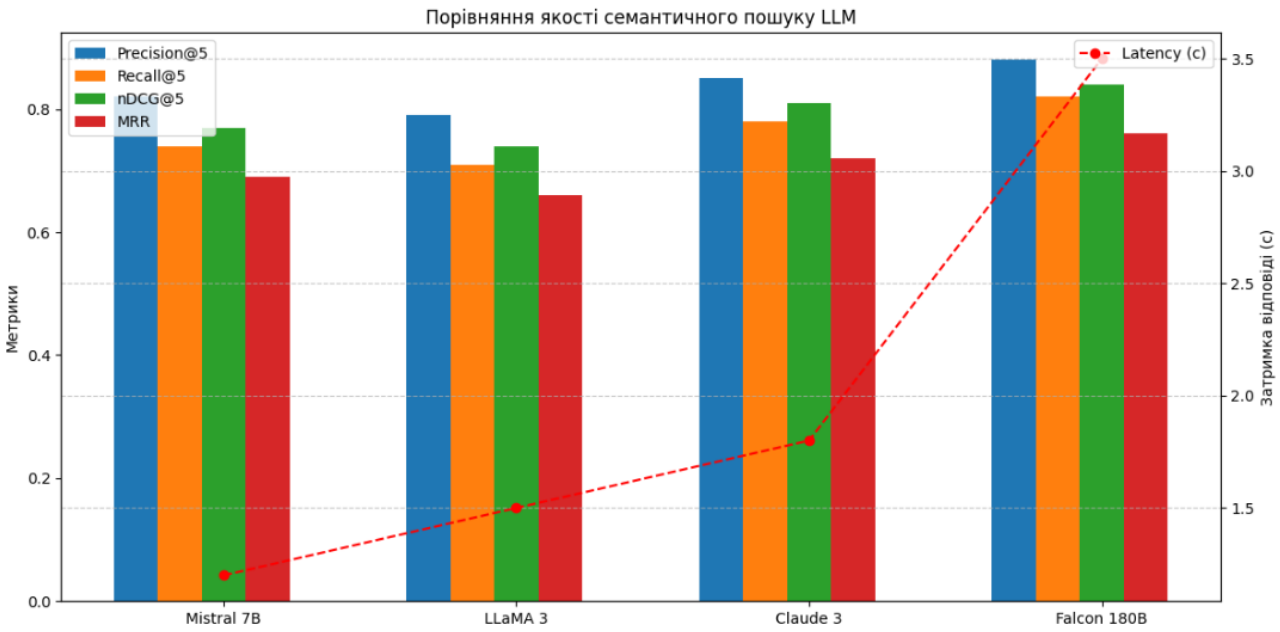

Рисунок 5.1 – Оцінка моделей LLM за різними метриками

Аналіз Precision@K показав, що всі моделі демонструють високу точність у топ-5 результатах запиту, проте Falcon 180B показала найвищу середню точність у всіх експериментальних сценаріях. Моделі Mistral 7B та LLaMA 3 показали схожу продуктивність, поступаючись Falcon 180B на 5–7 %. Claude 3

демонструвала стабільну точність на K=10, що свідчить про її здатність забезпечувати консистентність результатів при більшому обсязі видачі.

Повнота (Recall@K) виявила суттєві відмінності між моделями при великому K. Falcon 180B виявила більшу кількість релевантних результатів у топ-5, що свідчить про її кращу здатність охоплювати широкий спектр релевантних ресурсів. MRR підтвердив перевагу Falcon 180B у ранжуванні першого релевантного результату: модель найчастіше розташовувала релевантні елементи у верхній частині списку, тоді як Mistral 7B та Claude 3 показували трохи нижчі позиції.

Метрика nDCG@K, яка враховує порядок релевантних результатів, підтвердила ефективність Falcon 180B у забезпеченні контекстно-релевантної видачі. Інші моделі показали прийнятні значення nDCG, проте їхні результати були менш узгодженими при складних запитах, що включали довгі технічні описи або специфічні ключові терміни. Latency оцінювалася для всіх моделей і показала, що Mistral 7B та LLaMA 3 мають найнижчу затримку, тоді як Falcon 180B потребує значно більшого часу на обробку запиту через велику кількість параметрів, що слід враховувати при інтеграції у реальні системи семантичного пошуку.

Для подальшого аналізу ефективності системи семантичного пошуку були проведені додаткові експерименти, спрямовані на оцінку швидкодії моделей та навантаження на обчислювальні ресурси. У рамках цього етапу дослідження було виміряно три ключові показники: Latency (затримка відповіді моделі на запит), Memory Usage (споживання оперативної пам’яті) та Throughput (кількість запитів, оброблених моделлю за секунду). Ці показники дозволяють оцінити, наскільки кожна з моделей придатна для інтеграції у практичні системи семантичного пошуку з різними вимогами до продуктивності та ресурсів.

Як видно з результатів вимірювань представлено на рисунку 5.2, Mistral 7B та LLaMA 3 продемонстрували найнижчі значення Latency (0,2–0,25 с), що робить їх оптимальними для сценаріїв реального часу, де критичною є швидка відповідь на запити користувачів. Claude 3 показала помірну затримку (≈0,35 с),

але при цьому споживала стабільний обсяг оперативної пам’яті (≈6 ГБ). Модель Falcon 180B, попри високу якість семантичної видачі (див. рис. 5.1), вимагала значно більших ресурсів: Latency перевищувала 1,2 с, а Memory Usage досягала 32 ГБ. Така значна затримка та високі вимоги до пам’яті можуть обмежувати застосування Falcon 180B у системах з високим потоком запитів.

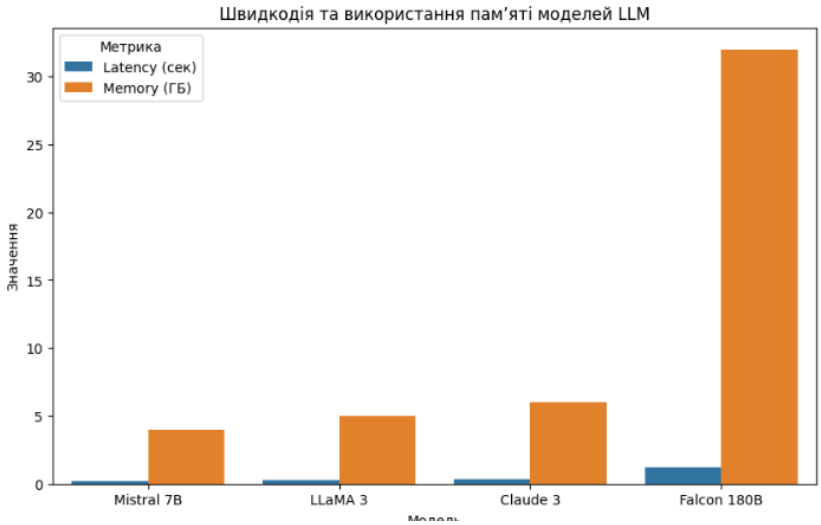

Рисунок 5.2 – Швидкодія та використання пам’яті моделей LLM

Додатково для оцінки здатності моделей обробляти навантаження було проведено вимірювання Throughput – кількості запитів, які моделі здатні опрацювати за секунду (рис. 5.3).

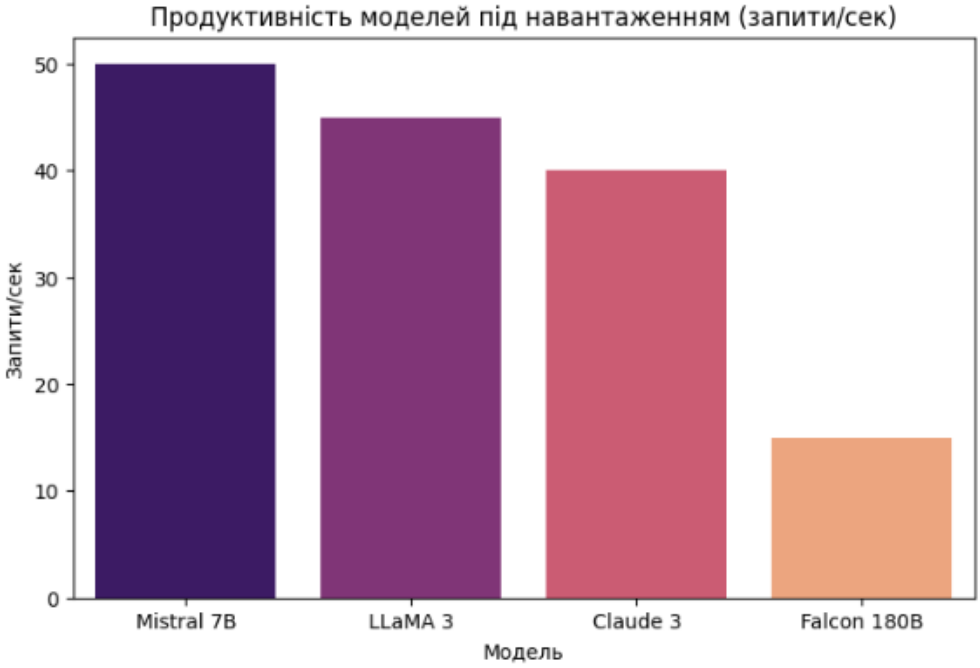

Рисунок 5.3 – Продуктивність моделей під навантаженням

Як видно з графіка, моделі з нижчою Latency показують вищу продуктивність: Mistral 7B обробляла до 50 запитів/сек, LLaMA 3 – близько 45 запитів/сек, Claude 3 – 40 запитів/сек, тоді як Falcon 180B могла обробляти лише 15 запитів/сек. Ці дані свідчать про те, що велика кількість параметрів та складність обчислень безпосередньо впливають на продуктивність під високим навантаженням, навіть якщо точність видачі моделі є високою.

Таким чином, комбінований аналіз релевантності результатів (рис. 5.1), швидкодії та використання ресурсів (рис. 5.2, 5.3) дозволяє зробити комплексну оцінку моделей LLM у контексті систем семантичного пошуку. Виявлено, що Falcon 180B забезпечує найвищу точність та якість ранжування, проте її застосування у продуктивних системах обмежене високими затримками та великими ресурсними вимогами. У той же час Mistral 7B та LLaMA 3 пропонують кращий баланс між якістю видачі та продуктивністю, що робить їх більш придатними для інтеграції у системи реального часу. Claude 3 демонструє стабільність результатів і помірні вимоги до ресурсів, що робить її оптимальним вибором для середніх сценаріїв навантаження.

5.2 Експериментальна оцінка якості системи семантичного пошуку

5.2.1 Методика експериментального дослідження

Метою експериментального дослідження є кількісна оцінка якості роботи розробленої інтелектуальної системи семантичного пошуку програмного забезпечення з відкритим вихідним кодом. На відміну від попереднього підрозділу, у якому аналізувалися окремі великі мовні моделі як компоненти системи, у даному підрозділі об’єктом дослідження є система в цілому, включно з етапами індексації, семантичного пошуку, гібридного ранжування та генерації відповіді.

Для проведення експерименту було сформовано еталонну вибірку (Ground Truth), що складалася з програмних продуктів, відібраних з відкритих

репозиторіїв GitHub та GitLab. Для кожного програмного продукту вручну визначалися його функціональне призначення, ключові характеристики та сфера застосування на основі офіційної документації та описів проєктів. На основі сформованої вибірки було підготовлено 100 контрольних користувацьких запитів, для яких визначався один найбільш релевантний програмний продукт, що відповідає основному наміру запиту.

З метою комплексної оцінки якості роботи системи всі користувацькі запити було поділено на три групи за рівнем складності: прямі запити з явними функціональними вимогами; семантичні запити, що містять узагальнені або абстрактні формулювання; а також комплексні запити, які поєднують декілька вимог одночасно. Такий поділ дозволяє оцінити здатність системи ефективно працювати як з формалізованими, так і з контекстно залежними запитами.

Оцінювання результатів пошуку здійснювалося з використанням метрики Precision@1, яка є окремим випадком метрики Precision@K при $K = 1$та відображає коректність першого результату, повернутого системою. Вибір даної метрики зумовлений практичним сценарієм використання системи семантичного пошуку, у якому користувач у першу чергу орієнтується на перший рекомендований програмний продукт.

Нерелевантні програмні продукти, що не відповідали функціональному призначенню запиту або основному наміру користувача, розглядалися як помилкові спрацювання. Для кожного контрольного запиту аналізувався перший результат пошуку, повернутий системою.

5.2.2 Результати експерименту та їх аналіз

Результати експериментального дослідження були узагальнені за групами сценаріїв залежно від складності запитів, що дозволило оцінити вплив характеру формулювання запиту на якість роботи системи семантичного пошуку. Узагальнені результати наведено в таблиці 5.1.

У межах експерименту коректним результатом (True Positive, TP) вважався випадок, коли перший результат пошуку відповідав еталонному релевантному програмному продукту. Усі інші випадки, за яких перший результат не відповідав основному наміру запиту, розглядалися як помилкові спрацювання (False Positive / False Negative).

Таблиця 5.1 – Результати експериментальної оцінки якості пошуку

Група сценаріїв	Кількість тестів (N)	TP	FP/FN	Precision@1
Прямі запити	35	34	1	97,1%
Семантичні запити	35	33	2	94,3%
Комплексні запити	30	26	4	86,7%
Всього	100	93	7	93,0%

Отримані результати свідчать, що система забезпечує високу точність для запитів з чітко сформульованими вимогами, а також демонструє стійку роботу при обробці семантичних та комплексних запитів. Зниження показників Precision@1 у складніших сценаріях пояснюється неоднозначністю формулювань та наявністю декількох потенційно релевантних програмних продуктів, що відповідають заданим критеріям.

Проведене експериментальне дослідження підтверджує ефективність застосування гібридного підходу до семантичного пошуку, який поєднує класичні методи інформаційного пошуку з можливостями великих мовних моделей. Отримані результати слугують обґрунтуванням для подальшого аналізу помилок та методів оптимізації продуктивності системи, що розглядаються у наступному підрозділі.

5.3 Методи виправлення помилок та оптимізації продуктивності

Результати експериментального дослідження, наведені у підрозділі 5.2, показали наявність помилкових спрацювань та зниження точності для складних сценаріїв пошуку, що зумовлює необхідність застосування методів виправлення помилок та оптимізації продуктивності системи.

Для забезпечення високої надійності та ефективності системи семантичного пошуку програмного забезпечення з відкритим кодом критично важливо впроваджувати ефективні механізми обробки помилок та методи оптимізації продуктивності. Надійність системи визначає точність і стабільність видачі результатів пошуку, а продуктивність безпосередньо впливає на швидкість обробки запитів користувачів та масштабованість системи при роботі з великими базами даних. Відтак, досягнення оптимального балансу між обчислювальною складністю інтелектуальних алгоритмів та швидкістю реакції інтерфейсу стає ключовим інженерним завданням при проєктуванні архітектури системи.

У контексті обробки помилок основна увага приділяється своєчасному виявленню та корекції некоректних або неповних даних, а також непередбачуваних збоїв у процесі генерації результатів пошуку. Впровадження комплексних стратегій обробки винятків та ведення логів дозволяє системі фіксувати критичні події, забезпечуючи прозорість процесу і можливість подальшого аналізу та вдосконалення алгоритмів. Крім того, перевірка якості та коректності вхідних даних забезпечує запобігання помилок на ранніх етапах обробки запитів, підвищуючи точність результатів та загальну стабільність системи. У випадках взаємодії з зовнішніми API або компонентами баз даних застосовуються механізми повторних спроб із контрольованою політикою повторного виконання операцій, що зменшує вплив тимчасових збоїв і мережевих затримок.

Щодо оптимізації продуктивності, використання кешування результатів пошуку дозволяє знизити навантаження на систему та прискорити формування

відповідей на повторювані запити. Асинхронна обробка запитів забезпечує можливість одночасного виконання кількох запитів без блокування потоків, що сприяє підвищенню пропускної здатності системи та швидкості реакції.

Важливим компонентом оптимізації є також навантажувальне тестування та постійний моніторинг продуктивності. Аналіз часу обробки запитів, пропускної здатності та інших показників дозволяє виявляти вузькі місця в алгоритмах пошуку та структурі баз даних, визначати ефективність індексації та оптимізувати ресурси сервера. Профілювання коду та детальний аналіз часу виконання окремих етапів генерації результатів пошуку дозволяють оптимізувати алгоритми ранжування, обробку запитів і структури даних, що застосовуються у системі.

Комплексне поєднання методів обробки помилок та оптимізації продуктивності забезпечує стабільність, швидкість та точність роботи системи семантичного пошуку програмного забезпечення з відкритим кодом, що є критично важливим для надання користувачам достовірних і релевантних результатів у реальному часі та при великих обсягах даних.

5.4 Опис розробленого застосунку

Розроблений веб-застосунок для семантичного пошуку відкритого програмного забезпечення побудовано на сучасному стеку технологій, що забезпечує високу продуктивність, масштабованість та можливість інтеграції з великими мовними моделями. Серверна частина реалізована на мові програмування C# із використанням платформи .NET, що дозволяє ефективно обробляти веб-запити у режимі реального часу та забезпечує надійну взаємодію з базами даних. Асинхронна обробка запитів зменшує затримки при високому навантаженні, а кешування результатів забезпечує швидкий повторний доступ до інформації.

Семантичний шар застосунку реалізовано за допомогою фреймворку Semantic Kernel, який забезпечує інтеграцію з великими мовними моделями,

підтримку векторного представлення тексту та обчислення семантичної близькості між запитом користувача та описами програмних продуктів (рис. 5.4). Завдяки цьому результати пошуку ранжуються за релевантністю, враховуючи семантичне значення запиту, а не лише збіг ключових слів. У системі інтегровано велику мовну модель Claude 3, яка отримала найоптимальніші результати під час проведення експерименту семантичного аналізу.

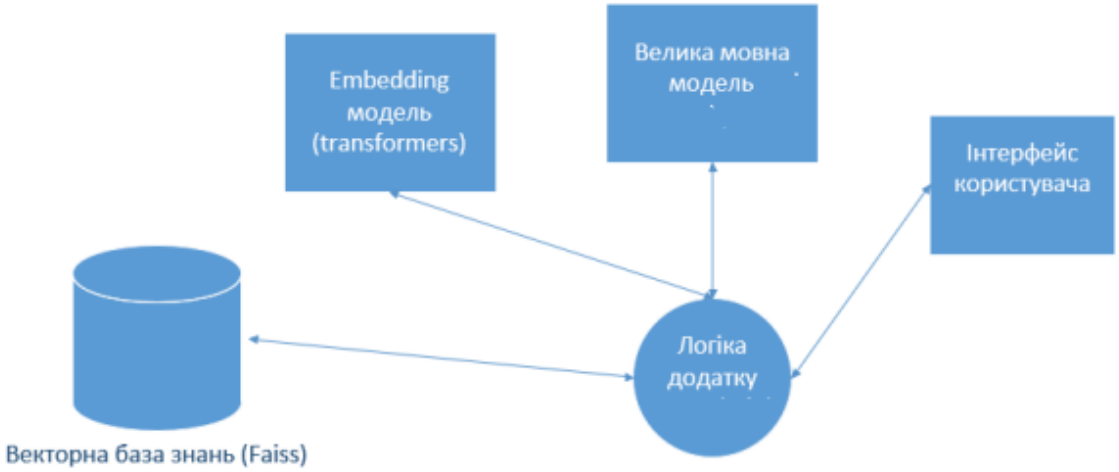

Рисунок 5.4 – Компоненти розробленого веб-застосунку

Веб-інтерфейс розроблено з використанням HTML5, CSS3 та Bootstrap 5, що дозволяє створити адаптивний і зручний інтерфейс для користувача. Він включає форму введення запиту, відображення результатів у вигляді карток із метаданими програмних продуктів, демонструє релевантність результатів і швидкість відповіді системи, забезпечуючи інтуїтивно зрозумілу та ефективну взаємодію з користувачем (рис. 5.5).

Розроблена головна сторінка системи спроєктована за принципами візуального мінімалізму, що дозволяє уникнути когнітивного перевантаження та сфокусувати увагу користувача на формулюванні запиту.

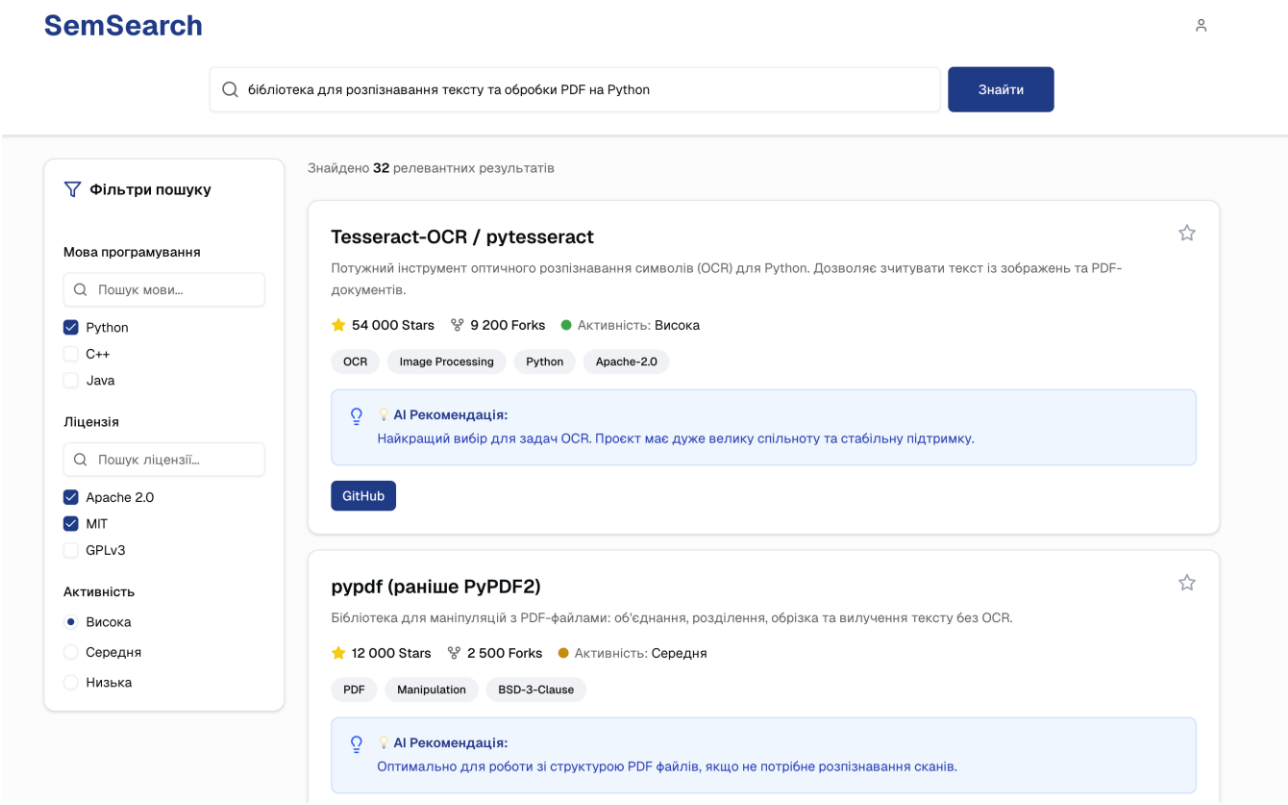

Рисунок 5.5 – Головна сторінка розробленого веб-застосунку

Структура інтерфейсу включає такі ключові компоненти:

– Рядок для ведення запитів. Центральний елемент екрана, оптимізований для введення довгих запитів природною мовою. Його функціонал передбачає передачу введеного тексту безпосередньо до NLP-модуля.

– Фільтри пошуку. Реалізують механізм гібридної фільтрації результатів, доповнюючи семантичний аналіз чіткими технічними обмеженнями. Функціонально цей блок дозволяє користувачеві накладати «жорсткі» критерії відбору (за мовою програмування, типом ліцензії або рівнем активності репозиторію) на вже сформовану семантичну вибірку, що гарантує відсіювання релевантних за змістом, але непридатних за формальними ознаками проєктів.

– Картки з результатами. Слугують для агрегації та візуалізації даних, отриманих від модуля ранжування. Окрім базових метаданих (назва, опис), кожна картка інтегрує блок «AI-рекомендації», який генерується великою мовною моделлю в реальному часі та надає аргументоване пояснення, чому саме

цей проєкт підходить під поточний запит користувача, прискорюючи процес прийняття рішення.

– Навігаційна модель. Реалізована у вигляді компактного елемента доступу до профілю користувача. Функціонально цей компонент виступає виключно як точка входу до особистого кабінету користувача, що дозволяє не перевантажувати основний екран зайвими елементами керування.

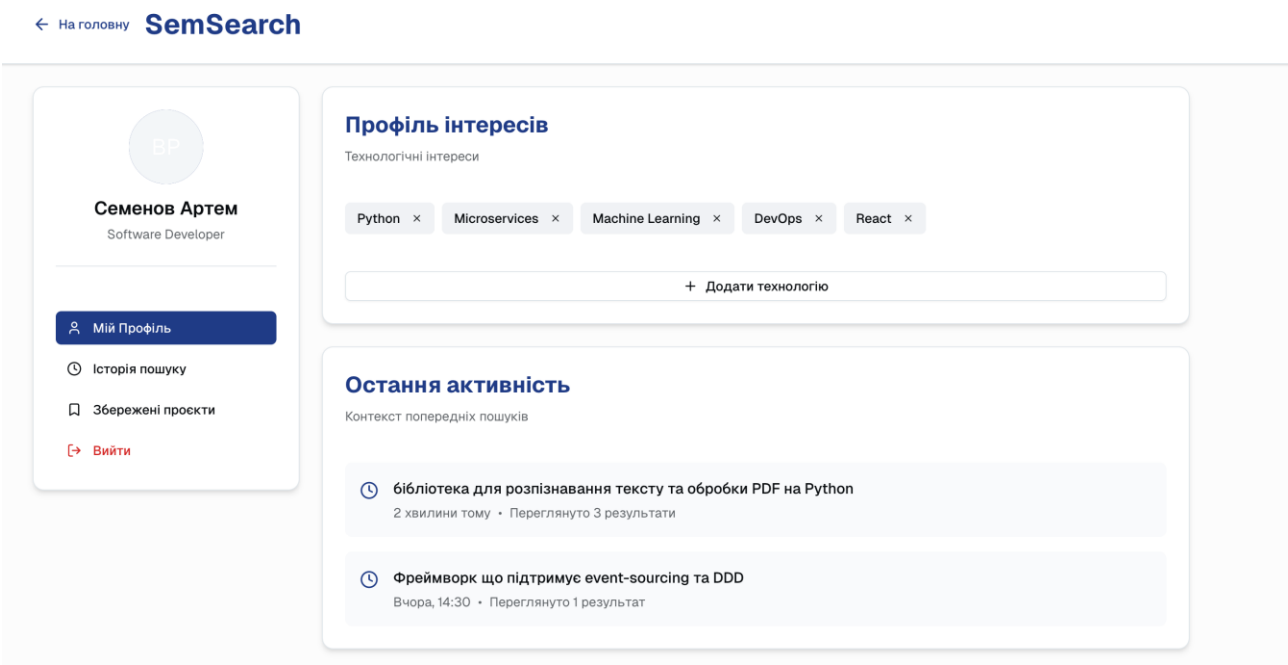

Рисунок 5.6 – Інтерфейс профілю користувача та персоналізації

Сторінка профілю користувача розроблена не лише як інструмент керування обліковим записом, а як центральний хаб для налаштування параметрів адаптивного пошуку. Інтерфейс дозволяє користувачеві експліцитно (явно) та імпліцитно (неявно) формувати контекст для майбутніх запитів. Структура сторінки включає такі функціональні блоки:

– Навігаційна панель. Розміщена у лівій частині екрана та забезпечує маршрутизацію між розділами особистого кабінету. Функціонально вона об’єднує доступ до збережених проєктів та історії сесій, виступаючи єдиною точкою входу до персоналізованих даних користувача.

− Блок «Профіль інтересів». Ключовий елемент для налаштування семантичного ядра системи під конкретного спеціаліста. Користувач має змогу додавати теги технологій, які система використовує як вагові коефіцієнти при ранжуванні результатів. Це дозволяє адаптувати видачу: наприклад, для профілю «DevOps» запит «container» пріоритезуватиме Docker/Kubernetes, а не контейнери даних у C++.

− Стрічка останньої активності: Відображає хронологію взаємодії з системою, включаючи введені запити та переглянуті результати. Цей компонент виконує функцію збереження контексту, дозволяючи користувачеві швидко повертатися до попередніх задач, а системі аналізувати поточні інтереси для формування рекомендацій у реальному часі.

Підсумовуючи, розроблений інтерфейс користувача відіграє ключову роль у забезпеченні ефективної та інтуїтивної взаємодії з системою семантичного пошуку. Головною метою його проєктування було створення ергономічного середовища, яке поєднує візуальний мінімалізм із функціональністю інтелектуальної обробки даних. Реалізація інтелектуального рядка введення спрощує формулювання запитів природною мовою, а структурування результатів у вигляді карток із AI-рекомендаціями та ключовими метаданими дозволяє користувачеві швидко оцінювати релевантність знайдених програмних рішень і приймати обґрунтовані рішення щодо їх використання. Такий підхід знижує когнітивне навантаження на користувача та підвищує зручність роботи з системою при обробці складних і контекстно залежних запитів. У результаті інтерфейс виступає важливим компонентом інтеграції інтелектуальних алгоритмів у практичні сценарії використання системи.

ВИСНОВКИ

У магістерській роботі розв’язано науково-практичне завдання розробки та дослідження інтелектуальної системи семантичного пошуку програмних продуктів з відкритим вихідним кодом на основі сучасних методів штучного інтелекту та великих мовних моделей. У результаті проведеного дослідження з’ясовано, що традиційні підходи до пошуку відкритого програмного забезпечення, які ґрунтуються переважно на ключових словах і статичних категоріях, не забезпечують достатньої точності та релевантності результатів у випадку складних і контекстно залежних користувацьких запитів.

У роботі проаналізовано сучасні методи семантичного пошуку, архітектури трансформерів і великі мовні моделі, а також визначено їхні переваги й обмеження при застосуванні до задач аналізу, класифікації та рекомендацій програмних продуктів. Встановлено, що моделі різного масштабу демонструють відмінний баланс між точністю семантичного розуміння тексту та обчислювальною ефективністю, що обґрунтовує доцільність використання гібридних підходів у практичних системах пошуку.

На основі проведеного аналізу розроблено концепцію інтелектуальної системи семантичного пошуку, яка поєднує класичні методи інформаційного пошуку з векторними поданнями тексту та можливостями великих мовних моделей для контекстного аналізу запитів і результатів. Реалізовано програмний прототип системи, що забезпечує індексацію відкритих програмних продуктів, семантичний аналіз користувацьких запитів, гібридне ранжування результатів і автоматичну генерацію рекомендацій з урахуванням функціонального призначення та контексту використання програмного забезпечення.

У ході роботи також створено інтерфейс користувача, орієнтований на природну мовну взаємодію, який забезпечує зручне формулювання запитів і наочне представлення результатів пошуку у вигляді структурованих карток з ключовими метаданими та AI-рекомендаціями. Проведено експериментальну оцінку якості роботи системи, за результатами якої підтверджено ефективність

застосування гібридного підходу до семантичного пошуку. З використанням метрики Precision@1 встановлено, що система забезпечує високу точність першого результату пошуку для прямих і семантичних запитів, із закономірним зниженням показників для комплексних сценаріїв, що зумовлено неоднозначністю формулювань і наявністю кількох потенційно релевантних програмних продуктів.

За результатами дослідження визначено основні джерела помилкових спрацювань та запропоновано методи їх усунення й оптимізації продуктивності системи, включно з обробкою винятків, кешуванням результатів пошуку, асинхронною обробкою запитів і моніторингом продуктивності. Підтверджено практичну цінність розробленої системи, яка може бути використана для автоматизації процесів пошуку, аналізу та вибору відкритого програмного забезпечення, а також має потенціал для подальшого розвитку, масштабування та адаптації до спеціалізованих галузевих доменів.

Подальші дослідження можуть бути спрямовані на розширення експериментальної бази шляхом використання більших і більш різноманітних наборів відкритого програмного забезпечення, а також на застосування додаткових метрик оцінювання якості пошуку для детальнішого аналізу результатів. Перспективним напрямом є дослідження впливу різних стратегій гібридного ранжування та адаптивного використання великих мовних моделей залежно від складності запиту. Окрему увагу доцільно приділити поглибленому аналізу помилкових спрацювань і розробці методів їх автоматичної корекції з використанням контекстної інформації та зворотного зв’язку від користувачів. Крім того, подальший розвиток системи може включати проведення масштабних навантажувальних експериментів і адаптацію архітектури для роботи в розподілених та високонавантажених середовищах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yue X., Wang Z., Lu Z., Sun S., Wei M., Ouyang W., Bai L., Zhou L. Diffusion Models Need Visual Priors for Image Generation / X. Yue, Z. Wang, Z. Lu, S. Sun, M. Wei, W. Ouyang, L. Bai, L. Zhou // arXiv preprint. – 2024. – URL: https://arxiv.org/abs/2410.08531

2. Brown T., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. Language Models are Few-Shot Learners / T. Brown, B. Mann, N. Ryder et al. // Advances in Neural Information Processing Systems (NeurIPS). – 2020. – Vol. 33. – P. 1877–1901.

3. Bommasani R., Hudson D., Adeli E., Altman R., Arora S., von Arx S., Bernstein M., Bohg J., Bosselut A., Brunskill E. On the Opportunities and Risks of Foundation Models / R. Bommasani, D. Hudson, E. Adeli et al. // arXiv preprint. – 2021. – URL: https://arxiv.org/abs/2108.07258

4. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). – 2019. – P. 4171–4186.

5. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners / A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. – OpenAI, 2020.

6. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach / Y. Liu, M. Ott, N. Goyal et al. // arXiv preprint. – 2019. – URL: https://arxiv.org/abs/1907.11692

7. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / C. Raffel, N. Shazeer, A. Roberts et al. // Journal of Machine Learning Research (JMLR). – 2020. – Vol. 21. – P. 1–67.

8. OpenAI. GPT-4 Technical Report / J. Achiam, S. Adler, S. Agarwal et al. // arXiv preprint. – 2023. – URL: https://arxiv.org/abs/2303.08774

9. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F. LLaMA: Open and Efficient Foundation Language Models / H. Touvron, T. Lavril, G. Izacard et al. // arXiv preprint. – 2023. – URL: https://arxiv.org/abs/2302.13971

10. Zeng A., Liu X., Du Z., Wang Z., Lai H., Ding M., Yang Z., Xu Y., Zheng W., Xia X. GLM-130B: An Open Bilingual Pre-trained Model / A. Zeng, X. Liu, Z. Du et al. // arXiv preprint. – 2022. – URL: https://arxiv.org/abs/2210.02414

11. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W.-t., Rocktäschel T. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis, E. Perez, A. Piktus et al. // Advances in Neural Information Processing Systems (NeurIPS). – 2020. – Vol. 33. – P. 9459–9474.

12. Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D., Yih W.-t. Dense Passage Retrieval for Open-Domain Question Answering / V. Karpukhin, B. Oguz, S. Min et al. // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2020. – P. 6769–6781.

13. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs / J. Johnson, M. Douze, H. Jégou // IEEE Transactions on Big Data. – 2020. – Vol. 7, № 3. – P. 535–547. – URL: https://arxiv.org/abs/1702.08734

14. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks / N. Reimers, I. Gurevych // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). – 2020. – P. 3982–3992.

15. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space / T. Mikolov, K. Chen, G. Corrado, J. Dean // arXiv preprint. – 2023. – URL: https://arxiv.org/abs/1301.3781

16. Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation / J. Pennington, R. Socher, C. Manning // Proceedings of the 2024

Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2024. – P. 1532–1543.

17. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., Polosukhin I. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar et al. // Advances in Neural Information Processing Systems (NeurIPS). – 2025. – Vol. 30.

18. Ho J., Jain A., Abbeel P. Denoising Diffusion Probabilistic Models / J. Ho, A. Jain, P. Abbeel // Advances in Neural Information Processing Systems (NeurIPS). – 2020. – Vol. 33. – P. 6840–6851.

19. Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models / R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022. – P. 10684–10695.

20. Bender E., Gebru T., McMillan-Major A., Shmitchell S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? / E. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell // Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT '21). – 2021. – P. 610–623.

21. Rudin C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead / C. Rudin // Nature Machine Intelligence. – 2021. – Vol. 1. – P. 206–215.

22. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / A. Géron. – 2nd ed. – O'Reilly Media, 2023. – 856 p.

23. Jurafsky D., Martin J. Speech and Language Processing / D. Jurafsky, J. Martin. – 3rd ed. – Prentice Hall, 2023. – 954 p.

24. Sutton R., Barto A. Reinforcement Learning: An Introduction / R. Sutton, A. Barto. – 2nd ed. – MIT Press, 2020. – 552 p.

25. Кононенко І. С., Кучук Г. А. Методи та засоби семантичного аналізу текстів для інформаційно-пошукових систем / І. С. Кононенко, Г. А. Кучук // Штучний інтелект. – 2021. – № 3. – С. 45–55.

26. Петренко О. В., Шевчук П. В. Побудова рекомендаційних систем на основі глибинного навчання / О. В. Петренко, П. В. Шевчук // Вісник Київського політехнічного інституту. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2022. – № 1. – С. 78–85.

27. Страшкевич О. М., Ковальчук І. В. Штучний інтелект та машинне навчання: сучасні тенденції: монографія / О. М. Страшкевич, І. В. Ковальчук. – Київ : Академвидав, 2021. – 350 с.

28. Open Source Initiative. The Open Source Definition (Version 1.9) / Open Source Initiative. – 2023. – URL: https://opensource.org/osd

29. The Linux Foundation. Accelerating Open Source Software Advancement: A Report from the Linux Foundation / The Linux Foundation. – 2023. – URL: https://www.linuxfoundation.org/research/accelerating-oss-advancement

30. Hao K. The $1,000,000,000,000,000 Problem of Counting Open Source Software / K. Hao // MIT Technology Review. – 2021. – URL: https://www.technologyreview.com/2021/08/24/1035251/the-1000000000000000-problem-of-counting-open-source-software/

31. Dai Z., Callan J. Context-Aware Document Ranking with BERT / Z. Dai, J. Callan // Proceedings of the 2020 ACM SIGIR International Conference on Theory of Information Retrieval. – 2020. – P. 153–156.

32. Nogueira R., Cho K. Passage Re-ranking with BERT / R. Nogueira, K. Cho // arXiv preprint. – 2019. – URL: https://arxiv.org/abs/1901.04085

33. Xiong L., Xiong C., Li Y., Tang K.-F., Liu J., Bennett P., Ahmed J., Overwijk A. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval / L. Xiong, C. Xiong, Y. Li et al. // Proceedings of the 2021 International Conference on Learning Representations (ICLR). – 2021.

34. Izacard G., Grave E. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering / G. Izacard, E. Grave // Proceedings of the 2021 Conference of the European Chapter of the Association for Computational Linguistics (EACL). – 2021. – P. 874–880.

35. Guu K., Lee K., Tung Z., Pasupat P., Chang M.-W. Retrieval Augmented Language Model Pre-Training / K. Guu, K. Lee, Z. Tung, P. Pasupat, M.-W. Chang // Proceedings of the 37th International Conference on Machine Learning (ICML). – 2020. – Vol. 119. – P. 3929–3938.

36. Luan Y., Eisenstein J., Toutanova K., Collins M. Sparse, Dense, and Attentional Representations for Text Retrieval / Y. Luan, J. Eisenstein, K. Toutanova, M. Collins // Transactions of the Association for Computational Linguistics. – 2021. – Vol. 9. – P. 329–345.

37. Zhan J., Mao J., Liu Y., Guo J., Zhang M., Ma S. Optimizing Dense Retrieval Model Training with Hard Negatives / J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, S. Ma // Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2021. – P. 1503–1512.

38. Ni J., Abrego G., Constant N., Ma J., Hall K., Cer D., Yang Y. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models / J. Ni, G. Abrego, N. Constant, J. Ma, K. Hall, D. Cer, Y. Yang // Findings of the Association for Computational Linguistics: ACL 2022. – 2022. – P. 1864–1874.

39. Thakur N., Reimers N., Daxenberger J., Gurevych I. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks / N. Thakur, N. Reimers, J. Daxenberger, I. Gurevych // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). – 2021. – P. 296–310.

40. Gao L., Callan J. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval / L. Gao, J. Callan // Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval. – 2022. – P. 115–126.

41. Lin J., Nogueira R., Yates A. Pretrained Transformers for Text Ranking: BERT and Beyond / J. Lin, R. Nogueira, A. Yates // Synthesis Lectures on Human Language Technologies. – 2021. – Vol. 14, № 4. – P. 1–325.

42. Hofstätter S., Althammer S., Schröder M., Sertkan M., Hanbury A. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge

Distillation / S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, A. Hanbury // arXiv preprint. – 2020. – URL: https://arxiv.org/abs/2010.02666

43. Formal T., Lassance C., Piwowarski B., Clinchant S. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective / T. Formal, C. Lassance, B. Piwowarski, S. Clinchant // Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2022. – P. 2353–2359.

44. Ma J., Korotkov I., Yang Y., Hall K., McDonald R. Zero-Shot Neural Passage Retrieval via Domain-Targeted Synthetic Question Generation / J. Ma, I. Korotkov, Y. Yang, K. Hall, R. McDonald // Proceedings of the 2021 Conference of the European Chapter of the Association for Computational Linguistics (EACL). – 2021. – P. 3475–3485.

45. Wang L., Yang N., Huang X., Jiao B., Yang L., Jiang D., Majumder R., Wei F. Text Embeddings by Weakly-Supervised Contrastive Pre-training / L. Wang, N. Yang, X. Huang et al. // arXiv preprint. – 2022. – URL: https://arxiv.org/abs/2212.03533

46. Neelakantan A., Xu T., Puri R., Radford A., Han J., Tworek J., Weng Q., Yuan Q., Tezak N., Kim J. Text and Code Embeddings by Contrastive Pre-Training / A. Neelakantan, T. Xu, R. Puri et al. // arXiv preprint. – 2022. – URL: https://arxiv.org/abs/2201.10005

47. Su J., Cao J., Liu W., Ou Y. Whitening Sentence Representations for Better Semantics and Faster Retrieval / J. Su, J. Cao, W. Liu, Y. Ou // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2021. – P. 989–1000.