НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технології

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

інформаційних систем і технологій

_____ Швиденко М.З.

(підпис) (ПІБ)

“___” _____ 20 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

“Розробка автоматизованої консультативної системи з використанням штучного інтелекту на основі БД інформацфйно доівдкового порталу AgroUA.net”

Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми

_____д.е.н., професор ____ _____ _____Руденський Р.А.__

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____к.е.н._____ _____ ____Саяпін С.П._____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _ Смородько Володимир Павлович

(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформаційних систем і технологій
_____ Швиденко М.З

«08» 01 2025р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Смородько Володимиру Павловичу

Спеціальність 126 – «Інформаційні системи та технології»

1. Тема бакалаврської кваліфікаційної роботи: **«Розробка автоматизованої консультативної системи з використанням штучного інтелекту на основі БД інформацфйно доівдкового порталу AgroUA.net»**

затверджена наказом ректора НУБіП від 16.12.2024 р. № 2246-С

2. Термін подання завершеної роботи на кафедру 25.05.2025 р.

3. Вихідні дані до бакалаврської кваліфікаційної роботи: Веб-платформа електронного дорадництва з складовою інформаційно-довідкового порталу AgroUA.net.

4. Перелік питань, що розглядаються:

1. Огляд та аналіз предметної області.
2. Інформаційне забезпечення інформаційно-довідкового порталу AgroUA.net як інформаційної бази для консультативної системи з використанням штучного інтелекту
3. Архітектури та алгоритмів для обробки природної мови (NLP) з урахуванням аграрної термінології є інноваційним підходом, адаптованим до української мови
4. Прикладне програмне забезпечення
4. Тестування та вимоги системи

5. Висновки

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Визначення теми, постановка завдання, збір теоретичного матеріалу.	20.02-05.03	Визначення мети, завдань
2	Аналіз існуючої архітектури і алгоритмів	06.03-15.03	1 розділ
3	Моделювання вдосконаленої системи обліку, створення архітектури і логіки взаємодії	16.03-05.04	2 розділ
4	Розробка прототипу	06.04-30.04	3 розділ
5	Підготовка остаточного тексту дипломної роботи	01.05-15.05	Завершення, перевірка

Керівник дипломного проекту _____ ____Саяпін С.П.___

Завдання прийняв до виконання _____ ____Смородько В.П.

Дата видачі завдання “08“ січня 2025 р.

АНОТАЦІЯ

Дипломна робота присвячена розробці автоматизованої консультативної системи для аграрного сектору на основі методів штучного інтелекту. Досліджено сучасні підходи до створення консультативних систем. Новизна роботи полягає у розробці архітектури та алгоритмів для автоматизованої консультативної системи, що спеціалізується на обробці запитів природною українською мовою в аграрному секторі, з глибокою інтеграцією та аналізом контенту аграгаційного порталу. Визначено особливості використання обробки природної мови та машинного навчання для аналізу специфічної української аграрної термінології, включаючи донавчання моделей для її коректного розпізнавання. Спроектовано структуру бази знань, інтегрованої з даними профільного порталу, та розроблено алгоритми для обробки запитів користувачів. Результати роботи можуть бути використані для надання аграріям оперативних та обґрунтованих рекомендацій, сприяючи підвищенню ефективності аграрного виробництва.

ANNOTATION

The diploma thesis is devoted to the development of an automated advisory system for the agricultural sector based on artificial intelligence methods. Modern approaches to creating advisory systems have been investigated. The novelty of this work lies in the development of an architecture and algorithms for an automated advisory system specializing in processing natural language queries in Ukrainian within the agricultural sector, featuring deep integration and content analysis from an agricultural aggregation portal. The features of using natural language processing and machine learning for the analysis of specific Ukrainian agricultural terminology are defined, including the fine-tuning of models for its correct recognition. The structure of a knowledge base, integrated with data from the profile portal, has been designed, and algorithms for processing user requests have been developed. The results of this work can be used to provide farmers with prompt and substantiated recommendations, contributing to the increased efficiency of agricultural production.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Annoy: Бібліотека для наближеного пошуку найближчих сусідів

API: Інтерфейс програмування додатків (Application Programming Interface)

BERT: Двонаправлені кодовані представлення трансформерів (Bidirectional Encoder Representations from Transformers)

FAISS: Бібліотека Facebook AI для пошуку подібностей (Facebook AI Similarity Search)

GRU: Керовані рекурентні блоки (Gated Recurrent Unit)

GUI: Графічний інтерфейс користувача (Graphical User Interface)

IE: Вилучення інформації (Information Extraction)

JSON: JavaScript Object Notation

LSTM: Довга короткочасна пам'ять (Long Short-Term Memory)

ML: Машинне навчання (Machine Learning)

NER: Розпізнавання іменованих сутностей (Named Entity Recognition)

NLP: Обробка природної мови (Natural Language Processing)

QA: Відповіді на питання (Question Answering)

RNN: Рекурентні нейронні мережі (Recurrent Neural Networks)

SQL: Мова структурованих запитів (Structured Query Language)

SQuAD: Стенфордський набір даних для відповіді на питання (Stanford Question Answering Dataset)

SVM: Метод опорних векторів (Support Vector Machines)

TF-IDF: Частота терміну – обернена частота документа (Term Frequency-Inverse Document Frequency)

XML: Extensible Markup Language

AI (AI): Штучний інтелект (Artificial Intelligence)

СУБД: Система управління базами даних

ЗМІСТ

1 ТЕОРЕТИЧНІ ЗАСАДИ ВИКОРИСТАННЯ NLP ТА МАШИННОГО НАВЧАННЯ ДЛЯ АГРАРНОЇ КОНСУЛЬТАТИВНОЇ СИСТЕМИ

1.1 Сучасні підходи до розробки консультативних систем на основі штучного інтелекту: огляд та аналіз

У сучасному світі, де інформація стає дедалі доступнішою, а процеси – складнішими, зростає потреба в ефективних інструментах підтримки прийняття рішень.

Консультативні системи на основі штучного інтелекту (ШІ) набувають все більшої значущості, особливо в таких складних та багатофакторних галузях, як аграрний сектор. Їх ключова роль полягає у наданні фермерам, агрономам та іншим фахівцям персоналізованих порад та рекомендацій, заснованих на аналізі великих обсягів даних та експертних знаннях. В аграрному секторі застосування ШІ дозволяє створювати системи, здатні аналізувати дані про стан ґрунтів, погодні умови, фітосанітарний стан посівів, врожайність попередніх періодів та ринкові тенденції. На основі цього аналізу такі системи можуть надавати фермерам оптимальні рекомендації щодо вибору сортів, строків сівби, норм внесення добрив, стратегій захисту рослин, а також прогнозувати потенційні ризики та допомагати в оптимізації збуту продукції [13, 14, 36, 37, 38].

Під консультативною системою розуміють інтелектуальну систему, здатну генерувати рекомендації, пропозиції або рішення на основі аналізу даних та знань предметної області, імітуючи процес консультування з експертом. У випадку, коли консультативна система використовує методи та алгоритми штучного інтелекту (ШІ) для обробки інформації, навчання та виведення висновків, її визначають як консультативну систему на основі ШІ. Застосування ШІ у розробці консультативних систем набуває все більшої актуальності, оскільки дозволяє автоматизувати складні процеси аналізу даних, виявляти приховані закономірності та адаптуватися до змінних умов [6, 10, 18, 22]. ШІ

надає можливість консультативним системам виходити за рамки простих правил та алгоритмів, забезпечуючи більш гнучкий та інтелектуальний підхід до вирішення проблем. Наприклад, в аграрному секторі застосування ШІ дозволяє створювати консультативні системи, здатні аналізувати великі обсяги даних про стан ґрунтів, погодні умови, врожайність та ринкові тенденції, щоб надавати фермерам оптимальні рекомендації щодо вирощування культур, боротьби зі шкідниками та збуту продукції [13, 14, 36, 37, 38]. Обґрунтуванням актуальності використання штучного інтелекту для розробки консультативних систем є можливість значного підвищення ефективності та точності прийняття рішень, особливо в умовах невизначеності та обмеженості ресурсів [15, 21, 39, 48, 49].

Розробка консультативних систем на основі штучного інтелекту передбачає використання різноманітних підходів, кожен з яких має свої особливості, переваги та недоліки. Одним з найбільш традиційних підходів є використання систем, заснованих на правилах. Принцип роботи таких систем полягає у формулюванні експертних знань у вигляді набору правил "ЯКЩО-ТО", де "ЯКЩО" представляє собою умову, а "ТО" – дію або висновок. Архітектура системи складається з бази правил, механізму виведення висновків (inference engine), який застосовує правила до вхідних даних, та інтерфейсу користувача. Перевагами систем на правилах є їх прозорість, простота реалізації та можливість експертної інтерпретації логіки прийняття рішень. Проте, вони мають обмеження у складності оновлення та масштабування, особливо коли кількість правил стає великою, а також чутливі до неповних або суперечливих даних [2, 30].

Інший підхід – системи, засновані на знаннях, які представляють знання про предметну область у більш структурованій формі, часто використовуючи онтології або семантичні мережі для опису сутностей, їх властивостей та взаємозв'язків [19, 27, 34, 42]. Механізми виведення висновків в таких системах використовують логічний висновок для генерації нових знань та відповідей на запити користувачів. Перевагою систем на знаннях є можливість представлення складних зв'язків між даними та гнучкість у використанні різних джерел

інформації. Недоліком є складність розробки та підтримки знаннєвої бази, яка вимагає значних зусиль експертів та інженерів знань. Прикладом застосування є система підтримки прийняття рішень в агропромисловому секторі, яка використовує онтологію для представлення знань про сільськогосподарські культури, технології вирощування та кліматичні умови [2, 14, 19].

Сучасні консультативні системи все частіше використовують методи машинного навчання, що дозволяє їм навчатися на великих обсягах даних та адаптуватися до змінних умов. Навчання з учителем (Supervised Learning) передбачає використання алгоритмів класифікації, регресії та ранжування. Алгоритми класифікації, такі як Support Vector Machines, Decision Trees, Random Forests та Logistic Regression [11, 48], використовуються для віднесення об'єктів до певних категорій. Регресія використовується для прогнозування числових значень, а ранжування – для впорядкування об'єктів за їх релевантністю. Навчання без учителя (Unsupervised Learning) застосовується для виявлення прихованих закономірностей у даних, використовуючи методи кластеризації (наприклад, K-means) та зменшення розмірності (наприклад, PCA). Навчання з підкріпленням (Reinforcement Learning) дозволяє консультативній системі навчатися, взаємодіючи з середовищем та отримуючи винагороди за правильні дії. Перевагами систем на основі машинного навчання є їх адаптивність, здатність до навчання на великих обсягах даних та автоматизація процесу розробки [10, 15, 21, 22, 26, 31, 35, 39, 45, 48, 49]. Однак, такі системи часто є "чорними скриньками", тобто їх логіка прийняття рішень не завжди зрозуміла, і вони вимагають великої кількості даних для навчання. Прикладом застосування є система прогнозування врожайності сільськогосподарських культур на основі аналізу супутникових знімків та метеорологічних даних [21, 39, 48, 49].

Перспективним напрямком є розробка гібридних систем, які комбінують різні підходи, наприклад, системи на правилах і машинного навчання [2]. Комбінування систем на правилах для експертних знань з машинним навчанням для обробки великих даних дозволяє поєднати сильні сторони обох підходів. Перевагою є можливість створення більш потужних та гнучких консультативних

систем. Недоліком є складність розробки та інтеграції різних компонентів. Прикладом застосування є система підтримки прийняття рішень в медицині, яка використовує правила для опису медичних протоколів та алгоритми машинного навчання для аналізу медичних зображень та прогнозування ризиків захворювань.

Ефективність різних підходів до розробки консультативних систем на основі штучного інтелекту суттєво варіюється залежно від контексту їх застосування та вимог до системи. Системи, засновані на правилах, відзначаються простотою реалізації та високим рівнем пояснюваності, оскільки логіка їх роботи чітко визначена набором правил [2]. Однак, їх точність може бути обмежена, а вартість підтримки та оновлення зростає експоненційно зі збільшенням кількості правил. Вони вимагають значних експертних знань для формування повної та несуперечливої бази правил. Системи, засновані на знаннях, демонструють більшу гнучкість у представленні складних залежностей між даними, але їх розробка та підтримка є більш складними та ресурсомісткими [19, 27, 34, 42]. Так само, для побудови якісної системи потрібні експертні знання у відповідній галузі.

Системи на основі машинного навчання пропонують високу адаптивність та здатність до навчання на великих обсягах даних, що робить їх привабливими для вирішення складних завдань прогнозування та класифікації [10, 15, 21, 22, 26, 31, 35, 39, 45, 48, 49]. Проте, вони часто є "чорними скриньками", що ускладнює інтерпретацію їх рішень, а їх точність безпосередньо залежить від обсягу та якості навчальних даних. Хоча потреба в експертних знаннях для побудови таких систем є меншою, ніж для систем на правилах, все ж потрібне розуміння принципів машинного навчання та правильний вибір алгоритмів.

Компроміси між цими характеристиками є ключовими при виборі підходу до розробки консультативної системи. Наприклад, якщо важлива прозорість та можливість експертної перевірки логіки системи, то системи на правилах або гібридні системи, що поєднують правила з машинним навчанням, будуть кращим вибором. Якщо ж точність та здатність до адаптації є пріоритетними, а

пояснюваність не є критичною, то системи на основі машинного навчання можуть бути більш ефективними [2, 30].

З огляду на це, системи, засновані на правилах, найбільш ефективні в областях, де знання чітко структуровані та не потребують частого оновлення, наприклад, у сфері медичної діагностики або юридичного консультування. Системи на основі знань добре підходять для областей, де необхідно представляти складні взаємозв'язки між різними сутностями, наприклад, в агропромисловому секторі для управління сільськогосподарськими процесами. Системи на основі машинного навчання особливо ефективні в областях, де є великі обсяги даних, які потребують аналізу, наприклад, для прогнозування врожайності сільськогосподарських культур або виявлення шахрайських операцій у фінансовій сфері [14, 16, 19, 26, 33, 37, 38, 45, 49]. Гібридні системи дозволяють поєднати переваги різних підходів, забезпечуючи баланс між точністю, поясненістю та адаптивністю, і можуть бути застосовані в широкому спектрі областей, де потрібна комплексна підтримка прийняття рішень.

Сучасні тенденції у розробці консультативних систем на основі штучного інтелекту визначаються прагненням до створення більш інтелектуальних, адаптивних та людино-орієнтованих систем. Однією з ключових тенденцій є активне використання глибокого навчання, що дозволяє консультативним системам обробляти складні та неструктуровані дані, такі як текст, зображення та аудіо [6, 21, 39, 48, 49]. Застосування обробки природної мови (NLP) є ключовим для створення інтуїтивних консультативних систем, що дозволяє їм розуміти запити користувачів та генерувати відповіді природною мовою. Роботи Бондаренка О.М. [3] та Кравченка Ю.М. [20] демонструють важливість використання NLP для аналізу саме аграрних текстів, зокрема, для вилучення термінології. Хоча ці дослідження зосереджені на аналізі, а не на генерації консультацій, вони підкреслюють специфіку аграрної лексики, що було враховано при виборі методів токенізації та лематизації в нашій системі. Загальні підходи до NLP, описані в навчальному посібнику під ред. Петрової О.О. [27],

лягли в основу розробки етапів попередньої обробки тексту. Однак, більшість загальних NLP-інструментів, як зазначають Романенко П.Л. [34] та Мельниченко К.Ю. [25], потребують адаптації для ефективної роботи зі специфічними доменами, такими як аграрний, що підтвердило необхідність донавчання моделей у нашому дослідженні. Роботи Олійника С.П. [28] та Остапенко Н.І. [29] щодо розробки чат-ботів та автоматизації агроконсалтингу, хоч і стосуються дещо інших архітектур, вказують на високий попит на подібні системи та підкреслюють актуальність обраної теми.

Персоніфікація стає все більш важливою, дозволяючи адаптувати рекомендації та поради до індивідуальних потреб та вподобань кожного користувача [41, 47]. Крім того, зростає інтерес до пояснюваного ШІ (Explainable AI, XAI), що передбачає розробку методів, які дозволяють пояснювати логіку прийняття рішень системами штучного інтелекту, підвищуючи довіру користувачів та забезпечуючи можливість контролю над процесом консультування [4, 32, 45].

Перспективи подальшого розвитку консультативних систем на основі ШІ пов'язані з інтеграцією нових технологій та розширенням можливостей систем. Одним з перспективних напрямків є розробка автономних консультативних систем, здатних самостійно приймати рішення та діяти в умовах невизначеності, без постійного втручання людини [2]. Інтеграція з IoT-пристроями (Інтернетом речей) дозволить консультативним системам отримувати дані в реальному часі з різних джерел, наприклад, сенсорів та датчиків, що забезпечить більш точний та оперативний аналіз ситуації та надання більш релевантних рекомендацій [10, 18, 19, 35]. Використання блокчейну може забезпечити безпеку та прозорість даних, що особливо важливо для консультативних систем, які працюють з конфіденційною інформацією. Крім того, блокчейн дозволяє створити децентралізовані консультативні системи, де знання та дані розподілені між багатьма учасниками, що підвищує їх стійкість та надійність [16, 18, 36, 37]. Розвиток консультативних систем на основі ШІ відкриває нові можливості для

автоматизації та оптимізації процесів у різних сферах діяльності, підвищення якості прийняття рішень та покращення досвіду користувачів.

Ринок та наукова сфера пропонують низку рішень для підтримки прийняття рішень в аграрному секторі. Розглянемо деякі з них для визначення поточного стану та виявлення ніш для вдосконалення.

Cropio (міжнародна) – комплексна платформа для управління агропідприємством, що використовує супутникові знімки, дані з техніки та метеорологічні дані. Надає рекомендації щодо моніторингу стану посівів, точного землеробства. Переваги: великий набір функцій, інтеграція з багатьма джерелами даних. Обмеження: висока вартість, може бути складною для малих господарств, менший акцент на консультаціях природною мовою.

FieldView (Bayer, міжнародна) – платформа для збору та аналізу даних з полів, допомагає в оптимізації сівби, внесення добрив та ЗЗР. Переваги: сильна інтеграція з продуктами компанії Bayer, детальний аналіз даних. Обмеження: фокус на конкретних продуктах, консультації переважно через аналітичні звіти.

AgroOnline (Україна) – платформа для управління земельним банком, планування операцій, моніторингу техніки. Переваги: адаптована до українських реалій, облік специфіки місцевого законодавства. Обмеження: менший акцент на ШІ-консультаціях щодо агротехнологій, ніж на управлінських аспектах.

Наукові розробки – існують численні дослідницькі проекти, що фокусуються на окремих задачах: діагностика хвороб за фото (з використанням CNN), прогнозування врожайності (ML-регресія), рекомендації щодо добрив на основі аналізу ґрунту. Переваги: глибоке опрацювання конкретних проблем. Обмеження: часто є прототипами, не мають комплексного характеру або зручного інтерфейсу для кінцевого користувача, обмежена доступність.

Розроблювана автоматизована консультативна система має на меті зайняти нішу, яка поєднує:

1.Глибоку інтеграцію з українськомовним контентом профільних порталів. Використання цього унікального та багатого джерела знань, адаптованого до потреб українських аграріїв.

2.Застосування сучасних NLP-моделей, донавчених на аграрній термінології, для забезпечення інтуїтивної взаємодії.

3. На відміну від дорогих комерційних платформ, система орієнтована на надання оперативних та обґрунтованих рекомендацій для різних категорій аграріїв.

4. Поєднання автоматичного вилучення інформації з профільних агрегаторів з можливістю інтеграції локальних знань користувача

5. На відміну від систем, що переважно орієнтовані на управління чи моніторинг, ключовою функцією є надання відповідей та рекомендацій на конкретні запити.

Запропонована нами система спрямована на створення спеціалізованого інструменту, який використовує сильні сторони сучасних ШІ-технологій для ефективної роботи з українським аграрним контентом.

1.2 Основи обробки природної мови (NLP) та машинного навчання для аналізу аграрної термінології

Аграрна термінологія, як специфічна підмножина лексики, охоплює сукупність термінів, що використовуються в сільському господарстві, рослинництві, тваринництві, агрономії та суміжних галузях [20, 27, 34]. Вона характеризується наявністю великої кількості спеціалізованих понять, абревіатур, професіоналізмів та термінів, запозичених з інших наук, таких як біологія, хімія та екологія. Аналіз аграрної термінології є надзвичайно важливим для різноманітних задач, включаючи автоматичну класифікацію аграрних текстів, інформаційний пошук, машинний переклад, вилучення інформації та розробку інтелектуальних систем підтримки прийняття рішень для аграрного сектору [3, 11, 14, 15, 22, 25, 26, 29, 30, 31, 33, 35, 38, 39, 45, 49]. Ефективний аналіз аграрної термінології сприяє підвищенню точності та релевантності результатів пошуку, покращенню якості машинного перекладу аграрних текстів, а також дозволяє автоматизувати процес вилучення інформації з великих обсягів

аграрної документації, що є критично важливим для підтримки прийняття обґрунтованих рішень в аграрному секторі [2, 6, 16, 19, 37, 48].

Проте, аналіз аграрної термінології стикається з певними проблемами та викликами. Специфіка аграрної мови, зокрема, наявність вузькоспеціалізованих термінів, полісемія (багатозначність) термінів, використання професійного жаргону та велика кількість скорочень, ускладнюють застосування стандартних методів обробки природної мови [20, 25, 27, 34]. Крім того, аграрна термінологія постійно розвивається, з'являються нові терміни та змінюються значення існуючих, що вимагає постійного оновлення та адаптації лінгвістичних ресурсів та алгоритмів. Недостатня кількість якісно розмічених даних для навчання моделей машинного навчання також є значним викликом [11, 26, 31]. Тому для ефективного аналізу аграрної термінології необхідна розробка спеціалізованих методів та інструментів, які враховують специфіку аграрної мови та дозволяють вирішувати вищезазначені проблеми.

Обробка природної мови (NLP) є міждисциплінарною галуззю, що поєднує комп'ютерні науки, лінгвістику та штучний інтелект, і спрямована на розробку методів та алгоритмів, які дозволяють комп'ютерам розуміти, інтерпретувати та генерувати людську мову [27, 34]. Процес обробки тексту в NLP включає кілька базових етапів, кожен з яких відіграє важливу роль у підготовці тексту до подальшого аналізу. Першим етапом є токенізація, яка полягає у поділі тексту на окремі слова або токени, що є базовими одиницями для подальшої обробки. Після токенізації зазвичай виконується видалення стоп-слів, тобто слів, які не несуть значущої інформації (наприклад, прийменники, сполучники, артиклі). Наступним етапом є стеммінг та лематизація, які передбачають зведення слова до його базової форми. Стеммінг – це спрощений процес, який відкидає закінчення слова, тоді як лематизація використовує словники та граматичні правила для визначення леми (словникової форми) слова. Завершальним етапом підготовки тексту є розмітка частин мови (Part-of-Speech Tagging), яка полягає у визначенні граматичної ролі кожного слова в реченні (наприклад, іменник, прикметник, дієслово) [3, 27, 34].

Після підготовки тексту необхідно обрати метод його представлення для подальшого аналізу. Одним з найпростіших методів є "мішок слів" (Bag-of-Words), який представляє текст як набір слів без урахування їх порядку та граматичних зв'язків. Інший метод – TF-IDF (Term Frequency-Inverse Document Frequency) – оцінює важливість слова в документі на основі частоти його появи в документі (Term Frequency) та оберненої частоти його появи в колекції документів (Inverse Document Frequency), дозволяючи виділити слова, які є специфічними для даного документа [3, 27, 34].

Більш сучасними методами представлення слів є векторні представлення слів (Word Embeddings), які відображають слова у вигляді векторів у багатовимірному просторі, де близькість векторів відображає семантичну подібність між словами. Популярними моделями Word Embeddings є Word2Vec, GloVe та FastText [27]. Word2Vec використовує нейронні мережі для навчання векторних представлень слів на основі контексту їх появи в тексті. GloVe використовує матрицю спільної появи слів для навчання векторних представлень. FastText розширює Word2Vec, враховуючи морфологічну структуру слів, що особливо корисно для мов з багатою морфологією та для обробки рідкісних слів.

На основі цих методів та алгоритмів NLP вирішуються різноманітні задачі, такі як класифікація тексту, розпізнавання іменованих сутностей, вилучення ключових слів та тематичне моделювання. Класифікація тексту передбачає віднесення тексту до певної категорії на основі його змісту [11, 25, 42]. Розпізнавання іменованих сутностей (NER) полягає у виявленні та класифікації іменованих сутностей в тексті, таких як імена людей, назви організацій, географічні назви та дати [27]. Вилучення ключових слів передбачає автоматичне виділення найбільш важливих слів та фраз з тексту, які відображають його основний зміст. Тематичне моделювання дозволяє виявити приховані теми та тематичні структури в колекції документів [3, 27, 34]. Ефективне використання цих методів та алгоритмів NLP є ключовим для аналізу

аграрної термінології та вирішення різноманітних задач в аграрному секторі [3, 20, 29, 34].

Машинне навчання (ML) є підрозділом штучного інтелекту, який зосереджується на розробці алгоритмів, що дозволяють комп'ютерам навчатися на даних без явного програмування [11, 26, 31, 39, 45, 49]. Існує кілька типів машинного навчання, кожен з яких має свої особливості та застосування. Навчання з учителем (Supervised Learning) передбачає використання розмічених даних, де для кожного вхідного об'єкта відомий правильний вихідний результат. Алгоритми навчання з учителем використовуються для побудови моделей, які можуть передбачати вихідний результат для нових, невідомих об'єктів. Навчання без учителя (Unsupervised Learning) використовує нерозмічені дані, де для вхідних об'єктів невідомі правильні вихідні результати. Алгоритми навчання без учителя використовуються для виявлення прихованих закономірностей, структур та зв'язків у даних. Навчання з підкріпленням (Reinforcement Learning) – це тип машинного навчання, де агент навчається приймати рішення, взаємодіючи з середовищем та отримуючи винагороди або штрафи за свої дії. Хоча навчання з підкріпленням рідко використовується для безпосереднього аналізу термінології, воно може бути застосоване для оптимізації процесів вилучення інформації або класифікації текстів.

Для аналізу тексту використовуються різноманітні алгоритми машинного навчання. Лінійні моделі, такі як Logistic Regression та Support Vector Machines (SVM), є простими та ефективними алгоритмами класифікації, які добре працюють на даних з великою розмірністю [11, 42]. Дерева рішень (Decision Trees) та ансамблеві методи (Ensemble Methods), такі як Random Forest та Gradient Boosting, є більш складними алгоритмами, які можуть обробляти нелінійні залежності між даними та забезпечують високу точність класифікації. Нейронні мережі (Neural Networks) є потужними моделями машинного навчання, які складаються з багатьох шарів штучних нейронів. Згорткові нейронні мережі (Convolutional Neural Networks, CNN) ефективно використовуються для аналізу текстових даних, особливо для задач класифікації тексту та розпізнавання

іменованих сутностей. Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) та LSTM (Long Short-Term Memory) призначені для обробки послідовностей даних, таких як текст, і здатні враховувати контекст та залежності між словами в реченні. Трансформери (Transformers) є найсучаснішими архітектурами нейронних мереж, які використовують механізми уваги (attention) для ефективної обробки текстових даних та досягають високої точності в різних задачах NLP.

Для оцінки якості моделей машинного навчання використовуються різноманітні метрики. Точність (Accuracy) вимірює частку правильно класифікованих об'єктів серед усіх об'єктів. Повнота (Precision) вимірює частку правильно класифікованих об'єктів певного класу серед усіх об'єктів, які були віднесені до цього класу. Відгук (Recall) вимірює частку правильно класифікованих об'єктів певного класу серед усіх об'єктів, які насправді належать до цього класу. F1-міра (F1-score) є гармонійним середнім між повнотою та відгуком і використовується для оцінки балансу між цими двома метриками. Вибір метрики для оцінки якості моделі залежить від специфіки задачі та відносної важливості повноти та відгуку.

Застосування методів обробки природної мови та машинного навчання для аналізу аграрної термінології вимагає врахування специфіки даної галузі. Аграрна термінологія характеризується наявністю специфічних термінів та абревіатур, які часто відсутні в загальних словниках та лінгвістичних ресурсах [20, 27, 34]. Багатозначність термінів (полісемія) також є поширеною проблемою, коли один і той самий термін може мати різні значення в різних контекстах. Крім того, аграрний сектор характеризується використанням жаргону та професіоналізмів, які можуть бути незрозумілі для широкої аудиторії [3, 20, 27, 34].

Для адаптації методів NLP та ML до особливостей аграрної термінології використовуються різні підходи. Розробка спеціалізованих словників та тезаурусів, що містять аграрні терміни, їх визначення та синоніми, є важливим кроком для підвищення точності аналізу. Застосування методів донавчання

(Fine-tuning) для моделей Word Embeddings та Transformers на аграрних текстах дозволяє адаптувати моделі до специфічної лексики та стилістики аграрної мови, покращуючи їх здатність розуміти та інтерпретувати аграрні тексти. Використання методів розпізнавання іменованих сутностей (NER) для вилучення ключових термінів та понять з аграрних текстів дозволяє автоматично виділяти важливу інформацію та структурувати знання про аграрний сектор [3, 27, 34].

Аналіз аграрної термінології може бути застосований для вирішення різноманітних практичних задач. Автоматична класифікація аграрних документів, наприклад, наукових статей, патентів та звітів, дозволяє швидко та ефективно організовувати великі обсяги інформації та забезпечувати доступ до релевантних документів для аграрних експертів [11, 25, 42]. Вилучення інформації з наукових статей та патентів дозволяє автоматично виділяти ключові факти, результати досліджень та технологічні рішення, сприяючи прискоренню наукових досліджень та інновацій в аграрному секторі. Розробка систем інформаційного пошуку для аграрних експертів забезпечує швидкий та зручний доступ до інформації про аграрні технології, методи вирощування культур, захисту рослин та управління аграрним виробництвом [2, 30, 33, 38]. Застосування методів NLP та ML для аналізу аграрної термінології відкриває нові можливості для автоматизації та оптимізації процесів управління знаннями та підтримки прийняття рішень в аграрному секторі.

Ефективна обробка природної української мови, особливо в такій спеціалізованій галузі, як аграрний сектор, вимагає наявності відповідних лінгвістичних ресурсів та моделей.

Існують великі корпуси української мови, такі як Генеральний регіонально анотований корпус української мови (ГРАК), корпуси проектів типу Brown Corpus Ukrainian, корпус українських текстів з Лейпцизького університету. Ці корпуси є цінними для навчання загальномовних моделей, але можуть не містити достатньої кількості специфічної аграрної лексики.

На жаль, великих, загальнодоступних та якісно анотованих корпусів текстів саме з аграрної тематики українською мовою наразі небагато. Для цілей даної роботи корпус формувався шляхом вилучення даних з профільних порталів.

Існують електронні версії тлумачних, орфографічних, синонімічних словників української мови (наприклад, СУМ-11, СУМ-20 онлайн). Український тезаурус системи WordNet також розвивається. Однак, спеціалізовані аграрні тезауруси та словники з чіткою семантичною розміткою для автоматичної обробки є рідкістю.

Існують попередньо навчені моделі Word2Vec, FastText, GloVe для української мови, навчені на великих загальномовних корпусах.

На платформі Hugging Face доступна низка трансформерних моделей, що підтримують українську мову (наприклад, BERT-base-Ukrainian-cased, ukr-RoBERTa, XLM-R). Ці моделі демонструють високу якість на різних задачах NLP.

Бібліотеки spaCy та Stanza надають попередньо навчені пайплайни для української мови, що включають токенізацію, лематизацію, POS-тегування та NER.

Виклики використання ресурсів для аграрної специфіки:

1. Головним викликом є обмежена кількість готових, структурованих та анотованих даних саме з аграрної тематики українською мовою. Це ускладнює навчання високоточних спеціалізованих моделей "з нуля".

2. Аграрна лексика характеризується наявністю вузькоспеціалізованих термінів, великою кількістю синонімів (народні назви, комерційні назви препаратів), абревіатур (ЗЗР, КАС), полісемією (слово "культура" може означати обробку землі або вид рослини). Загальномовні моделі часто погано справляються з такою лексикою.

3. Для досягнення прийнятної якості необхідно донавчати (fine-tuning) загальномовні моделі на специфічних аграрних текстах. Це вимагає створення або збору відповідних анотованих датасетів.

4. Аграрна сфера швидко розвивається, з'являються нові сорти, препарати, технології, що вимагає постійного оновлення словників та моделей.

5.Багато аграрних термінів можуть мати різні значення залежно від контексту, що створює додаткові труднощі для автоматичного розпізнавання.

Врахування цих викликів є ключовим при розробці NLP-модулів для аграрної консультативної системи. У даній роботі було зроблено спробу частково їх вирішити шляхом формування власного корпусу та адаптації існуючих моделей.

1.3 Особливості створення баз знань для аграрних консультативних систем на основі даних.

В аграрних консультативних системах бази знань відіграють центральну роль, слугуючи фундаментом для надання обґрунтованих та персоналізованих рекомендацій [2, 30, 38]. Якість консультацій безпосередньо залежить від повноти, актуальності та точності знань, що містяться в базі. База знань забезпечує систему інформацією про аграрні культури, технології вирощування, методи захисту рослин, агрометеорологічні умови, економічні показники та інші важливі аспекти аграрного виробництва, дозволяючи системі аналізувати вхідні дані та надавати експертні поради користувачам.

Традиційні підходи до створення баз знань, які часто базуються на ручному кодуванні знань експертами, можуть бути недостатніми для аграрних систем з кількох причин. По-перше, аграрна сфера характеризується великою динамічністю та постійною появою нових даних, технологій та наукових відкриттів [6, 10, 18, 22, 26, 31, 37, 39, 48, 49]. Ручне оновлення бази знань може бути трудомістким та неефективним, що призводить до її швидкого застарівання. По-друге, аграрні дані часто характеризуються різноманітністю форматів, джерел та рівнів структурованості, що ускладнює їх інтеграцію в єдину базу знань. По-третє, знання в аграрній сфері часто є неповними, невизначеними та

залежать від контексту, що вимагає використання спеціалізованих методів представлення та обробки знань.

З огляду на ці виклики, бази знань для аграрних консультативних систем повинні відповідати певним вимогам. Вони мають бути здатними інтегрувати дані з різноманітних джерел, включаючи структуровані бази даних, неструктуровані текстові документи та експертні знання [3, 19, 20, 25, 27, 29, 34]. База знань повинна забезпечувати ефективні механізми видобування, представлення, зберігання та оновлення знань, а також підтримувати обробку невизначеності та нечіткості. Важливою вимогою є можливість адаптації бази знань до специфічних потреб користувачів та підтримка різних рівнів деталізації знань. Крім того, база знань повинна бути інтегрована з іншими компонентами консультативної системи, такими як модулі аналізу даних, машинного навчання та інтерфейсу користувача, для забезпечення безперебійної та ефективної роботи системи.

Специфіка даних, що використовуються в аграрному секторі, вимагає особливого підходу до створення баз знань для консультативних систем [6, 14, 15, 19, 35, 39, 48, 49]. До основних типів аграрних даних належать агрометеорологічні дані, які включають інформацію про температуру, вологість, опади та сонячну радіацію, що безпосередньо впливають на ріст та розвиток рослин. Дані про ґрунти, такі як хімічний склад, структура та вологість, є критично важливими для визначення придатності ґрунту для вирощування певних культур. Інформація про рослини, включаючи сорти, врожайність, хвороби та шкідники, дозволяє оптимізувати технології вирощування та захисту рослин. Дані про тварин, зокрема породи, продуктивність та стан здоров'я, необхідні для ефективного управління тваринницьким господарством. Технології вирощування та управління, включаючи сівозміни, використання добрив та методи захисту рослин, також є важливим джерелом знань для консультативних систем. Економічні дані, такі як ціни на продукцію, витрати на виробництво та ринкові тенденції, дозволяють приймати обґрунтовані рішення щодо вибору культур та методів господарювання. Геопросторові дані,

включаючи супутникові знімки та карти полів, надають інформацію про стан посівів, розподіл ґрунтів та інші важливі параметри території [10, 18, 19, 35].

Аграрні дані характеризуються великим обсягом (Big Data), що зумовлено широким використанням сенсорів, супутників та інших джерел даних [6, 10, 18, 22, 23, 35, 39, 48, 49]. Різноманітність аграрних даних проявляється у різних форматах, джерелах та рівнях структурованості, що ускладнює їх інтеграцію в єдину базу знань. Неповнота та невизначеність даних є поширеною проблемою, оскільки аграрні процеси залежать від багатьох факторів, які не завжди можуть бути точно виміряні або спрогнозовані. Змінність у часі є характерною рисою аграрних даних, оскільки погодні умови, стан ґрунтів та інші параметри постійно змінюються, що вимагає постійного оновлення та адаптації бази знань. Географічна прив'язка є важливою характеристикою аграрних даних, оскільки багато параметрів залежать від місцезнаходження полів та інших об'єктів [10, 18, 19, 35]. Врахування цих особливостей аграрних даних є ключовим для створення ефективних та надійних баз знань для консультативних систем.

Для наповнення баз знань аграрних консультативних систем використовуються різноманітні джерела інформації, які можна класифікувати на три основні групи: експертні знання, структуровані дані та неструктуровані дані. Експертні знання є цінним джерелом інформації, що базується на досвіді та знаннях досвідчених агрономів та фермерів, науковців та дослідників в аграрній сфері, а також консультантів та спеціалістів з аграрного бізнесу [2, 30, 38]. Експертні знання дозволяють враховувати практичні аспекти аграрного виробництва, які не завжди відображені в наукових публікаціях або базах даних.

Структуровані дані представлені у вигляді баз даних, державних реєстрів та кадастрів, а також результатів лабораторних досліджень. Бази даних можуть містити агрометеорологічні дані, інформацію про сорти рослин та породи тварин, дані про внесення добрив та використання засобів захисту рослин, а також економічні показники аграрного виробництва. Державні реєстри та кадастри містять інформацію про земельні ділянки, власників землі та правові обмеження на використання землі. Результати лабораторних досліджень

надають інформацію про хімічний склад ґрунтів, якість води та інші параметри, що впливають на аграрне виробництво [19, 35, 39, 48, 49].

Неструктуровані дані включають наукові статті та публікації, технічну документацію та інструкції, новини та звіти з аграрного ринку, а також форуми та онлайн-спільноти аграріїв. Наукові статті та публікації містять результати новітніх досліджень в аграрній сфері, включаючи нові сорти рослин, методи вирощування та технології захисту рослин. Технічна документація та інструкції надають інформацію про використання сільськогосподарської техніки та обладнання, а також про застосування пестицидів та інших хімічних речовин. Новини та звіти з аграрного ринку містять інформацію про ціни на сільськогосподарську продукцію, попит та пропозицію, експорт та імпорт, а також інші економічні показники. Форуми та онлайн-спільноти аграріїв є цінним джерелом практичного досвіду та знань, якими діляться фермери та інші учасники аграрного ринку [3, 5, 13, 17, 24, 28, 29, 31, 33, 36, 37, 40, 41, 43, 44, 46, 47]. Інтеграція інформації з усіх цих джерел дозволяє створити повну та актуальну базу знань для аграрних консультативних систем.

Представлення знань в аграрних базах знань вимагає врахування специфіки предметної області та забезпечення можливості ефективного використання знань для підтримки прийняття рішень [2, 30, 38]. Існує кілька моделей представлення знань, кожна з яких має свої переваги та недоліки. Модель, заснована на правилах (Rule-based systems), представляє знання у вигляді набору правил "ЯКЩО-ТО", що зручно для формалізації експертних знань та досвіду. Семантичні мережі (Semantic Networks) представляють знання у вигляді графу, де вузли представляють об'єкти та поняття, а ребра – зв'язки між ними, що дозволяє відображати складні семантичні відносини між різними елементами аграрної екосистеми. Онтології (Ontologies) є більш формальним способом представлення знань, що визначає поняття, їх властивості та відносини між ними, забезпечуючи семантичну узгодженість та можливість автоматичного виведення нових знань [19, 35, 39, 48, 49]. Фрейми (Frames) представляють знання у вигляді структурованих об'єктів, що складаються з набору слотів, які

описують властивості та характеристики об'єкта, що дозволяє ефективно представляти інформацію про конкретні аграрні об'єкти, такі як сорти рослин або породи тварин.

Видобування знань (Knowledge Extraction) є важливим етапом створення аграрних баз знань, який передбачає отримання знань з різних джерел. Ручне видобування знань (Knowledge Engineering) передбачає безпосередню взаємодію з експертами для отримання та формалізації їх знань, що є трудомістким, але дозволяє отримати високоякісні та надійні знання. Автоматичне видобування знань з текстів (Text Mining) використовує методи обробки природної мови для виявлення та вилучення знань з неструктурованих текстових даних, таких як наукові статті та звіти [3, 20, 25, 27, 29, 34]. Видобування знань з баз даних (Data Mining) використовує методи машинного навчання для виявлення прихованих закономірностей та залежностей у структурованих даних. Комбіновані підходи поєднують ручне та автоматичне видобування знань для отримання більш повної та точної інформації.

Представлення невизначеності та нечіткості є важливим аспектом створення аграрних баз знань, оскільки аграрні процеси часто залежать від багатьох факторів, які не можуть бути точно виміряні або спрогнозовані. Використання нечіткої логіки (Fuzzy Logic) дозволяє представляти та обробляти знання, які містять нечіткі поняття та оцінки [45]. Застосування ймовірнісних моделей (Probabilistic Models) дозволяє враховувати ймовірність виникнення певних подій або станів, що корисно для прогнозування врожайності або поширення хвороб рослин.

Обробка часових даних є важливим аспектом аграрних баз знань, оскільки багато параметрів, таких як погодні умови та стан ґрунтів, змінюються з часом. Використання тимчасових логік (Temporal Logics) дозволяє представляти та міркувати про часові відносини між різними подіями та станами. Застосування методів часових рядів (Time Series Analysis) дозволяє аналізувати часові дані та прогнозувати їх майбутні значення, що корисно для прогнозування погодних умов або врожайності.

Представлення геопросторових даних є важливим для аграрних баз знань, оскільки багато параметрів залежать від місцезнаходження полів та інших об'єктів. Використання геоінформаційних систем (GIS) дозволяє зберігати та аналізувати геопросторові дані, такі як карти полів, дані про стан ґрунтів та поширення шкідників [18, 19, 35]. Застосування просторової логіки (Spatial Logic) дозволяє формалізувати просторові відносини між різними об'єктами та використовувати їх для прийняття рішень.

Для створення ефективних аграрних баз знань використовуються різноманітні інструменти та технології, які дозволяють зберігати, обробляти та аналізувати великі обсяги даних [6, 10, 15, 18, 19, 22, 23, 31, 35, 39, 48, 49]. Системи управління базами даних (СУБД) є необхідним компонентом будь-якої бази знань, оскільки вони забезпечують зберігання, управління та доступ до даних. Реляційні СУБД, такі як MySQL та PostgreSQL, є добре зарекомендованими рішеннями для зберігання структурованих даних, що відповідають реляційній моделі. Вони забезпечують ACID-властивості (Atomicity, Consistency, Isolation, Durability) та підтримують мову SQL для запитів та управління даними. NoSQL СУБД, такі як MongoDB та Cassandra, є більш гнучкими рішеннями, які дозволяють зберігати дані в різних форматах, таких як JSON або ключ-значення, що робить їх придатними для зберігання неструктурованих або напівструктурованих даних, а також для обробки великих обсягів даних з високою швидкістю [19, 35].

Інструменти для побудови онтологій, такі як Protégé, Web Ontology Language (OWL) та Resource Description Framework (RDF), дозволяють створювати формальні моделі знань, які визначають поняття, їх властивості та відносини між ними. Protégé є популярним інструментом для редагування та візуалізації онтологій, що підтримує різні формати представлення знань, включаючи OWL та RDF. OWL є мовою для опису онтологій, яка базується на Web-технологіях та дозволяє представляти знання у машинозчитуваному форматі. RDF є стандартом для представлення знань у вигляді графу, де вузли представляють об'єкти та поняття, а ребра – зв'язки між ними.

Для видобування знань з текстів використовуються різноманітні інструменти обробки природної мови (NLP), такі як GATE (General Architecture for Text Engineering), Apache OpenNLP та NLTK (Natural Language Toolkit) [3, 20, 27, 29, 34]. GATE є платформою для розробки та розгортання систем обробки тексту, яка надає широкий спектр інструментів для токенізації, розмітки частин мови, розпізнавання іменованих сутностей та інших задач NLP. Apache OpenNLP є бібліотекою з відкритим кодом, яка надає інструменти для виконання різних задач NLP, включаючи токенізацію, виявлення частин мови, вилучення іменованих сутностей, синтаксичний аналіз та інші. NLTK є бібліотекою Python, яка надає інструменти для обробки природної мови, включаючи токенізацію, стеммінг, лематизацію, розмітку частин мови та інші.

Для аналізу даних використовуються такі інструменти, як R та Python (з бібліотеками scikit-learn та pandas) [6, 11, 15, 21, 22, 23, 26, 31, 33, 39, 48, 49]. R є мовою програмування та середовищем для статистичних обчислень та візуалізації даних, яка надає широкий спектр інструментів для аналізу даних, статистичного моделювання та машинного навчання. Python є універсальною мовою програмування, яка широко використовується для аналізу даних та машинного навчання завдяки наявності таких бібліотек, як scikit-learn та pandas. Scikit-learn надає інструменти для класифікації, регресії, кластеризації та інших задач машинного навчання. Pandas надає інструменти для маніпулювання та аналізу структурованих даних. Використання цих інструментів та технологій дозволяє ефективно створювати та використовувати аграрні бази знань для підтримки прийняття рішень в аграрному секторі.

Висновки до розділу 1

У зв'язку зі зростанням потреби в ефективних інструментах підтримки прийняття рішень, консультативні системи на основі ШІ набувають все більшого значення. Розробка таких систем вимагає застосування різних підходів: від традиційних систем на правилах і на знаннях до сучасних, заснованих на машинному навчанні та гібридних архітектурах. Ефективність кожної з цих

систем варіюється залежно від контексту застосування, тому компроміс між простотою, точністю, адаптивністю та пояснюваністю є ключовим. Сучасні тенденції розвитку консультативних систем включають використання глибокого навчання, обробки природної мови, персоналізації та поясненого ШІ.

Специфіка аграрної термінології вимагає спеціалізованих підходів до NLP та ML, враховуючи терміни, абревіатури та полісемію. Для цього використовуються етапи токенізації, видалення стоп-слів, лематизації, а також різні методи представлення тексту (TF-IDF, Word Embeddings) та алгоритми машинного навчання (лінійні моделі, дерева рішень, нейронні мережі). Успішна розробка системи передбачає не лише вибір оптимальних алгоритмів, а й створення якісної бази знань, що інтегрує експертні знання, структуровані дані та неструктуровані текстові джерела, з використанням відповідних СУБД та інструментів для побудови онтологій та аналізу даних.

На основі проведеного теоретичного аналізу, для розробки автоматизованої консультативної системи для аграрного сектору в рамках даної дипломної роботи було обрано наступні ключові підходи:

Використання методів машинного навчання та глибокого навчання (зокрема, трансформерних архітектур типу BERT) для обробки природної української мови, що дозволить ефективно розпізнавати інтенти користувачів та вилучати аграрні сутності.

Створення інтегрованої бази знань на основі СУБД MySQL, що наповнюватиметься даними з інформаційно-довідкового порталу AgroUA.net шляхом автоматизованого вилучення інформації, а також даних, наданих користувачем.

Застосування комбінованого підходу до пошуку інформації, що поєднуватиме лексичний пошук та елементи семантичного пошуку на основі векторних представлень.

Ці рішення спрямовані на створення системи, здатної надавати релевантні та обґрунтовані консультації, враховуючи специфіку аграрної галузі та

української мови. Подальші розділи детально описують проектування, реалізацію та тестування такої системи.

2 РОЗРОБКА МОДЕЛІ ОБРОБКИ ПРИРОДНОЇ МОВИ ТА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ АГРАРНОЇ КОНСУЛЬТАТИВНОЇ СИСТЕМИ НА ОСНОВІ AGROUA.NET

2.1 Розробка архітектури модуля обробки природної мови (NLP) для аналізу запитів користувачів.

Функціонування ефективної автоматизованої консультативної системи, особливо в такій спеціалізованій галузі, як аграрний сектор, неможливе без інтелектуального модуля, здатного адекватно інтерпретувати запити користувачів, сформульовані природною мовою. Даний підрозділ присвячено розробці архітектури такого модуля обробки природної мови (Natural Language Processing, NLP), який слугуватиме фундаментом для розуміння потреб користувачів інформаційно-довідкового порталу AgroUA.net та надання релевантних консультацій.

Першим кроком у розробці є огляд існуючих підходів до побудови NLP-модулів для консультативних систем. Історично, перші спроби створення подібних систем спиралися на підходи, засновані на правилах (rule-based approaches). Ці системи використовували великі набори вручну створених граматичних правил та шаблонів для аналізу вхідних речень. Їх перевагою була висока точність у вузькоспеціалізованих доменах, для яких правила були ретельно розроблені, проте вони демонстрували низьку гнучкість, складність масштабування та погану адаптивність до варіативності природної мови. З розвитком обчислювальних потужностей та доступністю великих текстових корпусів, домінуючими стали статистичні методи та підходи на основі машинного навчання. Класичні алгоритми машинного навчання, такі як Naïve Bayes, Support Vector Machines (SVM) та логістична регресія, успішно застосовувалися для завдань класифікації тексту, зокрема для розпізнавання намірів користувача. На сучасному етапі розвитку NLP провідну роль відіграють

моделі глибокого навчання (Deep Learning), зокрема рекурентні нейронні мережі (RNN), довгої короткострокової пам'яті (LSTM), керовані рекурентні блоки (GRU) та, особливо, архітектури на основі механізму уваги, такі як Трансформери (Transformers). Ці моделі демонструють значно вищу ефективність у розумінні контексту, семантичних зв'язків та нюансів природної мови, що є критично важливим для консультативних систем, які прагнуть до максимально природної взаємодії. Гібридні підходи, що поєднують переваги моделей на основі правил та машинного навчання, також знаходять своє застосування, дозволяючи підвищити точність та контрольованість системи.

Наступним етапом є визначення ключових компонентів NLP-модуля, які забезпечать комплексний аналіз запитів. Фундаментальним завданням виступає розпізнавання інтенту (intent recognition) користувача. Цей компонент відповідає за ідентифікацію основної мети або наміру, що стоїть за запитом. Наприклад, у запиті "Які сорти озимої пшениці стійкі до посухи в Степовій зоні?" інтентом може бути "запит_рекомендації_сортів". Коректне розпізнавання інтенту дозволяє системі спрямувати запит до відповідного модуля обробки або алгоритму пошуку інформації. Не менш важливим компонентом є виокремлення іменованих сутностей (Named Entity Recognition, NER). Для аграрної консультативної системи це означає ідентифікацію та класифікацію ключових об'єктів у тексті запиту, таких як назви сільськогосподарських культур (наприклад, "соняшник", "кукурудза"), шкідників ("колорадський жук"), хвороб рослин ("фітофтороз"), добрив ("аміачна селітра"), регіонів ("Черкаська область"), агротехнічних операцій ("сівба", "обприскування") та інших специфічних термінів. Виокремлені сутності слугують параметрами для уточнення інтенту та формування точного запиту до бази знань. Третім потенційно корисним компонентом може бути аналіз настрою (sentiment analysis). Хоча він не завжди є обов'язковим для суто інформаційних систем, його включення може бути виправданим для оцінки задоволеності користувача, виявлення проблемних запитів, що потребують особливої уваги, або для адаптації тональності відповіді системи. Для даної розробки акцент буде

зроблено на розпізнаванні інтенту та виокремленні сутностей як основних компонентів, з можливістю інтеграції аналізу настрою на подальших етапах.

Після визначення компонентів здійснюється проектування загальної схеми взаємодії компонентів NLP-модуля та потоків даних. Вхідний текстовий запит користувача спочатку проходить етап попередньої обробки, що включає токенізацію (розбиття тексту на окремі слова або субслова), лематизацію або стемінг (приведення слів до їхньої базової форми), видалення стоп-слів та нормалізацію тексту. Очищений та нормалізований текст послідовно або паралельно подається на вхід модуля розпізнавання інтенту та модуля виокремлення іменованих сутностей. Модуль розпізнавання інтенту класифікує запит, визначаючи його основну мету. Одночасно або на основі попередньо визначеного інтенту, модуль NER ідентифікує та маркує ключові сутності в тексті. Результати роботи цих двох модулів – визначений інтент та список виокремлених сутностей з їх типами – об'єднуються у структуроване представлення запиту. Це представлення може мати формат, наприклад, JSON-об'єкта, що містить поля "intent", "entities" (зі списком сутностей та їх значеннями), а також вихідний текст запиту. Якщо використовується аналіз настрою, його результат (наприклад, позитивний, негативний, нейтральний) також додається до цього структурованого представлення. Сформований таким чином структурований запит далі передається на наступні рівні системи – модулю взаємодії з базою знань та модулю генерації відповідей. Ця послідовна обробка забезпечує перетворення неструктурованого мовного вводу в формалізовані дані, придатні для подальшої машинної обробки.

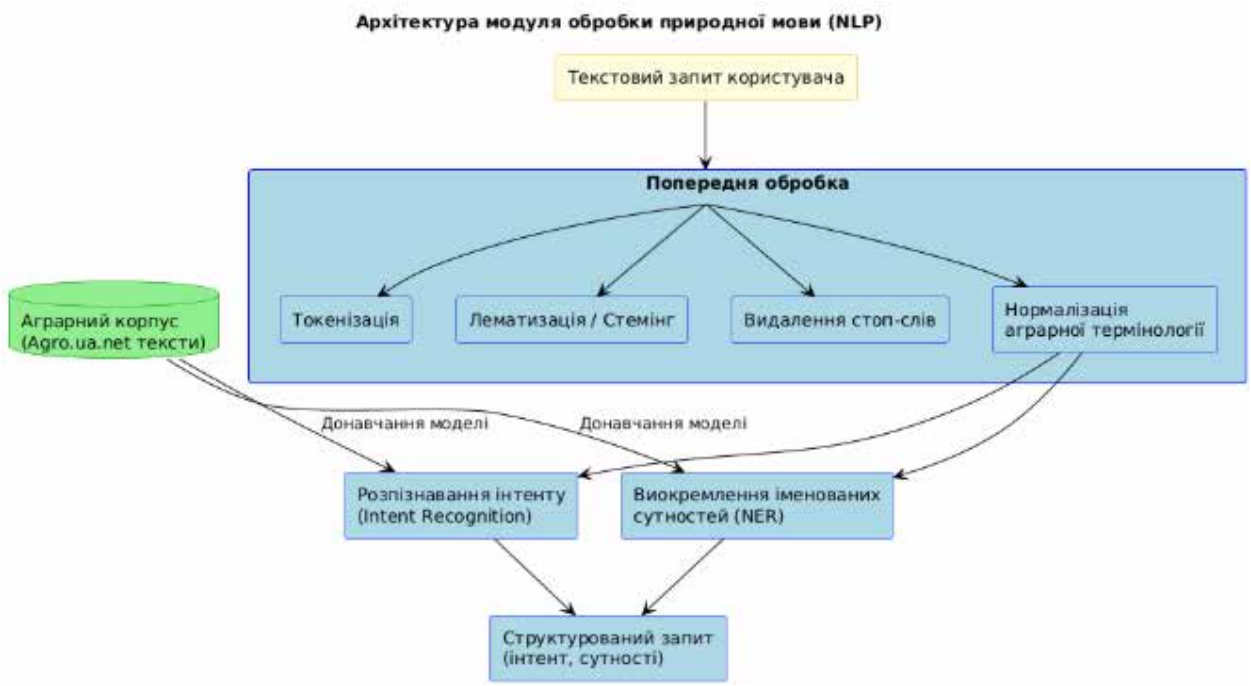

Рис. 2.1 Архітектура модуля обробки природної мови (NLP)

Завершальним аспектом розробки архітектури є обґрунтування вибору технологічного стеку для реалізації NLP-модуля, включаючи бібліотеки та фреймворки. Враховуючи специфіку української мови та аграрної тематики, вибір інструментів має базуватися на кількох ключових критеріях: наявність якісних попередньо навчених моделей для української мови, активна підтримка та розвиток спільнотою, гнучкість для донавчання та адаптації моделей під специфіку домену, а також продуктивність та масштабованість. Для мови програмування Python, яка є де-факто стандартом в галузі машинного навчання та NLP, існує низка потужних бібліотек. Бібліотека spaCy вирізняється високою продуктивністю, наявністю якісних моделей для української мови (включаючи токенізацію, лематизацію, POS-тегування та NER), а також зручним API для створення власних компонентів NLP-пайплайну. Stanza (від Stanford NLP Group) також надає високоякісні інструменти для обробки української мови, часто демонструючи передові результати на стандартних задачах. Для реалізації моделей на основі трансформерів, таких як BERT чи RoBERTa, незамінною є бібліотека Transformers від Hugging Face, яка пропонує доступ до великої кількості попередньо навчених моделей, включаючи ті, що підтримують українську мову, та інструменти для їх тонкого налаштування (fine-tuning) під

конкретні завдання, такі як класифікація тексту (для інтентів) чи розпізнавання іменованих сутностей. Вибір конкретної бібліотеки або їх комбінації буде залежати від результатів експериментальних досліджень на етапі розробки моделей, проте spaCy та Transformers розглядаються як основні кандидати завдяки їхній функціональності, підтримці української мови та можливостям адаптації. Для управління експериментами та версіонування моделей можуть бути використані інструменти на кшталт MLflow.

Якість та ефективність роботи будь-якої системи, що базується на обробці природної мови, значною мірою залежать від ретельності та адекватності етапу попередньої обробки вхідних текстових даних. Цей етап є критично важливим як для формування репрезентативного корпусу знань з порталу AgroUA.net, так і для коректної інтерпретації запитів користувачів. Необроблені текстові дані містять значну кількість шуму, надлишкової інформації та варіативності, що може суттєво ускладнити або навіть унеможливити ефективну роботу моделей машинного навчання.

Першочерговим завданням є збір та підготовка корпусу текстів з AgroUA.net для навчання та формування бази знань. Цей процес передбачає систематичне вилучення текстового контенту з різноманітних розділів порталу, таких як статті, новини, довідкові матеріали, обговорення на форумах (за умови дотримання етичних норм та правил порталу). Для автоматизації цього процесу можуть бути використані методи веб-скрейпінгу, що дозволяють програмно отримувати HTML-сторінки та вилучати з них релевантний текстовий вміст, очищуючи його від HTML-тегів, скриптів, навігаційних елементів та іншої нетекстової інформації. Важливо забезпечити репрезентативність зібраного корпусу, щоб він максимально повно відображав тематику аграрного сектору, представлену на порталі. Зібрані дані структурують, наприклад, за типом контенту або тематичними категоріями, що полегшить подальшу обробку та використання. Також на цьому етапі проводиться первинне очищення даних, наприклад, видалення дублікатів статей або нерелевантних фрагментів.

Наступним кроком є токенізація та сегментація тексту. Токенізація полягає у розбитті суцільного потоку тексту на окремі лексичні одиниці – токени, якими зазвичай є слова, числа або знаки пунктуації. Сегментація тексту, зокрема сегментація на речення, є важливою для подальшого аналізу, оскільки багато NLP-задач оперують саме на рівні речень. Для української мови токенізація може мати певні особливості, пов'язані з використанням дефісів, апострофів та складних слів. Вибір правильного токенізатора, що враховує морфологічні та синтаксичні особливості української мови, є критичним для збереження семантичної цілісності тексту.

Після токенізації здійснюється лематизація або стемінг слів української мови з урахуванням аграрної специфіки. Метою цих процесів є приведення різних граматичних форм слова до його канонічної або базової форми. Семінг є більш грубим методом, який полягає у відсіканні закінчень та суфіксів для отримання основи слова (стему), що не завжди є реально існуючим словом. Лематизація ж передбачає приведення слова до його словникової форми (леми) з використанням морфологічного аналізатора та словників. Для української мови, яка характеризується багатою флективною морфологією, лематизація є більш пріоритетним підходом, оскільки дозволяє точніше зберегти семантичне значення слова. Особливу увагу слід приділити аграрній специфіці, оскільки стандартні лематизатори можуть некоректно обробляти вузькоспеціалізовані терміни (наприклад, назви сортів, препаратів, техніки). Можливо, знадобиться створення або доповнення спеціалізованих словників для коректної лематизації аграрної лексики.

Рис. 2.2 Концептуальна схема бази знань системи

Далі проводиться видалення стоп-слів та пунктуації, а також нормалізація тексту. Стоп-слова – це часто вживані слова, які зазвичай не несуть значного семантичного навантаження (наприклад, прийменники, сполучники, частки). Їх видалення дозволяє зменшити розмірність простору ознак та сконцентруватися на більш інформативних словах. Список стоп-слів для української мови має бути ретельно підібраний, можливо, з додаванням специфічних для аграрної тематики слів, які не є інформативними в даному контексті. Видалення знаків пунктуації також є стандартною процедурою, хоча в деяких випадках пунктуація може нести певну семантичну інформацію (наприклад, знак питання). Нормалізація тексту включає приведення всіх слів до єдиного регістру (зазвичай нижнього) та обробку спеціальних символів, чисел, дат, які можуть бути представлені в різних форматах.

Завершальним етапом попередньої обробки є обробка специфічної аграрної термінології та абревіатур. Аграрний сектор насичений вузькоспеціалізованими термінами, синонімами, омонімами та великою кількістю абревіатур (наприклад, ЗЗР – засоби захисту рослин, КАС – карбамідно-аміачна суміш). Для коректної інтерпретації таких елементів необхідно розробити механізми їх розпізнавання та нормалізації. Це може включати створення спеціалізованих тезаурусів, словників синонімів та розшифрувань абревіатур для аграрної галузі. Наприклад, різні написання назв сортів рослин або діючих речовин препаратів мають бути зведені до єдиного

канонічного вигляду. Така обробка суттєво підвищить якість векторних представлень тексту та ефективність подальшого пошуку й аналізу інформації.

Після попередньої обробки текстові дані мають бути перетворені у числовий формат, придатний для використання моделями машинного навчання. Цей процес, відомий як векторизація або інженерія ознак (feature engineering) для текстових даних, полягає у представленні слів, речень або цілих документів у вигляді векторів чисел. Вибір методу векторизації суттєво впливає на продуктивність та якість роботи всієї консультативної системи.

Почнемо з аналізу класичних методів, таких як Bag-of-Words (BoW), TF-IDF та N-грами, а також їхніх переваг та недоліків для аграрної тематики. Модель "мішка слів" (Bag-of-Words) є одним з найпростіших підходів, де текст представляється як невпорядкований набір слів, а вектор ознак містить частоти входження кожного слова зі словника у документ. Перевагою BoW є простота реалізації та інтерпретації. Однак, цей метод ігнорує порядок слів та семантичні зв'язки між ними, що є суттєвим недоліком для розуміння змісту. Метод TF-IDF (Term Frequency-Inverse Document Frequency) є вдосконаленням BoW, де вага кожного слова визначається не лише його частотою в документі, але й його рідкістю у всьому корпусі текстів. Це дозволяє надавати більшу вагу словам, які є специфічними для конкретного документа, а не загальновживаними. N-грами, що є послідовностями з N слів (наприклад, біграми – 2 слова, триграми – 3 слова), дозволяють частково врахувати локальний контекст та порядок слів. Для аграрної тематики, де часто зустрічаються стійкі словосполучення (наприклад, "озима пшениця", "засоби захисту рослин"), N-грами можуть бути корисними. Однак, класичні методи страждають від високої розмірності простору ознак, розрідженості даних та нездатності вловлювати синонімію чи семантичну близькість слів (наприклад, "хвороба" та "захворювання" будуть різними ознаками).

Зважаючи на обмеження класичних підходів, наступним кроком є огляд та обґрунтування використання методів на основі векторних представлень слів (word embeddings), таких як Word2Vec, GloVe, FastText, та обговорення

необхідності адаптації або донавчання моделей для аграрного домену. Векторні представлення слів (word embeddings) є щільними низькорозмірними векторами, які відображають семантичні властивості слів. Ці моделі навчаються на великих текстових корпусах і здатні вловлювати семантичну близькість між словами (наприклад, слова "трактор" і "комбайн" матимуть близькі вектори). Word2Vec, розроблений Google, пропонує два основних алгоритми: CBOW (Continuous Bag-of-Words), що передбачає слово за його контекстом, та Skip-gram, що передбачає контекст за словом. GloVe (Global Vectors for Word Representation) від Стенфордського університету будує векторні представлення на основі глобальної статистики співвживаності слів у корпусі. FastText, розроблений Facebook AI Research, відрізняється тим, що представляє кожне слово як набір N-грам символів, що дозволяє генерувати вектори для слів, відсутніх у словнику (Out-Of-Vocabulary, OOV), та краще враховувати морфологію слів, що особливо актуально для української мови. Для аграрної консультативної системи використання попередньо навчених на великих українськомовних корпусах моделей word embeddings є відправною точкою. Однак, для досягнення максимальної ефективності, необхідна їх адаптація або донавчання (fine-tuning) на специфічному корпусі текстів з порталу AgroUA.net. Це дозволить моделям краще зрозуміти нюанси аграрної термінології та семантичні зв'язки між специфічними для галузі поняттями.

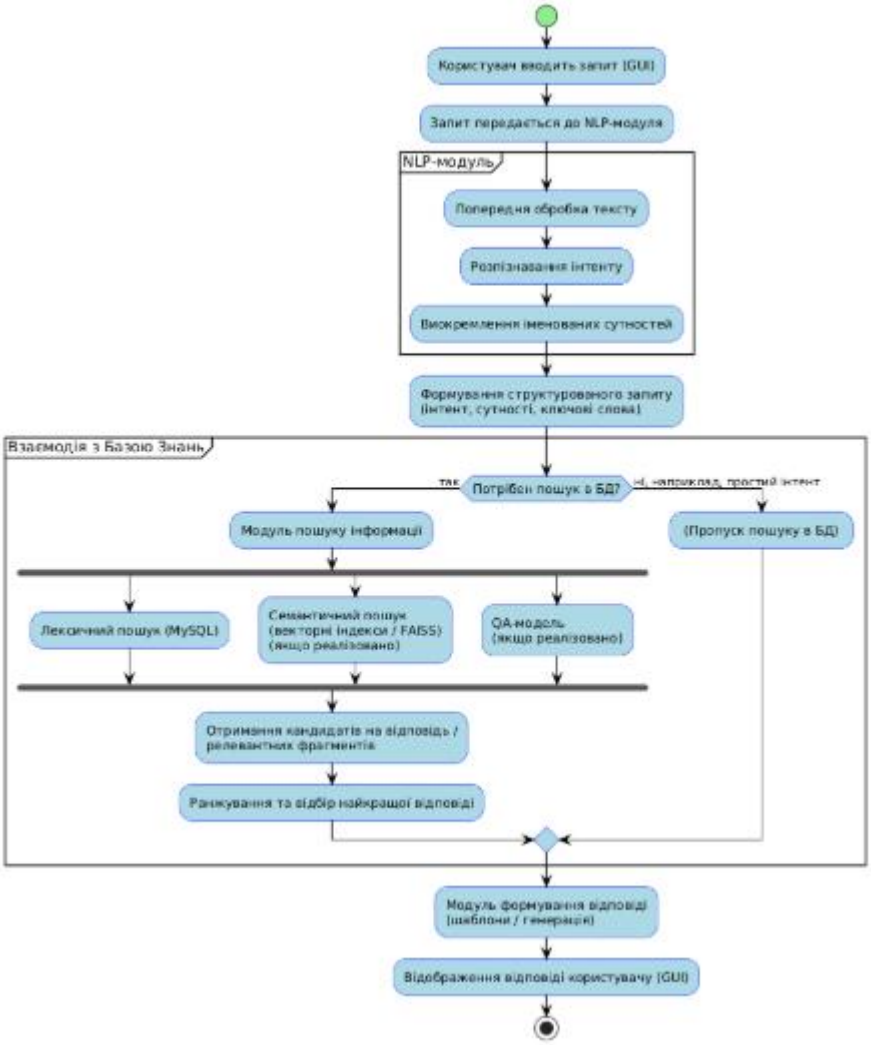

Рис. 2.3 Загальна блок-схема обробки запиту користувача

Окрім векторних представлень окремих слів, важливим є розгляд методів векторного представлення речень або документів (sentence/document embeddings), наприклад, з використанням Sentence-BERT або шляхом агрегації word embeddings. Для завдань, таких як пошук схожих запитів, класифікація інтентів або зіставлення запиту користувача з документами в базі знань, необхідні векторні представлення цілих текстових фрагментів. Простим підходом є агрегація векторів слів, що входять до речення/документа (наприклад, усереднення або сумування, можливо, з TF-IDF вагами). Більш просунуті методи, такі як Doc2Vec (Paragraph Vectors), є розширенням Word2Vec для навчання векторів абзаців або документів. Сучасні трансформерні архітектури, зокрема Sentence-BERT (SBERT), демонструють високу ефективність у генерації семантично насичених векторних представлень речень. SBERT модифікує

попередньо навчені моделі BERT для отримання фіксованих векторів речень, які можна порівнювати за допомогою косинусної подібності, що робить їх надзвичайно корисними для завдань семантичного пошуку та кластеризації. Використання таких моделей для представлення запитів користувачів та документів з бази знань AgroUA.net може значно підвищити точність та релевантність консультативної системи.

Центральним елементом будь-якої консультативної системи є її база знань (Knowledge Base, KB), яка слугує сховищем інформації, необхідної для надання відповідей на запити користувачів. Для розроблюваної аграрної консультативної системи, основним джерелом знань виступає контент інформаційно-довідкового порталу AgroUA.net. Процес формування та структурування цієї бази знань є багатоетапним і вимагає ретельного підходу до вилучення, організації та забезпечення доступу до інформації.

Першим етапом є застосування методів автоматичного вилучення інформації з текстового контенту порталу, такого як статті, новини, довідники та інші релевантні матеріали. Зважаючи на великі обсяги неструктурованих текстових даних, ручне наповнення бази знань є непрактичним. Тому використовуються методики Information Extraction (IE), які дозволяють автоматично ідентифікувати та вилучати значущі факти, сутності та відношення між ними з текстових джерел. Це може включати використання раніше розробленого модуля Named Entity Recognition (NER) для ідентифікації аграрних термінів, а також технік вилучення відношень (Relation Extraction), які допомагають встановлювати зв'язки між сутностями (наприклад, "сорт X стійкий до хвороби Y", "препарат Z використовується для культури W"). Для специфічних, добре структурованих розділів порталу (наприклад, таблиць з характеристиками сортів чи добрив) можуть застосовуватися підходи, засновані на аналізі DOM-структури веб-сторінок або регулярних виразах, адаптованих під конкретний формат представлення даних.

Наступним критично важливим кроком є структурування вилучених знань та вибір підходів до їх організації. Вилучена інформація, навіть якщо вона вже є

частково структурованою (наприклад, у вигляді трійок "суб'єкт-предикат-об'єкт"), потребує подальшої організації для ефективного використання. Один з можливих підходів полягає у формуванні пар "питання-відповідь" (Question-Answer pairs). Це може бути досягнуто шляхом аналізу FAQ-розділів порталу, або шляхом перетворення декларативних речень зі статей на пари запитань та відповідей, наприклад, за допомогою шаблонів або моделей машинного навчання, навчених на відповідних даних. Інший підхід – це організація знань у вигляді тематичних блоків або графів знань, де вузлами є сутності (культури, хвороби, агротехнічні прийоми), а ребрами – відношення між ними. Такий підхід дозволяє моделювати складні залежності та забезпечувати більш гнучкий пошук. Для менш структурованого контенту, наприклад, довгих статей чи новин, ефективним може бути підхід, заснований на індексації цілих документів або значущих фрагментів (абзаців), які потім можуть бути знайдені за допомогою інформаційного пошуку. Ймовірно, оптимальним буде гібридний підхід, що поєднує різні методи структурування для різних типів контенту з AgroUA.net.

Після структурування знань необхідна розробка механізмів індексації та швидкого пошуку в базі знань. Незалежно від обраної структури, база знань повинна забезпечувати швидкий доступ до релевантної інформації у відповідь на запит користувача. Для текстових даних, індексованих як документи або фрагменти, можуть використовуватися повнотекстові пошукові рушії, такі як Elasticsearch або Apache Solr. Ці системи дозволяють створювати інвертовані індекси, підтримують складні запити, ранжування результатів за релевантністю (наприклад, за алгоритмами BM25 або TF-IDF) та фасетний пошук. Для баз знань, що використовують векторні представлення текстів (отримані на етапі 2.3), критично важливим є застосування технологій для ефективного пошуку найближчих сусідів у багатовимірному просторі. Бібліотеки, такі як FAISS (Facebook AI Similarity Search) або Annoy, дозволяють реалізувати швидкий семантичний пошук, знаходячи документи або фрагменти, векторні представлення яких найбільш близькі до вектора запиту користувача. Комбінація повнотекстового та семантичного пошуку може забезпечити найкращі

результати, поєднуючи точність ключових слів з гнучкістю семантичного розуміння.

Завершальним етапом є опис формату зберігання даних в базі знань. Вибір формату залежить від обраних методів структурування та індексації. Якщо база знань складається переважно з пар "питання-відповідь" або структурованих записів про сутності та їх атрибути, то документо-орієнтовані бази даних (наприклад, MongoDB) або реляційні бази даних (наприклад, PostgreSQL) можуть бути доцільними. JSON або XML можуть використовуватися як формати для обміну даними та зберігання окремих записів. У випадку використання графів знань, спеціалізовані графові бази даних (наприклад, Neo4j) пропонують оптимальні засоби для зберігання та запитів до таких структур. Для великих обсягів текстових даних, що індексуються пошуковими рушіями, самі ці рушії (як Elasticsearch) виступають і як система зберігання, і як система індексації. Важливо, щоб обраний формат забезпечував гнучкість, масштабованість та ефективність виконання запитів, необхідних для функціонування консультативної системи.

Для реалізації інтелектуальних функцій аграрної консультативної системи, таких як розуміння намірів користувача, вилучення ключової інформації з запитів та пошук релевантних відповідей, необхідне застосування різноманітних моделей машинного навчання (МН). Вибір конкретних моделей та алгоритмів має ґрунтуватися на специфіці завдань, характеристиках наявних даних та сучасних досягненнях у галузі обробки природної мови.

Ключовим завданням є класифікація інтентів користувача. Для цього необхідний вибір алгоритмів та їх обґрунтування. Класичні алгоритми, такі як Support Vector Machines (SVM) з лінійними або нелінійними ядрами, та логістична регресія, можуть демонструвати хорошу ефективність на текстових даних, особливо після ретельного відбору ознак (наприклад, TF-IDF). Їх перевагами є відносна простота реалізації та інтерпретації. Однак, для складніших випадків, де інтент залежить від тонких семантичних нюансів, більш потужними є нейронні мережі. Згорткові нейронні мережі (CNN) добре

зарекомендували себе у класифікації текстів, здатні виявляти локальні ознаки (N-грами), тоді як рекурентні нейронні мережі (RNN, LSTM, GRU) краще вловлюють послідовні залежності в тексті. Сучасні трансформерні архітектури (наприклад, BERT та його варіанти) демонструють найвищу якість у задачах розуміння мови, включаючи класифікацію інтентів, завдяки механізмам уваги, що дозволяють моделювати довгострокові залежності та контекст. Вибір буде залежати від обсягу та якості навчальних даних, а також від доступних обчислювальних ресурсів. Наступним кроком є підготовка навчальних даних для класифікації інтентів. Це включає збір репрезентативного набору запитів користувачів (реальних або синтезованих) та їх ручну розмітку, тобто присвоєння кожному запиту відповідної мітки інтенту (наприклад, "запит_інформації_про_хворобу", "запит_рекомендації_добрив", "запит_агротехніки_культури"). Якість та кількість розмічених даних є критично важливими для успішного навчання моделі. Нарешті, проводиться навчання та тестування моделі класифікації інтентів, що включає поділ даних на навчальну, валідаційну та тестову вибірки, вибір функції втрат, оптимізатора та метрик оцінки (наприклад, точність, повнота, F1-міра).

Другим важливим завданням є визначення сутностей (Named Entity Recognition - NER) в аграрній сфері. Це передбачає адаптацію існуючих NER-моделей або розробку власної для аграрних термінів, таких як назви культур, сортів, хвороб, шкідників, діючих речовин препаратів, агротехніки, географічних назв тощо. Стандартні NER-моделі, навчені на загальновживаних текстах, можуть не розпізнавати специфічну аграрну лексику. Тому необхідне або донавчання існуючих моделей на аграрному корпусі текстів, або розробка спеціалізованої моделі. Для цього можуть використовуватися як класичні статистичні методи, наприклад, Умовні Випадкові Поля (Conditional Random Fields, CRF), які добре моделюють послідовності міток, так і більш сучасні підходи на основі глибокого навчання. Комбінації BiLSTM-CRF (двонаправлена LSTM з шаром CRF) довгий час були стандартом для NER, ефективно поєднуючи здатність LSTM вивчати контекстні ознаки з можливістю CRF

враховувати залежності між мітками. Трансформерні моделі (BERT, RoBERTa), адаптовані для завдання NER (наприклад, шляхом додавання класифікаційного шару на вихід кожного токена), на сьогодні демонструють передові результати, особливо при наявності достатньої кількості навчальних даних або можливості донавчання на доменних даних.

Третім комплексом завдань є розробка моделей для пошуку релевантної інформації та генерації відповідей (Question Answering/Information Retrieval). Тут ключовим є застосування методів семантичного пошуку на основі векторних представлень для пошуку в базі знань. Запит користувача та документи/фрагменти в базі знань представляються у вигляді векторів (як обговорювалося в п. 2.3), і пошук зводиться до знаходження документів, вектори яких є найбільш близькими до вектора запиту (зазвичай за косинусною подібністю). Це дозволяє знаходити семантично релевантну інформацію, навіть якщо запит не містить точних ключових слів з документа. Далі, важливим є розгляд можливості використання екстрактивних QA моделей (наприклад, на базі BERT) для знаходження точних відповідей у текстах. Такі моделі, навчені на завданнях типу SQuAD (Stanford Question Answering Dataset), здатні виділяти фрагмент тексту (span) з наданого контексту, який є відповіддю на поставлене запитання. Це дозволяє надавати більш конкретні та сфокусовані відповіді, а не просто посилання на релевантний документ. Окремо слід обговорити можливість (та складність) використання генеративних моделей для формування відповідей. Моделі, такі як GPT (Generative Pre-trained Transformer) або BART, здатні генерувати текстові відповіді природною мовою, а не просто вилучати їх. Це може призвести до більш природної та гнучкої взаємодії. Однак, такі моделі є обчислювально складними, вимагають значних обсягів навчальних даних, а також несуть ризик генерації неточної або неправдивої інформації ("галюцинацій"). Тому їх використання в рамках даної дипломної роботи може бути обмеженим або розглядатися як напрямок для подальших досліджень.

Нарешті, для всіх обраних моделей необхідне обґрунтування вибору гіперпараметрів та стратегій їх оптимізації. Гіперпараметри (наприклад,

швидкість навчання, розмір батчу, кількість шарів у нейронній мережі, параметри регуляризації) суттєво впливають на якість моделі. Їх підбір може здійснюватися за допомогою таких методів, як пошук по сітці (grid search), випадковий пошук (random search) або більш просунуті методи байєсівської оптимізації. Валідаційна вибірка використовується для оцінки якості моделі з різними наборами гіперпараметрів та вибору найкращого з них, щоб уникнути перенавчання на тестовій вибірці.

Після того, як запит користувача оброблений NLP-модулем, а відповідні моделі машинного навчання надали свою оцінку та знайшли потенційно релевантну інформацію, наступним кроком є формування зрозумілої, точної та корисної відповіді або рекомендації. Цей процес вимагає розробки чіткого алгоритму, який інтегрує різні компоненти системи.

Першим елементом цього алгоритму є інтеграція результатів роботи NLP-модуля (визначений інтент, виокремлені сутності) та моделей машинного навчання (наприклад, результати пошуку в базі знань або відповідь від QA-моделі). Наприклад, якщо інтент визначено як "запит_інформації_про_хворобу", а сутності включають назву культури та назву хвороби, ці дані використовуються для фільтрації та пріоритезації інформації, отриманої від модуля пошуку. Якщо екстрактивна QA-модель надала конкретний фрагмент тексту як відповідь, цей фрагмент стає основним кандидатом для відповіді користувачеві. Цей етап передбачає створення логіки, яка визначає, як різні частини обробленої інформації взаємодіють та впливають на кінцеву відповідь.

Наступним важливим кроком є механізми ранжування та відбору найбільш релевантних фрагментів інформації з бази знань. Навіть після семантичного пошуку або роботи QA-моделі, система може отримати декілька кандидатів на відповідь. Необхідно впровадити механізми, які оцінюють релевантність кожного кандидата та ранжують їх. Це може базуватися на ступені подібності векторних представлень, оцінці впевненості (confidence score) від QA-моделі, наявності всіх ключових сутностей із запиту у фрагменті відповіді, актуальності інформації (наприклад, дати публікації статті) або на популярності певних

сторінок порталу AgroUA.net. Система повинна вибрати один або декілька найкращих фрагментів для формування остаточної відповіді.

Після відбору інформації відбувається розробка шаблонів або стратегій для генерування зв'язних та зрозумілих відповідей користувачеві. Просто надати витягнутий фрагмент тексту не завжди є оптимальним. Відповідь має бути представлена у зручній для сприйняття формі. Це може бути реалізовано за допомогою системи шаблонів, де в заздалегідь підготовлені фрази підставляється знайдена інформація. Наприклад: "За вашим запитом про [назва_хвороби] на [назва_культури], знайдено наступну інформацію: [фрагмент_відповіді]. Детальніше ви можете ознайомитися за посиланням: [URL_джерела]". Для більш складних сценаріїв можуть розроблятися динамічні стратегії, які адаптують структуру відповіді залежно від типу інтенту, кількості знайденої інформації та її характеру. Наприклад, якщо знайдено декілька релевантних аспектів, відповідь може бути структурована у вигляді списку. Важливою є чіткість, лаконічність та відсутність двозначностей у генерованій відповіді.

Нарешті, для підвищення якості взаємодії, бажаним є врахування контексту діалогу (за потреби та можливості). Якщо користувач ставить уточнюючі запитання, система повинна бути здатною зрозуміти, що нове запитання пов'язане з попереднім. Це вимагає зберігання історії діалогу (принаймні на короткий період) та використання інформації з попередніх реплік для коректної інтерпретації поточного запиту. Наприклад, якщо користувач спочатку запитав "Які симптоми фітофторозу?", а потім "Як його лікувати?", система повинна зрозуміти, що "його" відноситься до фітофторозу. Врахування контексту значно ускладнює систему, але суттєво покращує досвід користувача та природність взаємодії.

Для об'єктивної оцінки ефективності розроблених моделей обробки природної мови та алгоритмів машинного навчання, а також всієї консультативної системи в цілому, необхідно визначити набір адекватних

критеріїв та метрик. Ці метрики дозволять кількісно оцінити якість роботи на різних етапах та порівнювати різні підходи.

Для оцінки компонентів NLP-модуля використовуються специфічні метрики. Так, метрики для оцінки класифікації інтентів та NER включають точність (accuracy), яка показує частку правильно класифікованих прикладів; повноту (recall), що вимірює частку правильно знайдених прикладів певного класу серед усіх реальних прикладів цього класу; та F1-міру (F1-score), яка є гармонійним середнім між точністю та повнотою і дає збалансовану оцінку, особливо важливу при незбалансованих класах. Для NER ці метрики розраховуються на рівні окремих токенів або цілих сутностей.

Для оцінки ефективності пошуку інформації в базі знань застосовуються метрики для оцінки інформаційного пошуку. До них належать Precision@k (точність на перших k результатах), яка показує частку релевантних документів серед k перших виданих системою; Recall@k (повнота на перших k результатах), що вимірює частку знайдених релевантних документів серед k перших відносно загальної кількості релевантних документів у базі; MAP (Mean Average Precision), що усереднює точність для кожного релевантного документа у видачі по всіх запитах; та NDCG (Normalized Discounted Cumulative Gain), яка враховує не тільки наявність релевантного документа, але і його позицію у видачі, а також ступінь релевантності (якщо вона градуйована).

Якщо система використовує моделі для відповіді на питання, то необхідні метрики для оцінки Question Answering систем. Для екстрактивних QA-систем, які виділяють відповідь з тексту, основними метриками є Exact Match (EM), що вимірює відсоток відповідей, які точно збігаються з еталонною відповіддю, та F1-score, яка оцінює перекриття на рівні слів між згенерованою та еталонною відповідями. Для генеративних QA-систем, які формують відповідь самостійно, використовуються метрики, такі як ROUGE (Recall-Oriented Understudy for Gisting Evaluation), що порівнює N-грами згенерованої відповіді з еталонними, та BLEU (Bilingual Evaluation Understudy), яка оцінює подібність

згенерованого тексту до одного або декількох еталонних перекладів (в даному випадку, еталонних відповідей).

2.2 Проектування алгоритмів машинного навчання для класифікації запитів та пошуку релевантної інформації.

Ефективність автоматизованої консультативної системи на основі порталу AgroUA.net безпосередньо залежить від здатності системи точно інтерпретувати запити користувачів та надавати відповідну інформацію з наявної бази знань. Це вимагає розробки та інтеграції складних алгоритмів машинного навчання, які б вирішували два ключові завдання: класифікацію намірів користувача та ефективний пошук релевантних даних.

Першочерговим етапом є аналіз та формалізація задачі класифікації запитів користувачів. Для аграрної консультативної системи, що базується на контенті AgroUA.net, критично важливим є визначення та категоризація типів інтентів (намірів) користувачів, специфічних для даної тематики. На основі аналізу потенційних запитів та змісту порталу, можна виділити такі основні категорії інтентів: запит інформації про конкретний сорт сільськогосподарської культури, запит даних про хвороби рослин або шкідників, пошук інформації про технології вирощування (наприклад, строки сівби, норми внесення добрив, методи захисту), запит контактної інформації організацій чи фахівців, потреба в загальній консультації з невизначеної проблеми, пошук новин чи аналітичних матеріалів тощо. Створення детального та вичерпного переліку можливих інтентів є фундаментальною передумовою для побудови точного класифікатора. Паралельно з цим відбувається формування вимог до точності, повноти та швидкості класифікації інтентів. Для консультативної системи висока точність (precision) є пріоритетною, щоб уникнути надання нерелевантних консультацій через неправильно визначений намір. Повнота (recall) також важлива, щоб система могла коректно обробляти максимально широкий спектр запитів.

Швидкість класифікації має бути достатньою для забезпечення комфортної взаємодії з користувачем в режимі реального часу.

На основі формалізованої задачі здійснюється розробка та обґрунтування вибору моделей машинного навчання для класифікації інтентів. Насамперед проводиться огляд існуючих підходів до класифікації текстів. Традиційні методи, такі як наївний баєсівський класифікатор (Naive Bayes), метод опорних векторів (Support Vector Machines, SVM) та логістична регресія, при використанні з ознаками типу TF-IDF або N-грамами, можуть забезпечити прийнятну якість для відносно простих задач класифікації, особливо за наявності ретельно підготовлених даних. Їх перевагами є менша потреба в обчислювальних ресурсах та простіша інтерпретація. Однак, для врахування складних семантичних залежностей та контексту, що є характерним для природномовних запитів, більш ефективними є нейромережеві архітектури. Згорткові нейронні мережі (Convolutional Neural Networks, CNN) здатні виявляти локальні патерни в тексті, рекурентні нейронні мережі (Recurrent Neural Networks, RNN) та їх модифікації, такі як LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit), ефективно обробляють послідовності та враховують довгострокові залежності. Найсучасніші результати в задачах розуміння природної мови, включаючи класифікацію текстів, демонструють трансформерні архітектури (наприклад, BERT, RoBERTa, XLM-RoBERTa). Їх механізми уваги дозволяють моделювати взаємозв'язки між усіма словами в реченні, забезпечуючи глибоке розуміння контексту. Далі відбувається обґрунтування вибору конкретної моделі (або ансамблю моделей) для класифікації інтентів, враховуючи специфіку української мови, доступність навчальних даних та обчислювальні ресурси. Наприклад, використання попередньо навченої на українськомовних текстах трансформерної моделі з подальшим донавчанням (fine-tuning) на специфічних для AgroUA.net даних може бути оптимальним рішенням, якщо наявні достатні ресурси. В іншому випадку, більш прості моделі, такі як SVM з ретельно підібраними ознаками, можуть бути компромісним варіантом.

Невід'ємною частиною є опис процесу підготовки навчальних даних: методи збору, розмітки та аугментації даних для навчання класифікатора інтентів на основі матеріалів AgroUA.net та симульованих запитів. Це може включати збір реальних запитів (якщо такі доступні), генерацію синтетичних запитів на основі контенту порталу, ручну розмітку даних експертами в аграрній сфері, а також застосування технік аугментації (наприклад, заміна синонімів, зворотний переклад) для збільшення обсягу навчальної вибірки та підвищення робастності моделі.

Паралельно з класифікацією запитів, здійснюється аналіз та формалізація задачі пошуку релевантної інформації в базі знань. На цьому етапі ключовим є визначення критеріїв релевантності інформації для різних типів аграрних запитів. Те, що є релевантним для запиту про технологію вирощування, може відрізнятися від критеріїв для запиту про конкретний сорт чи хворобу. Необхідний також аналіз типів пошукових запитів: від простих ключових слів до складних природномовних питань. Система повинна бути здатною обробляти як короткі запити ("хвороби томатів"), так і більш розгорнуті питання ("Які сорти озимої пшениці найбільш стійкі до вилягання та хвороб в умовах Лісостепу України?"). Важливим аспектом є формулювання вимог до повноти, точності та швидкості пошуку. Система повинна прагнути знаходити всю релевантну інформацію (висока повнота), при цьому мінімізуючи кількість нерелевантних результатів (висока точність), і робити це за прийнятний для користувача час.

На основі цих вимог проводиться розробка та обґрунтування вибору алгоритмів для пошуку та ранжування інформації. Перш за все, здійснюється огляд методів інформаційного пошуку (Information Retrieval, IR). Традиційний лексичний пошук, заснований на алгоритмах, таких як TF-IDF (Term Frequency-Inverse Document Frequency) або BM25 (Okapi BM25), зіставляє ключові слова із запиту з термінами в документах бази знань. Ці методи є швидкими та ефективними для пошуку за точними збігами, але не враховують синонімію та семантичну близькість. Більш просунутим є семантичний пошук на основі векторних представлень (word/sentence embeddings), детальний опис яких буде

надано у наступному підпункті (2.3). Цей підхід дозволяє представляти запити та документи у вигляді векторів у семантичному просторі, де близькість векторів відповідає смисловій схожості, що дозволяє знаходити релевантну інформацію навіть за відсутності точних ключових слів. Часто оптимальними є гібридні підходи, що поєднують переваги лексичного та семантичного пошуку, наприклад, шляхом комбінування їхніх оцінок релевантності. Важливим є також розгляд та обґрунтування використання моделей для відповіді на питання (Question Answering, QA), зокрема екстрактивних QA-моделей (наприклад, на базі BERT, RoBERTa) для знаходження точних відповідей у текстах бази знань. Замість повернення цілого документа, QA-моделі здатні виділяти конкретний фрагмент тексту, що містить відповідь на запит користувача, що значно підвищує зручність та точність консультації. Нарешті, необхідне проектування алгоритмів ранжування результатів пошуку. Навіть якщо знайдено декілька потенційно релевантних документів чи фрагментів, їх потрібно впорядкувати за ступенем відповідності запиту. Ранжування може базуватися на різних факторах: косинусна подібність векторних представлень, оцінка релевантності від лексичного пошуку, наявність ключових сутностей із запиту, популярність чи авторитетність джерела інформації на порталі AgroUA.net, дата публікації (актуальність) та інші.

Завершальним етапом проектування алгоритмів машинного навчання є проектування взаємодії між модулем класифікації запитів, модулем виокремлення сутностей та модулем пошуку інформації. Необхідно розробити чітку схему передачі даних: як результати класифікації інтенту та виокремлені сутності впливають на формування пошукового запиту та вибір стратегії пошуку. Наприклад, якщо інтент визначено як "запит_інформації_про_сорт", а сутність "назва_культури" – "пшениця", а "назва_сорту" – "Богдана", то пошуковий запит буде сформовано з урахуванням цих параметрів, можливо, з пріоритезацією пошуку в розділі порталу, присвяченому сортам пшениці. Також важлива розробка логіки для адаптивного пошуку в залежності від визначеного інтенту. Наприклад, для інтенту "пошук_контактів" система може звертатися до

специфічного розділу бази знань, що містить контактну інформацію, тоді як для інтенту "консультація_з_проблеми_вирощування" буде активовано більш загальний пошук по статтях та довідниках з відповідними фільтрами, сформованими на основі виокремлених сутностей (назва культури, симптом проблеми тощо). Ця інтеграція забезпечить цілісність роботи системи та підвищить релевантність надаваних консультацій.

Ефективність моделей машинного навчання, що застосовуються для класифікації запитів та пошуку інформації в аграрній консультативній системі, нерозривно пов'язана з якістю та адекватністю представлення вхідних текстових даних. Необроблені тексти містять значну кількість "шуму" та варіативності, що ускладнює їх аналіз. Тому етапи попередньої обробки даних та їх подальшого перетворення у числовий формат (векторизації) є критично важливими для успішної реалізації системи.

Першим кроком є застосування методів попередньої обробки текстових даних з порталу AgroUA.net та запитів користувачів. Цей процес включає декілька послідовних операцій. По-перше, токенізація, яка полягає у розбитті суцільного тексту на окремі лексичні одиниці – токени (слова, знаки пунктуації, числа). Для української мови важливо використовувати токенізатори, що враховують її специфіку, наприклад, правильну обробку слів з дефісами та апострофами. Наступним етапом є лематизація або стемінг. Лематизація приводить різні граматичні форми слова до його базової словникової форми (леми), що є більш пріоритетним для української мови з її багатою морфологією, ніж стемінг, який лише відсікає закінчення та суфікси. Якісна лематизація допомагає зменшити розмірність словника та об'єднати семантично однакові слова. Далі виконується видалення стоп-слів – часто вживаних слів, що не несуть значного смислового навантаження (прийменники, сполучники тощо), та пунктуації (хоча в деяких випадках пунктуація може бути інформативною). Також проводиться нормалізація тексту, така як приведення до єдиного регістру (зазвичай нижнього) та обробка спеціальних символів. Особливу увагу слід приділити обробці специфічної аграрної термінології. Це включає розпізнавання

та уніфікацію назв сортів, хвороб, препаратів, техніки, а також розкриття абревіатур, поширених в аграрній галузі. Може знадобитися створення спеціалізованих словників або тезаурусів для коректної обробки таких термінів. Ці кроки спрямовані на очищення даних та приведення їх до стандартизованого вигляду, що полегшує подальшу обробку.

Після попередньої обробки здійснюється вибір та обґрунтування методів векторизації текстових даних, тобто перетворення тексту в числовий формат, придатний для моделей машинного навчання. Одним з класичних підходів є TF-IDF (Term Frequency-Inverse Document Frequency), який представляє документи у вигляді векторів, де кожна компонента відповідає вазі слова, розрахованій на основі його частоти в документі та оберненої частоти в усьому корпусі. Цей метод простий у реалізації, але не враховує семантичної близькості слів. Більш сучасні та ефективні підходи базуються на щільних векторних представленнях слів (word embeddings), таких як Word2Vec, FastText та GloVe. Word2Vec (алгоритми CBOW та Skip-gram) навчається передбачати слово за його контекстом або контекст за словом, формуючи вектори, що відображають семантичні зв'язки. GloVe будує векторні представлення на основі глобальної статистики співвживаності слів. FastText представляє слова як набір N-грам символів, що дозволяє генерувати вектори для слів, відсутніх у словнику, та краще враховувати морфологію. Для представлення цілих речень або документів можуть використовуватися трансформерні ембедінги типу BERT embeddings (отримання векторів для окремих токенів або усереднення для всього речення) або спеціалізовані моделі, як-от Sentence-BERT (SBERT), які генерують семантично насичені вектори для цілих речень, оптимізовані для задач порівняння схожості. Вибір конкретного методу залежить від специфіки завдання (наприклад, для семантичного пошуку SBERT є сильним кандидатом, тоді як для деяких класифікаторів можуть бути достатніми TF-IDF або FastText).

Для досягнення найкращих результатів в аграрній доменній області необхідна адаптація або донавчання (fine-tuning) моделей векторних представлень на корпусі текстів AgroUA.net для аграрної специфіки. Попередньо

навчені на загальновживаних корпусах моделі можуть не повною мірою відображати семантику вузькоспеціалізованих аграрних термінів. Донавчання існуючих моделей (наприклад, Word2Vec, FastText або трансформерних) на текстових даних з порталу AgroUA.net дозволить "налаштувати" векторні представлення так, щоб вони краще вловлювали специфічні для аграрної тематики смислові зв'язки між словами та поняттями. Це може значно покращити якість роботи як класифікатора інтентів, так і системи пошуку інформації.

Завершальним етапом цього підрозділу є експериментальне порівняння обраних методів векторизації. Теоретичні переваги того чи іншого методу мають бути підтверджені практичними результатами на даних, специфічних для проекту. Це порівняння може включати оцінку якості векторних представлень на проміжних задачах, таких як пошук семантично схожих слів або речень, а також оцінку впливу різних методів векторизації на кінцеву продуктивність моделей класифікації інтентів та інформаційного пошуку (використовуючи метрики, описані в п. 2.7). Наприклад, можна порівняти точність класифікатора інтентів, навченого з використанням TF-IDF, FastText та BERT-ембедінгів. Результати експериментів дозволять обґрунтовано вибрати найбільш ефективний метод або комбінацію методів векторизації для подальшої розробки системи.

База знань (БЗ) є ядром консультативної системи, що містить інформацію, необхідну для надання відповідей на запити користувачів. Для системи, що розробляється, основним джерелом наповнення БЗ є інформаційно-довідковий портал AgroUA.net. Процес створення ефективної БЗ охоплює вилучення, структурування, індексацію та визначення формату зберігання даних.

На першому етапі здійснюється застосування методів автоматичного вилучення інформації з текстового контенту порталу. Враховуючи значні обсяги інформації на порталі (статті, новини, довідники, описи продуктів тощо), ручне формування БЗ є нереалістичним. Тому використовуються техніки Information Extraction (IE). Це може включати застосування розробленого раніше модуля Named Entity Recognition (NER) для ідентифікації ключових аграрних сутностей

(назви культур, сортів, хвороб, препаратів, регіонів, агротехнічних прийомів) та Relation Extraction (RE) для виявлення семантичних зв'язків між цими сутностями (наприклад, "сорт X стійкий до хвороби Y", "препарат Z рекомендований для культури W"). Для структурованих даних на порталі, таких як таблиці з характеристиками, можуть застосовуватися методи веб-скрейпінгу, що базуються на аналізі HTML-структури сторінок або використанні регулярних виразів.

Після вилучення інформації критично важливим є її структурування та вибір підходів до організації в базі знань. Існує декілька підходів. Одним із них є створення пар "питання-відповідь" (QA-pairs), які можуть бути отримані шляхом аналізу розділів FAQ на порталі, або шляхом автоматичного чи напівавтоматичного перетворення декларативних речень зі статей на запитання та відповіді. Інший підхід – організація знань у вигляді тематичних блоків або більш формалізованих структур, таких як граф знань (knowledge graph). У графі знань сутності представляються вузлами, а відношення між ними – ребрами, що дозволяє моделювати складні залежності та здійснювати більш глибокий семантичний пошук. Для великих обсягів текстового контенту, таких як довгі статті чи новини, ефективною стратегією є індексація цілих документів або їх значущих фрагментів (наприклад, абзаців), які потім можуть бути знайдені за допомогою повнотекстового або семантичного пошуку. Найімовірніше, для системи на базі AgroUA.net буде застосовано гібридний підхід, що поєднує різні методи структурування для різних типів контенту, наприклад, QA-пари для типових запитань, індексовані документи для статей, та, можливо, елементи графа знань для ключових аграрних понять.

Для забезпечення ефективного доступу до структурованих знань необхідна розробка механізмів індексації, що підтримують алгоритми пошуку, розроблені у підпункті 2.2.4. Для текстових даних, індексованих як документи чи фрагменти, широко використовуються повнотекстові пошукові рушії, такі як Elasticsearch або Apache Solr. Вони створюють інвертовані індекси, що дозволяють швидко знаходити документи за ключовими словами, та

підтримують складні запити й ранжування результатів за релевантністю. У випадку використання семантичного пошуку, заснованого на векторних представленнях текстів, необхідні інструменти для ефективного пошуку найближчих сусідів у багатовимірному просторі. Бібліотеки, такі як FAISS (Facebook AI Similarity Search) або Annoy, надають оптимізовані алгоритми для такого пошуку, дозволяючи швидко знаходити документи або фрагменти, семантично близькі до запиту користувача. Інтеграція цих технологій індексації є ключовою для забезпечення швидкого та точного пошуку в БЗ.

Нарешті, важливим є опис формату зберігання даних в базі знань. Вибір формату залежить від обраних методів структурування та індексації. Для зберігання структурованих записів, таких як QA-пари або атрибути сутностей, можуть використовуватися документо-орієнтовані бази даних (наприклад, MongoDB, що дозволяє зберігати дані у форматі JSON/BSON) або реляційні бази даних (наприклад, PostgreSQL) з відповідною схемою. Якщо реалізується граф знань, то спеціалізовані графові бази даних (наприклад, Neo4j) будуть найбільш підходящим рішенням. Для великих обсягів неструктурованих або напівструктурованих текстових даних, які індексуються пошуковими рушіями, самі ці рушії (наприклад, Elasticsearch) можуть виступати як основне сховище. Обраний формат зберігання має забезпечувати гнучкість для можливих змін у структурі даних, масштабованість для зростання обсягів інформації та ефективність доступу для швидкої роботи консультативної системи.

Після проектування архітектури NLP-модуля, алгоритмів машинного навчання, підготовки даних та формування бази знань, наступним логічним кроком є практична реалізація процесів навчання, тестування та подальшої оптимізації розроблених моделей. Цей етап є вирішальним для досягнення високої продуктивності та надійності аграрної консультативної системи.

Центральним завданням є реалізація процесу навчання та тестування моделі класифікації інтентів. На основі підготовлених та розмічених навчальних даних, обрана модель машинного навчання (наприклад, SVM, логістична регресія, або нейронна мережа на базі трансформера) навчається розпізнавати

закономірності, що пов'язують текстовий зміст запиту користувача з відповідним інтентом. Процес навчання включає подачу на вхід моделі векторних представлень запитів та відповідних їм міток інтентів, оптимізацію параметрів моделі (ваг) за допомогою вибраного алгоритму (наприклад, стохастичного градієнтного спуску) з метою мінімізації функції втрат (наприклад, перехресної ентропії). Важливою складовою є використання валідаційної вибірки для моніторингу процесу навчання, запобігання перенавчанню та вибору оптимального моменту для зупинки навчання. Після завершення навчання проводиться всебічне тестування моделі на відкладеній тестовій вибірці, яка не використовувалася на етапах навчання та валідації. Оцінка якості класифікації здійснюється за допомогою метрик, таких як точність, повнота, F1-міра та аналіз матриці помилок (confusion matrix) для виявлення слабких місць моделі.

Аналогічно, якщо в рамках системи розробляється або адаптується окрема модель для розпізнавання іменованих сутностей, відбувається реалізація процесу навчання/адаптації та тестування моделі NER. Для цього завдання також необхідний розмічений корпус текстів, де аграрні терміни (назви культур, хвороб, препаратів тощо) анотовані відповідними тегами сутностей. Якщо використовується попередньо навчена модель (наприклад, на базі BERT для NER), то здійснюється її донавчання (fine-tuning) на специфічному для AgroUA.net корпусі. Процес навчання та тестування подібний до класифікації інтентів, але оцінка якості проводиться на рівні розпізнавання окремих токенів або цілих сутностей за допомогою метрик, таких як точність, повнота та F1-міра для кожного типу сутностей.

Наступним кроком є налаштування та тестування моделей пошуку та QA. Для моделей інформаційного пошуку, що базуються на лексичних (TF-IDF, BM25) або семантичних (векторні представлення) підходах, налаштування може включати конфігурацію параметрів пошукового рушія (наприклад, Elasticsearch) або вибір порогів для косинусної подібності. Тестування ефективності пошуку проводиться на наборі тестових запитів, для яких відомі релевантні документи або фрагменти з бази знань. Оцінка здійснюється за метриками Precision@k,

Recall@k, MAP, NDCG. Для екстрактивних QA-моделей (наприклад, на базі BERT) процес тестування передбачає подачу на вхід моделі запитання та контекстного документа з бази знань, та порівняння виділеної моделлю відповіді з еталонною відповіддю за метриками Exact Match та F1-score. Важливо забезпечити, щоб моделі пошуку та QA ефективно інтегрувалися з розробленою базою знань.

Критично важливим аспектом є обґрунтування вибору гіперпараметрів для моделей та стратегій їх оптимізації. Ефективність моделей машинного навчання значною мірою залежить від правильно підібраних гіперпараметрів (наприклад, швидкість навчання, кількість епох, розмір шарів у нейронній мережі, параметри регуляризації, кількість сусідів для k-NN у семантичному пошуку тощо). Оскільки ручний підбір є трудомістким та неефективним, застосовуються автоматизовані стратегії оптимізації. До них належать GridSearch (пошук по сітці), який перебирає всі можливі комбінації заданих значень гіперпараметрів; RandomSearch (випадковий пошук), який випадковим чином вибирає комбінації гіперпараметрів із заданих діапазонів, що часто є більш ефективним за GridSearch при великій кількості гіперпараметрів; та більш просунуті методи, такі як Bayesian Optimization (баєсівська оптимізація), яка будує ймовірнісну модель функції залежності якості моделі від гіперпараметрів та використовує її для вибору наступних кандидатів для тестування. Вибір конкретної стратегії залежить від складності моделі, кількості гіперпараметрів та доступних обчислювальних ресурсів. Оптимізація гіперпараметрів проводиться на валідаційній вибірці для уникнення перенавчання та забезпечення узагальнюючої здатності моделей.

Після того, як запит користувача успішно проаналізовано NLP-модулем, а моделі машинного навчання надали результати класифікації, розпізнавання сутностей та пошуку релевантної інформації, необхідно сформувати кінцеву відповідь або рекомендацію для користувача. Цей процес вимагає розробки комплексного алгоритму, що об'єднує різні компоненти системи.

Основою цього алгоритму є інтеграція результатів роботи NLP-модуля та моделей машинного навчання (інтент, сутності, результати пошуку). На цьому етапі відбувається об'єднання інформації, отриманої від різних компонентів. Наприклад, визначений інтент запиту (наприклад, "запит_інформації_про_хворобу") та виокремлені сутності (наприклад, "культура: томат", "хвороба: фітофтороз") використовуються для фільтрації та уточнення результатів, отриманих від модуля інформаційного пошуку або QA-моделі. Якщо QA-модель надала конкретний текстовий фрагмент як відповідь, він стає основним кандидатом. Якщо результати пошуку складаються з декількох релевантних документів, то інтент та сутності допомагають визначити, які аспекти цих документів є найбільш значущими.

Далі включаються механізми відбору та агрегації найбільш релевантної інформації. Навіть після попереднього фільтрування, система може мати декілька кандидатів на відповідь. Необхідно застосувати додаткові критерії для ранжування та вибору найкращого варіанту або комбінації варіантів. Це може включати оцінку впевненості (confidence score) від класифікатора інтентів та QA-моделі, ступінь відповідності знайдених сутностей запиту, актуальність інформації (дата публікації), авторитетність джерела на порталі AgroUA.net, а також, можливо, довжину та зрозумілість потенційної відповіді. В деяких випадках може знадобитися агрегація інформації з декількох джерел для формування більш повної відповіді.

Після відбору релевантної інформації відбувається розробка шаблонів або стратегій для генерування зв'язних та зрозумілих відповідей користувачеві. Просто надати сирий фрагмент тексту або набір фактів не завжди є найкращим рішенням. Відповідь має бути представлена у чіткій, лаконічній та легкозрозумілій формі. Для цього можуть використовуватися попередньо розроблені шаблони відповідей, куди підставляється знайдена інформація. Наприклад: "За Вашим запитом щодо [інтент] стосовно [сутності], на порталі AgroUA.net знайдено наступну інформацію: [фрагмент_відповіді]. Більш детально Ви можете ознайомитися тут: [посилання_на_джерело]". Для більш

складних інтентів або коли необхідно надати комплексну рекомендацію, можуть розроблятися динамічні стратегії генерації відповідей, які адаптують структуру та зміст відповіді залежно від конкретного запиту та знайденої інформації. Можливо, відповідь буде структурована у вигляді списку, таблиці або міститиме декілька абзаців.

Для підвищення якості діалогу та природності взаємодії, бажано передбачити врахування контексту діалогу (за потреби та можливості). Якщо користувач ставить послідовні уточнюючі запитання, система повинна бути здатною пов'язувати їх з попередньою історією взаємодії. Це вимагає зберігання контексту діалогу (наприклад, останніх кількох запитів та відповідей, виокремлених сутностей) та використання цієї інформації для правильної інтерпретації нових запитів (наприклад, для розв'язання анафоричних посилань – "а як щодо цього сорту?"). Врахування контексту значно покращує користувацький досвід, хоча й ускладнює логіку системи.

Для об'єктивної та кількісної оцінки ефективності всіх розроблених компонентів аграрної консультативної системи – від моделей обробки природної мови до алгоритмів формування відповідей – необхідно визначити та застосувати відповідний набір критеріїв та метрик. Це дозволить не тільки оцінити поточну якість системи, але й порівнювати різні підходи та ітерації розробки.

Для оцінки окремих модулів машинного навчання використовуються специфічні метрики. Так, метрики для оцінки класифікації інтентів та NER включають точність (accuracy) – частка правильно класифікованих прикладів; повноту (recall) – частка правильно ідентифікованих прикладів певного класу серед усіх реальних прикладів цього класу; та F1-міру (F1-score) – гармонійне середнє між точністю та повнотою, що є особливо корисним при незбалансованих класах. Для NER ці метрики розраховуються для кожного типу сутності окремо та усереднено.

Для оцінки ефективності пошуку інформації в базі знань застосовуються метрики для оцінки інформаційного пошуку. До них належать

Precision@k (точність на перших k виданих результатах), Recall@k (повнота на перших k виданих результатах), MAP (Mean Average Precision), що усереднює точність по всіх запитах з урахуванням позицій релевантних документів, та NDCG (Normalized Discounted Cumulative Gain), яка враховує не тільки релевантність, але й позицію документа у видачі та ступінь його релевантності (якщо така оцінка наявна).

Якщо система використовує моделі для відповіді на питання (Question Answering), то необхідні метрики для оцінки Question Answering систем. Для екстрактивних QA-систем, що виділяють відповідь з тексту, основними метриками є Exact Match (EM), яка вимірює відсоток відповідей, що точно збігаються з еталонною, та F1-score на рівні слів, що оцінює ступінь перекриття між згенерованою та еталонною відповідями. Для генеративних QA-систем (якщо вони використовуються) застосовуються метрики, такі як ROUGE (Recall-Oriented Understudy for Gisting Evaluation), що порівнює N-грами згенерованої відповіді з еталонними, та BLEU (Bilingual Evaluation Understudy), яка оцінює схожість згенерованого тексту до еталонних відповідей.

На завершення, надзвичайно важливим є розробка протоколу експериментального дослідження та комплексної оцінки ефективності системи в цілому. Це передбачає створення репрезентативних тестових наборів даних, що включають різноманітні запити користувачів з відповідними еталонними відповідями або очікуваними діями системи. Протокол повинен чітко описувати процедуру проведення експериментів, збір даних та методи їх аналізу. Окрім автоматичних метрик, доцільно провести ручну оцінку якості відповідей системи експертами в аграрній галузі та/або потенційними користувачами. Критеріями такої оцінки можуть бути: релевантність відповіді запиту, точність наданої інформації, повнота відповіді, зрозумілість та ясність викладу, а також загальна корисність консультації. Тільки комплексний підхід до оцінки дозволить отримати об'єктивне уявлення про ефективність розробленої автоматизованої консультативної системи.

2.3 Розробка структури бази знань, інтегрованої з даними порталу AgroUA.net.

Фундаментом будь-якої ефективної автоматизованої консультативної системи є добре структурована та релевантна база знань (БЗ). Для системи, що розробляється, основним джерелом інформації слугує інформаційно-довідковий портал AgroUA.net. Цей підрозділ присвячений комплексному процесу розробки структури такої бази знань, починаючи від аналізу джерела даних і закінчуючи вибором технологій для її реалізації, з метою забезпечення ефективної інтеграції та використання аграрної інформації.

Першочерговим етапом є аналіз контенту та структури інформаційного порталу AgroUA.net як основного джерела знань. Це передбачає детальне дослідження порталу для ідентифікації ключових типів інформації, яка може бути використана для наповнення БЗ. До таких типів належать, зокрема, наукові та практичні статті про технології вирощування сільськогосподарських культур, детальні описи сортів та гібридів із зазначенням їх характеристик, інформація про хвороби рослин та шкідників, включаючи симптоми, методи боротьби та рекомендовані препарати, актуальні новини аграрного сектору, різноманітні довідники (наприклад, по добривах, засобах захисту рослин), а також дані про компанії, їхні продукти та послуги. Наступним кроком є аналіз існуючої структури даних на порталі, включаючи наявну категоризацію матеріалів, використання тегів, внутрішні посилання між сторінками та будь-які інші метадані, що можуть допомогти у вилученні та структуруванні інформації. Також важливим є визначення потенційних джерел для автоматичного вилучення як структурованих (наприклад, таблиці з характеристиками, списки), так і неструктурованих (тексти статей, новин) даних, що дозволить максимально повно охопити інформаційний простір порталу.

На основі аналізу джерела формулюється визначення вимог до бази знань для аграрної консультативної системи. По-перше, необхідно встановити необхідний рівень деталізації та гранулярності знань. Чи буде БЗ зберігати

інформацію на рівні окремих фактів, абзаців, цілих статей, чи комбінації цих рівнів. По-друге, БЗ повинна забезпечувати підтримку різних типів запитів, з якими може звертатися користувач: фактологічні (наприклад, "Яка врожайність сорту X?"), процедурні (наприклад, "Як боротися з хворобою Y?"), порівняльні (наприклад, "Який сорт A кращий за сорт W за стійкістю до посухи?") та рекомендаційні (наприклад, "Порадьте добриво для культури A в умовах регіону B"). По-третє, критично важливими є вимоги до оновлюваності та актуальності даних, оскільки інформація в аграрній сфері (нові сорти, препарати, технології, ринкові умови) швидко змінюється. БЗ повинна мати механізми для регулярного оновлення. Нарешті, необхідно врахувати вимоги до швидкості доступу та масштабованості БЗ, щоб забезпечити швидку відповідь системи навіть при зростанні обсягів даних та кількості користувачів.

Далі проводиться огляд та вибір моделей представлення знань, найбільш придатних для аграрного домену. Здійснюється аналіз класичних реляційних моделей, де дані зберігаються у вигляді таблиць зі зв'язками між ними. Цей підхід добре підходить для чітко структурованої інформації, але може бути менш гнучким для представлення складних семантичних відношень у текстах. Розглядаються документо-орієнтовані моделі (наприклад, що використовують JSON), які дозволяють зберігати інформацію у вигляді ієрархічних документів, що є зручним для представлення статей або описів об'єктів з різною кількістю атрибутів. Особливу увагу приділяється оцінці можливостей використання графових моделей знань (knowledge graphs). Графи знань дозволяють представляти інформацію у вигляді сутностей (вузлів) та відношень між ними (ребер), що є надзвичайно потужним інструментом для моделювання складних взаємозв'язків, характерних для аграрної сфери (наприклад, зв'язки між культурами, їх хворобами, рекомендованими препаратами, технологіями вирощування та кліматичними умовами). На основі цього аналізу відбувається обґрунтування вибору оптимальної моделі (або гібридного підходу, що поєднує переваги різних моделей) для структури бази знань на основі контенту AgroUA.net, враховуючи різноманітність та специфіку наявних даних.

Наступним кроком є безпосереднє проектування логічної структури бази знань. Це включає визначення основних сутностей (entities), які будуть представлені в БЗ. Для аграрного домену це такі сутності, як "Культура", "Сорт", "Гібрид", "Хвороба", "Шкідник", "Добриво", "Засіб захисту рослин", "Агротехнічний прийом", "Ґрунт", "Кліматична зона", "Регіон", "Стаття", "Новина", "Компанія" тощо. Для кожної визначеної сутності проводиться визначення її атрибутів (properties), тобто характеристик, що її описують. Наприклад, для сутності "Сорт" атрибутами можуть бути: назва, належність до культури, потенційна врожайність, стійкість до певних хвороб, рекомендовані зони вирощування, терміни дозрівання, посилання на опис на порталі. Важливим етапом є проектування відношень (relationships) між сутностями, що відображають семантичні зв'язки між ними. Наприклад: "Сорт належить до Культури", "Хвороба вражає Культуру", "Добриво застосовується для Культури", "Агротехнічний прийом рекомендований для Сорту в умовах Кліматичної зони". Результатом цього проектування є розробка формальної схеми даних або онтології для аграрної бази знань, яка визначає типи сутностей, їх атрибути та типи відношень між ними, забезпечуючи консистентність та цілісність даних.

Після проектування логічної структури розробляються механізми інтеграції та наповнення бази знань даними з порталу AgroUA.net. Це включає використання методів автоматичного вилучення інформації (Information Extraction) з неструктурованого та напівструктурованого текстового контенту порталу (статті, новини, довідники) для ідентифікації екземплярів визначених сутностей та значень їх атрибутів. Для цього активно використовуються раніше розроблені NLP-компонентів, зокрема модель Named Entity Recognition (NER), яка допомагає розпізнавати аграрні терміни в текстах. Розробляються стратегії парсингу структурованих даних з порталу, таких як інформація, представлена у вигляді таблиць (наприклад, характеристики сортів, норми внесення добрив) або списків. Нарешті, проектуються механізми оновлення та синхронізації бази знань з контентом порталу AgroUA.net, щоб забезпечити актуальність інформації

в БЗ, наприклад, шляхом періодичного повторного сканування порталу або відстеження змін на ключових сторінках.

Основним джерелом знань для розроблюваної системи є приватний інформаційно-довідковий портал - агрегатор сільськогосподарських знань. На момент розробки системи, портал AgroUA.net не надавав публічного API для доступу до своїх даних. У зв'язку з цим, для початкового наповнення бази знань та формування навчальних корпусів було застосовано метод веб-скрейпінгу.

Вилучався переважно текстовий контент, назви, а також, де можливо, метадані (дати публікації, категорії). Отримана інформація проходила попередню очистку від HTML-тегів, скриптів та навігаційних елементів. Далі дані зберігалися у проміжному форматі (наприклад, JSON файли або CSV) для подальшого структурування та завантаження в основну базу знань системи (MySQL).

Вилучені дані структурувалися відповідно до розробленої моделі бази знань. Наприклад, статті зберігалися з полями: URL, назва, текст, дата публікації, категорія.

Для підтримки актуальності бази знань передбачався періодичний запуск скрейпінгових скриптів (наприклад, раз на тиждень) для виявлення нових публікацій та оновлень. Однак, такий підхід має обмеження, пов'язані з можливими змінами у структурі HTML-коду порталу AgroUA.net, що може вимагати адаптації скрейперів. В ідеальному випадку, наявність API значно спростила б та підвищила надійність процесу інтеграції даних.

Завершальним етапом розробки структури БЗ є вибір технологій та інструментів для її реалізації та зберігання. Проводиться огляд та обґрунтування вибору відповідної системи управління базами даних (СУБД). Залежно від обраної моделі представлення знань, це можуть бути реляційні СУБД (наприклад, PostgreSQL, MySQL), які добре підходять для структурованих даних з чіткими зв'язками; документо-орієнтовані СУБД (наприклад, MongoDB, Elasticsearch), що забезпечують гнучкість у зберіганні напівструктурованих даних; або графові СУБД (наприклад, Neo4j, Amazon Neptune), які оптимізовані

для зберігання та обробки даних, представлених у вигляді графів. Окремо розглядаються інструменти для індексації та ефективного пошуку в базі знань, такі як повнотекстові пошукові рушії Elasticsearch або Apache Solr, а також бібліотеки для пошуку за семантичною близькістю, наприклад, FAISS, що є важливим для підтримки алгоритмів пошуку. Нарешті, здійснюється опис конкретного формату зберігання даних в обраній СУБД (наприклад, JSON для документо-орієнтованих баз, специфічні формати таблиць для реляційних баз, або формати вузлів та ребер для графових баз), що забезпечить уніфікований підхід до роботи з даними в системі.

Висновки до розділу 2

Розробка ефективної консультативної системи для аграрного сектору вимагає інтелектуального модуля NLP, який адекватно інтерпретує запити користувачів. Історично системи використовували підходи на основі правил, які були точними у вузьких доменах, але не гнучкими. Сучасні NLP-модулі використовують машинне навчання, зокрема глибоке навчання з архітектурами на основі Transformers, для кращого розуміння контексту та семантики. NLP-модуль повинен включати розпізнавання інтенту користувача та виокремлення іменованих сутностей (наприклад, назв культур, хвороб), які об'єднуються у структуроване представлення запиту. Для української мови ефективними є бібліотеки spaCy та Transformers, що надають попередньо навчені моделі, які можна адаптувати для аграрної термінології. Система потребує бази знань, сформованої з контенту AgroUA.net, для чого застосовуються автоматичне вилучення інформації та структурування, зокрема перетворення на пари "питання-відповідь" або організація у вигляді графів знань. Для швидкого пошуку використовують повнотекстові пошукові рушії (Elasticsearch) та алгоритми семантичного пошуку (FAISS). Важливим є вибір формату зберігання даних (наприклад, JSON, реляційна таблиця, графова база) в СУБД, що відповідає структурі бази знань. Всі моделі потребують ретельного навчання, тестування та оптимізації, що передбачає ретельну підготовку даних та використання метрик для оцінки точності, повноти та релевантності результатів.

Для забезпечення актуальності інформації в базі знань, отриманої з профільних порталів, передбачено механізм періодичного оновлення. Було реалізовано стратегію періодичного веб-скрейпінгу ключових розділів порталу. Скрейпінговий модуль запускається за розкладом раз на тиждень і перевіряє наявність нового контенту або оновлень існуючих сторінок за хешем вмісту. Нові дані вилучаються, проходять етапи попередньої обробки та інтегруються до основної бази знань системи (MySQL). Цей процес може бути автоматизований за допомогою планувальника завдань операційної системи або спеціалізованих інструментів типу Hangfire.

У випадку наявності API від AgroUA.net з підтримкою запитів на отримання оновлених даних (наприклад, за часовою міткою), механізм оновлення був би реалізований через періодичні запити до відповідних ендпоінтів API, що є більш надійним та ефективним підходом.

3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ КОНСУЛЬТАТИВНОЇ СИСТЕМИ

3.1 Реалізація програмного забезпечення для обробки природної мови та машинного навчання.

Ефективна реалізація автоматизованої консультативної системи з використанням штучного інтелекту на основі бази даних інформаційно-довідкового порталу AgroUA.net вимагає ретельного підходу до вибору інструментальних засобів та технологій розробки. Цей вибір має ґрунтуватися на аналізі специфічних вимог до системи, а також на доступності, надійності та відповідності обраних технологій поставленим завданням.

Першим кроком у цьому процесі є аналіз вимог до програмної реалізації системи. Розроблювана система повинна забезпечувати зручний та інтуїтивно зрозумілий інтерфейс користувача для введення запитів природною мовою та отримання консультативних відповідей. Важливою є ефективна взаємодія з базою даних, яка слугуватиме сховищем аграрних знань, вилучених з порталу AgroUA.net, та інших системних даних. Ключовою вимогою є можливість інтеграції моделей обробки природної мови (NLP) та машинного навчання (ML), розроблених на попередніх етапах, для аналізу запитів, класифікації інтентів, виокремлення сутностей та пошуку релевантної інформації. Система також повинна демонструвати належну швидкодію та стабільність роботи.

На основі цих вимог здійснюється обґрунтування вибору мови програмування C# та платформи .NET для розробки основної логіки системи та інтерфейсу користувача. Мова програмування C# є сучасною, об'єктно-орієнтованою мовою, що характеризується високою продуктивністю розробки, строгим контролем типів, що сприяє зменшенню кількості помилок, та потужною стандартною бібліотекою класів. Платформа .NET надає широкі

можливості для створення різноманітних додатків, включаючи настільні системи, та забезпечує ефективне управління пам'яттю, безпеку та інтероперабельність. Багата екосистема .NET та велика кількість доступних бібліотек значно спрощують розробку складних програмних рішень.

Для реалізації графічного інтерфейсу користувача (GUI) консультативної системи здійснено обґрунтування вибору Windows Forms (WinForms). WinForms є зрілою та добре документованою технологією в рамках платформи .NET, призначеною для створення настільних додатків для операційної системи Windows. Вона надає широкий набір стандартних елементів управління, що дозволяє швидко проектувати та реалізовувати функціональні інтерфейси. Для цілей дипломного проекту, де акцент робиться на інтелектуальних функціях системи, а не на складності візуального дизайну, WinForms пропонує достатній інструментарій для створення функціонального та зрозумілого для користувача інтерфейсу без надмірних витрат часу на розробку.

Щодо зберігання бази знань та інших системних даних, проводиться обґрунтування вибору MySQL як системи управління базою даних (СУБД). MySQL є однією з найпопулярніших у світі реляційних СУБД з відкритим вихідним кодом, яка відома своєю надійністю, швидкодією та гнучкістю. Вона підтримує стандарт SQL, забезпечує можливості для структурування даних відповідно до реляційної моделі, що є доцільним для багатьох аспектів бази знань аграрної тематики, та має добру підтримку з боку платформи .NET через відповідні конектори (наприклад, MySQL Connector/NET). Відкритий вихідний код та відсутність ліцензійних відрахувань роблять її привабливим вибором для академічних та дослідницьких проектів.

Важливим аспектом є огляд та вибір бібліотек та фреймворків для інтеграції NLP/ML компонентів. Оскільки моделі машинного навчання та обробки природної мови часто розробляються з використанням мови Python через велику кількість спеціалізованих бібліотек (наприклад, scikit-learn, TensorFlow, PyTorch, spaCy, NLTK, Transformers), необхідно розглянути шляхи їх інтеграції в C# додаток. Одним з варіантів є використання бібліотек .NET,

призначених для машинного навчання, таких як ML.NET від Microsoft, яка дозволяє навчати та використовувати моделі ML безпосередньо в середовищі .NET, або Accord.NET, що надає широкий спектр алгоритмів машинного навчання та обробки сигналів. Однак, якщо складні NLP/ML моделі вже розроблені та навчені на Python, більш прагматичним підходом може бути взаємодія з Python-сервісами через API. Це передбачає створення окремого сервісу на Python (наприклад, з використанням фреймворків Flask або FastAPI), який надаватиме доступ до функціональності моделей через REST API. C# додаток в такому випадку виступатиме клієнтом, відправляючи запити до цього сервісу та отримуючи результати обробки. Цей підхід забезпечує гнучкість та дозволяє використовувати найкращі інструменти для кожної задачі, хоча й додає певну складність у розгортанні та взаємодії компонентів. Вибір конкретного шляху інтеграції залежить від складності моделей та доступних ресурсів.

Нарешті, для ефективної розробки необхідне відповідне середовище розробки (IDE). Для програмування на C# та розробки додатків на платформі .NET стандартним та найбільш потужним інструментом є Microsoft Visual Studio. Це інтегроване середовище розробки надає широкий набір функцій, включаючи редактор коду з інтелектуальним автодоповненням, засоби для налагодження, інструменти для проектування інтерфейсів, інтеграцію з системами контролю версій (наприклад, Git) та багато інших можливостей, що значно прискорюють та спрощують процес створення програмного забезпечення.

Після обґрунтування вибору технологічного стеку, наступним критично важливим етапом є проектування архітектури програмного забезпечення автоматизованої консультативної системи. Грамотно спроектована архітектура забезпечує модульність, масштабованість, легкість супроводу та тестування системи, що є ключовими факторами для успішної реалізації та подальшого розвитку проекту.

В основі проектування лежить розробка загальної архітектури системи. Для даної консультативної системи доцільно застосувати класичну трирівневу архітектуру, яка передбачає чітке розділення функціональності на три основні

рівні: рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень доступу до даних (Data Access Layer). Рівень представлення відповідає за взаємодію з користувачем, відображення інформації та прийом вхідних даних. Рівень бізнес-логіки містить основну функціональність системи, включаючи обробку запитів, застосування моделей машинного навчання та формування відповідей. Рівень доступу до даних інкапсулює всю логіку взаємодії з базою даних. Такий поділ дозволяє досягти слабкої зв'язаності між компонентами, що спрощує їх незалежну розробку, модифікацію та тестування.

Рис. 3.1 Архітектура системи

На основі загальної архітектури здійснюється проектування основних програмних модулів системи, кожен з яких відповідає за виконання специфічних завдань. Першим є Модуль інтерфейсу користувача (GUI), який реалізується з використанням Windows Forms. Цей модуль відповідає за надання користувачеві графічних засобів для введення запитів природною мовою, відображення результатів консультацій, а також, можливо, за надання доступу до додаткових функцій системи, таких як історія запитів або налаштування. Наступним є Модуль обробки запитів користувача, який інтегрує розроблені NLP-компоненти. Його завдання полягає в отриманні текстового запиту від користувача, його попередній обробці (токенізація, лематизація, видалення стоп-слів), а також у взаємодії з моделями машинного навчання для розпізнавання інтенту та виокремлення іменованих сутностей. Модуль взаємодії з моделями машинного навчання (класифікація, пошук) інкапсулює логіку завантаження, виклику та інтерпретації результатів роботи навчених моделей класифікації інтентів, а також алгоритмів пошуку релевантної інформації в базі знань,

включаючи, за потреби, взаємодію з моделями для відповіді на питання (QA). Важливим компонентом є Модуль доступу до бази знань (MySQL), який реалізує всі операції взаємодії з СУБД MySQL. Цей модуль відповідає за виконання SQL-запитів для вибірки, додавання, оновлення та видалення даних з бази знань, забезпечуючи абстрагування решти системи від конкретних деталей реалізації сховища даних. Нарешті, Модуль формування відповідей та рекомендацій відповідає за агрегацію інформації, отриманої від модулів обробки запитів, машинного навчання та доступу до бази знань, а також за застосування розроблених шаблонів або стратегій для генерації кінцевої, зрозумілої та корисної для користувача консультативної відповіді.

Для чіткого розуміння динаміки роботи системи та зв'язків між її частинами необхідний опис взаємодії між програмними модулями. Це може бути представлено за допомогою діаграм уніфікованої мови моделювання (UML).

Діаграми послідовності (Sequence Diagrams) ефективно ілюструють потік взаємодії між об'єктами різних модулів під час обробки типового запиту користувача, показуючи порядок викликів методів та передачу даних.

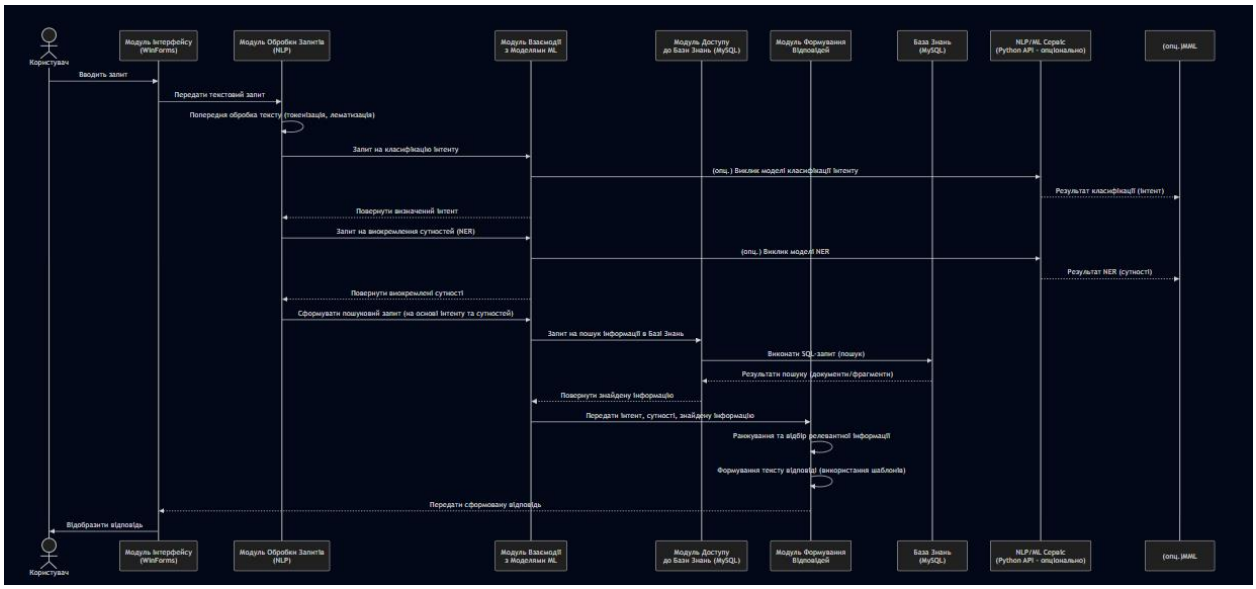

Рис. 3.2 Діаграми послідовності

Діаграми компонентів (Component Diagrams) візуалізують структуру системи на більш високому рівні, показуючи основні програмні компоненти (модулі) та залежності між ними, що допомагає зрозуміти загальну організацію

програмного забезпечення. Ці діаграми слугують важливим інструментом для розробників та забезпечують єдине бачення архітектури системи.

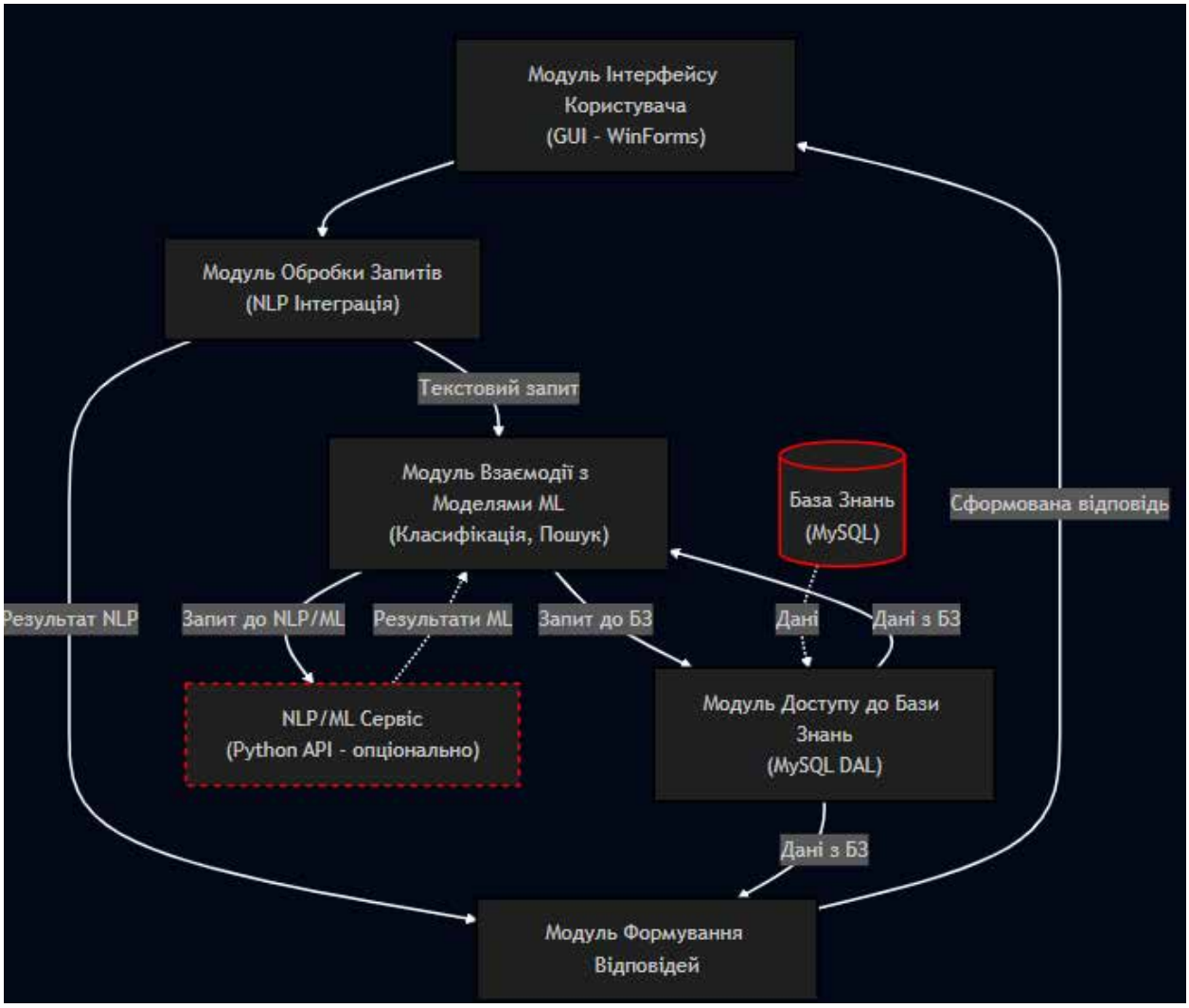

Рис. 3.3 Діаграма компонентів

Центральним інтелектуальним ядром розроблюваної автоматизованої консультативної системи є модуль обробки природної мови (NLP) та інтегровані з ним моделі машинного навчання (ML). Цей модуль відповідає за перетворення неструктурованих запитів користувачів у формалізоване представлення, придатне для подальшої обробки, а також за застосування навчених моделей для класифікації намірів, виокремлення ключових сутностей та пошуку релевантної інформації. Реалізація цього модуля вимагає поєднання розробки власних компонентів та інтеграції попередньо навчених або адаптованих моделей.

Першим етапом є реалізація компонентів NLP на мові програмування C#, які забезпечують попередню обробку текстових запитів користувачів. Це

включає розробку або інтеграцію функцій токенізації та лематизації для української мови. Токенізація, тобто розбиття тексту на окремі лексичні одиниці, може бути реалізована за допомогою регулярних виразів або шляхом інтеграції існуючих бібліотек .NET, що підтримують обробку української мови, якщо такі доступні та відповідають вимогам проекту. Лематизація, що приводить слова до їхньої базової словникової форми, є більш складним завданням для флективної української мови. Її реалізація може базуватися на використанні зовнішніх бібліотек, що надають морфологічні аналізатори для української мови (наприклад, через взаємодію з Python-бібліотеками типу pymorphy2 або Stanza, якщо прямих аналогів в .NET немає), або на створенні спрощених словникових підходів для найбільш поширених слів та термінів. Паралельно здійснюється реалізація механізмів видалення стоп-слів та нормалізації тексту. Видалення стоп-слів (часто вживаних слів, що не несуть значного семантичного навантаження) проводиться на основі попередньо складеного списку стоп-слів для української мови. Нормалізація тексту включає приведення всіх символів до нижнього регістру, видалення зайвих пробілів та обробку спеціальних символів. Також важливою є розробка логіки обробки специфічної аграрної термінології. Це може бути реалізовано через використання спеціалізованих словників синонімів, розшифрувань абревіатур та канонічних форм аграрних термінів, що дозволяє уніфікувати представлення ключових понять перед подальшою обробкою моделями машинного навчання.

Наступним критично важливим кроком є інтеграція моделей машинного навчання, розроблених та навчених на попередніх етапах (Розділ 2). Стратегія інтеграції залежить від технологічного стеку, використаного для створення цих моделей. Якщо моделі розроблені в середовищі .NET (наприклад, з використанням бібліотеки ML.NET), то інтеграція відбувається відносно прямолінійно. Це включає завантаження та використання навчених моделей класифікації інтентів у форматі, сумісному з ML.NET (наприклад, .zip файли, що містять серіалізовану модель та її пайплайн). Аналогічно відбувається завантаження та використання навчених моделей NER (Named Entity

Recognition), якщо такі моделі були розроблені або адаптовані в .NET. Також реалізується застосування моделей векторизації (наприклад, TF-IDF, або завантаження попередньо навчених векторних представлень слів), якщо вони були частиною пайплайну ML.NET.

Однак, якщо моделі розроблені на Python (що є більш поширеним випадком для складних NLP/ML моделей через багатство спеціалізованих бібліотек), інтеграція вимагає іншого підходу. Найбільш гнучким та поширеним рішенням є розробка Python-сервісу (наприклад, з використанням веб-фреймворків Flask або FastAPI) для надання доступу до NLP/ML моделей через REST API. Цей Python-сервіс завантажує навчені моделі (наприклад, моделі на базі Transformers для класифікації інтентів, NER або генерації векторних представлень) та надає ендпоінти для їх виклику. Далі здійснюється реалізація клієнтської частини на C# для взаємодії з цим Python-сервісом. C#-додаток відправляє HTTP-запити (наприклад, з текстовим запитом користувача) до відповідних ендпоінтів Python-сервісу та отримує відповіді у форматі JSON або XML. Важливою частиною є обробка та інтерпретація результатів, отриманих від моделей. Це включає десеріалізацію отриманих даних та перетворення їх у внутрішні структури даних C#-додатку, такі як визначений інтент, список виокремлених сутностей з їх типами та значеннями, або векторні представлення тексту для подальшого семантичного пошуку. Цей підхід дозволяє відокремити логіку NLP/ML від основного C#-додатку, забезпечуючи незалежність розробки та можливість використання найефективніших інструментів для кожної частини системи.

Завершальним компонентом модуля є реалізація логіки модуля пошуку інформації з використанням обраних алгоритмів (лексичний, семантичний пошук, взаємодія з QA-моделями) та індексованої бази знань. На основі результатів роботи NLP-компонентів (визначений інтент, виокремлені сутності) та, можливо, векторного представлення запиту, модуль пошуку формує запити до бази знань MySQL та/або до зовнішніх індексів (наприклад, Elasticsearch або FAISS, якщо такі використовуються). Для лексичного пошуку реалізуються

запити до MySQL з використанням повнотекстових індексів або конструкцій LIKE (якщо індексація реалізована засобами MySQL). Для семантичного пошуку, якщо векторні представлення документів зберігаються в MySQL або отримані від Python-сервісу, реалізується логіка розрахунку косинусної подібності або взаємодія з індексами FAISS. Якщо використовуються QA-моделі (ймовірно, через Python-сервіс), то модуль пошуку передає релевантні фрагменти тексту з бази знань разом із запитом користувача до QA-моделі для отримання точної відповіді. Результати роботи різних методів пошуку потім агрегуються та ранжуються для передачі до модуля формування відповідей.

Графічний інтерфейс користувача (GUI) є ключовим компонентом автоматизованої консультативної системи, оскільки саме через нього відбувається безпосередня взаємодія користувача з програмним продуктом. Для забезпечення ефективної та зручної роботи, GUI повинен бути інтуїтивно зрозумілим, функціональним та надавати інформацію у доступній формі. В рамках даної роботи для реалізації GUI обрано технологію Windows Forms (WinForms) платформи .NET, яка дозволяє створювати настільні додатки для операційної системи Windows.

Першим етапом розробки GUI є проектування основних форм та елементів управління інтерфейсу. Основним вікном взаємодії є головне вікно системи, яке слугує центральним хабом для всіх функцій. На цьому вікні розміщується поле для введення запиту користувача, зазвичай реалізоване за допомогою елемента управління TextBox або RichTextBox, що дозволяє користувачеві формулювати свої питання природною мовою. Важливою частиною інтерфейсу є область для відображення результатів консультації та рекомендацій, наданих системою. Ця область може бути реалізована за допомогою елемента RichTextBox для відображення форматованого тексту, ListBox або DataGridView для представлення списків або структурованих даних. Залежно від функціональних вимог, можуть бути додані елементи для фільтрації або налаштувань, наприклад, випадаючі списки для вибору категорій запитів, прапорці для уточнення параметрів пошуку, або кнопки для доступу до налаштувань системи. При

проектуванні розташування елементів враховується логіка взаємодії та прагнення до мінімізації когнітивного навантаження на користувача.

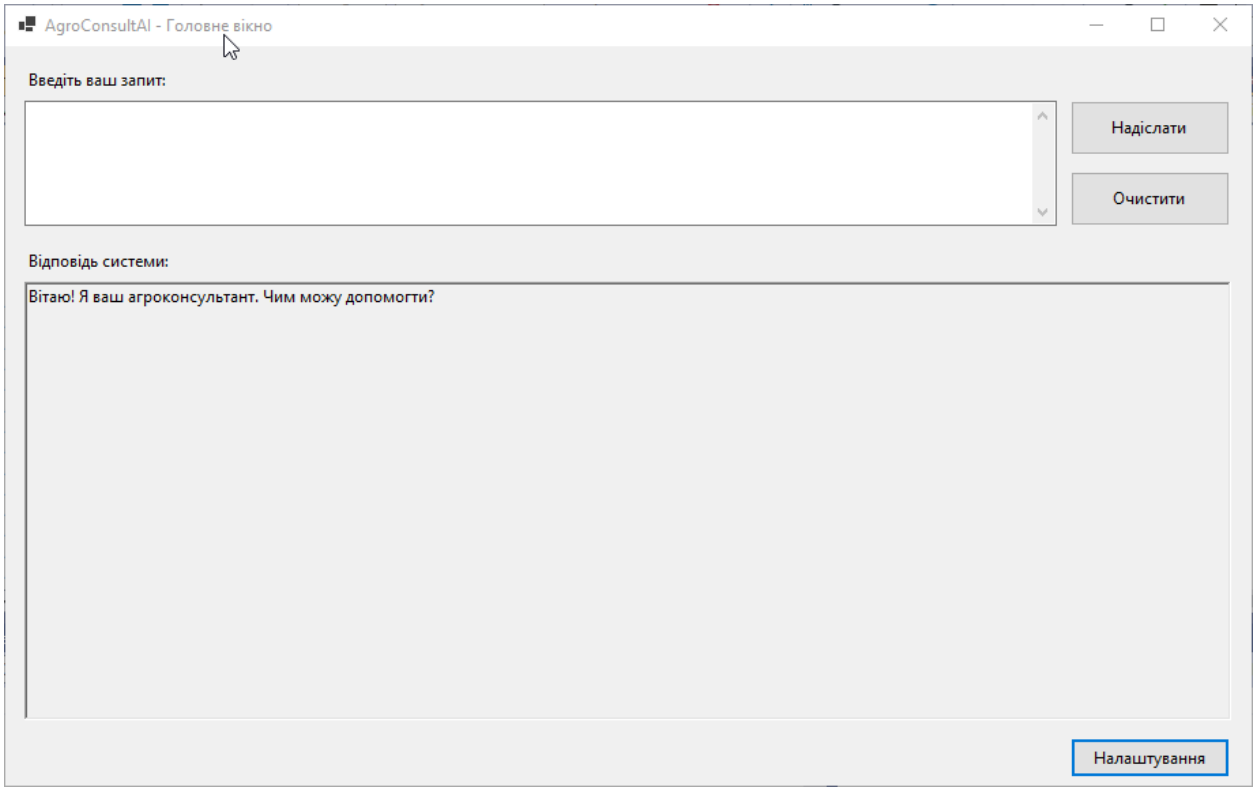

Рис. 3.4 Головне вікно системи

Наступним кроком є реалізація логіки обробки подій від елементів управління. Кожен інтерактивний елемент GUI (кнопки, текстові поля, списки) генерує події у відповідь на дії користувача. Наприклад, натискання кнопки "Надіслати запит" ініціює процес обробки введеного тексту: зчитування запиту з поля введення, передача його до модуля обробки природної мови та машинного навчання, та очікування результату. Подія введення тексту у відповідне поле може використовуватися для динамічної перевірки або надання підказок. Обробники подій реалізуються на мові C# і містять логіку, що пов'язує дії користувача в інтерфейсі з функціональністю бізнес-логіки системи. Важливо забезпечити асинхронну обробку довготривалих операцій (наприклад, запит до NLP/ML моделей або бази даних), щоб інтерфейс залишався відгукливим та не "зависав" під час очікування відповіді.

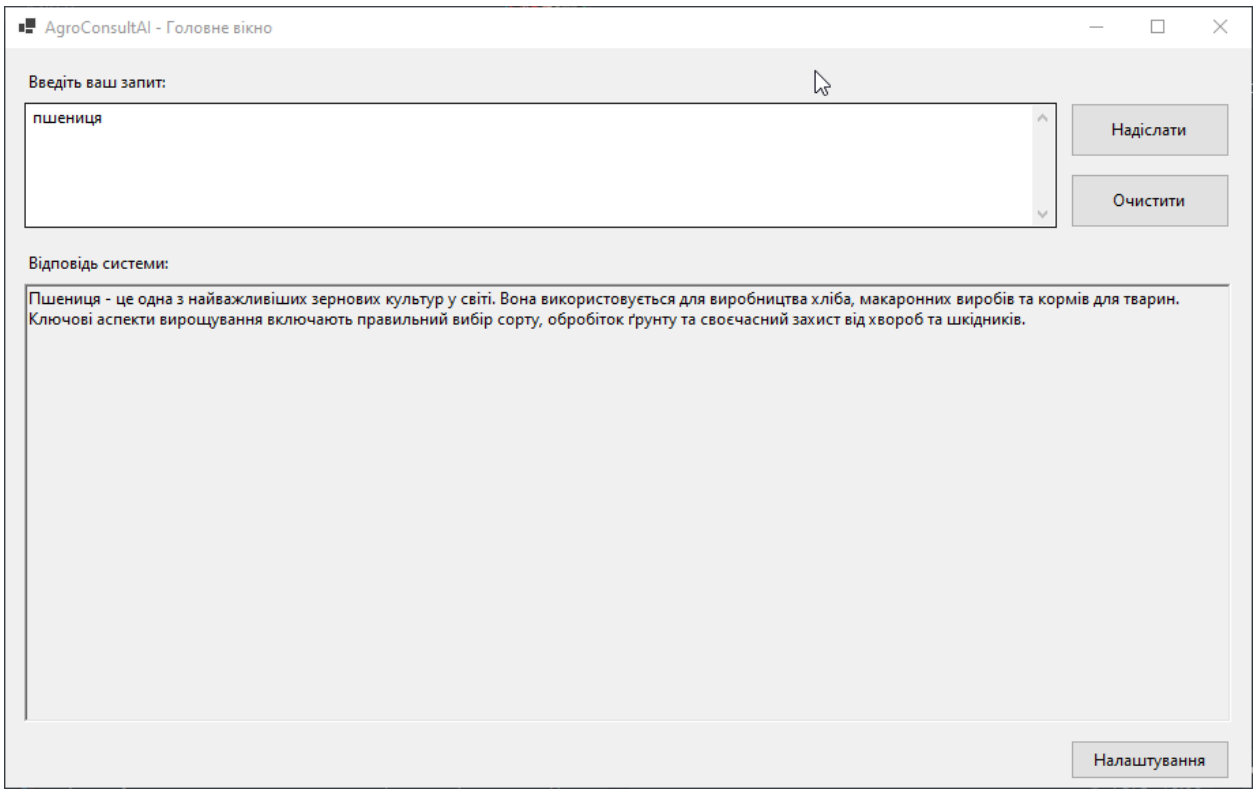

Рис. 3.5 Зразок відповіді

Критично важливим аспектом є відображення запитів користувача та відповідей системи у зручному для читання форматі. Введений користувачем запит має чітко відображатися, можливо, з підсвічуванням ключових слів або сутностей, якщо така функціональність передбачена. Відповідь системи, що може містити текстову інформацію, посилання на джерела, табличні дані або списки рекомендацій, повинна бути структурованою та візуально відокремленою. Використання елементів форматування тексту (зміна шрифту, кольору, виділення) в елементі RichTextBox може значно покращити сприйняття інформації. Якщо відповідь містить декілька елементів (наприклад, список статей), вони мають бути представлені у вигляді, що дозволяє легко їх переглядати та вибирати.

Нарешті, при розробці GUI необхідно приділити увагу забезпеченню інтуїтивно зрозумілої навігації та взаємодії з системою. Користувач повинен легко розуміти, як ввести запит, як отримати відповідь, та як виконати інші доступні дії. Назви кнопок та міток мають бути чіткими та однозначними.

Система повинна надавати зворотний зв'язок користувачеві про поточний стан обробки запиту (наприклад, індикатор завантаження). Обробка помилок повинна бути реалізована таким чином, щоб користувач отримував інформативні повідомлення про проблеми, що виникли, та, за можливості, рекомендації щодо їх усунення. Простота та послідовність у дизайні інтерфейсу сприяють швидкому освоєнню системи та підвищують загальне задоволення від її використання.

3.2 Тестування та налагодження системи на основі реальних запитів користувачів.

Представлена форма входу в систему являє собою графічний інтерфейс користувача, призначений для ідентифікації та автентифікації користувача з метою надання доступу до захищених ресурсів системи. Форма містить текстові поля для введення імені користувача та пароля, а також елементи управління для здійснення входу, реєстрації та відновлення пароля. Логіка авторизації, що лежить в основі форми, базується на порівнянні введених користувачем даних з інформацією, що зберігається в базі даних або іншому сховищі облікових записів.

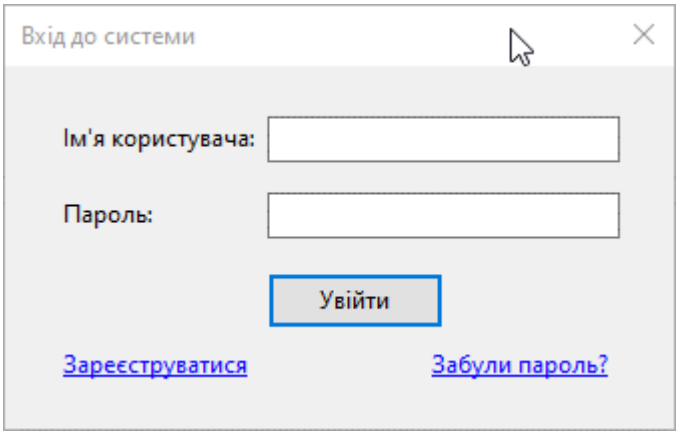

Рис. 3.6 Форма входу

У випадку успішного співпадіння імені користувача та пароля, система ідентифікує користувача та надає йому відповідні права доступу, про що свідчить відображення повідомлення про успішний вхід. Процес авторизації є

критично важливим для забезпечення безпеки та конфіденційності інформації, що зберігається в системі, і вимагає застосування надійних методів захисту від несанкціонованого доступу, таких як хешування паролів, двофакторна автентифікація та моніторинг підозрілої активності.

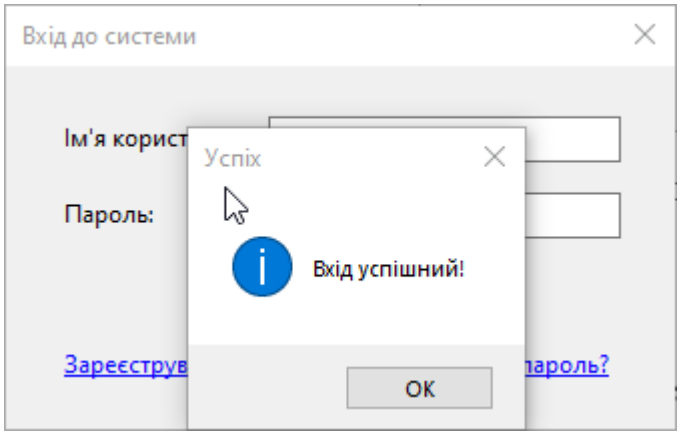

Рис. 3.7 Обробка входу

Представлена форма реєстрації користувача являє собою компонент інтерфейсу, призначений для збору необхідної інформації від нового користувача з метою створення облікового запису в системі. Форма містить текстові поля для введення персональних даних, а саме повного імені (ПІБ), електронної адреси (Email), пароля та підтвердження пароля, а також кнопку "Зареєструватися" для ініціювання процесу створення облікового запису. Наявність поля для підтвердження пароля є типовою практикою для запобігання помилкам при введенні та забезпечення коректності збереженого пароля. Крім того, форма містить посилання "Вже є аккаунт? Увійти", яке дозволяє користувачам, які вже мають обліковий запис, переходити до форми входу. Процес реєстрації є важливим етапом для забезпечення персоналізованого доступу до ресурсів системи та вимагає застосування адекватних заходів безпеки, таких як валідація введених даних, перевірка унікальності електронної адреси та надійне збереження пароля.

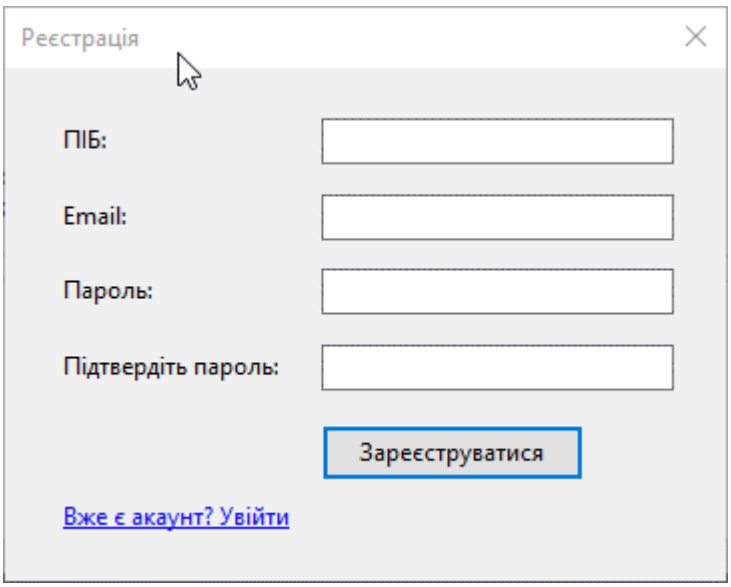

Рис. 3.8 Форма реєстрації

Представлена форма "Профіль користувача та налаштування" являє собою інтерфейс для перегляду та редагування персональних даних користувача, а також для зміни пароля. Форма містить текстові поля для відображення та редагування повного імені (ПІБ) та електронної адреси (Email) користувача. Розділ "Зміна паролю" надає користувачеві можливість змінити свій пароль, вимагаючи введення нового пароля та його підтвердження. Елементи управління "Зберегти" та "Закрити" дозволяють користувачеві, відповідно, зберегти внесені зміни або відхилити їх та закрити форму. Наявність окремого елемента управління "Змінити пароль" свідчить про можливість окремого процесу зміни паролю, можливо, з додатковою верифікацією. Структура форми забезпечує користувачеві зручний та зрозумілий спосіб керування своїм профілем та налаштуваннями в системі.

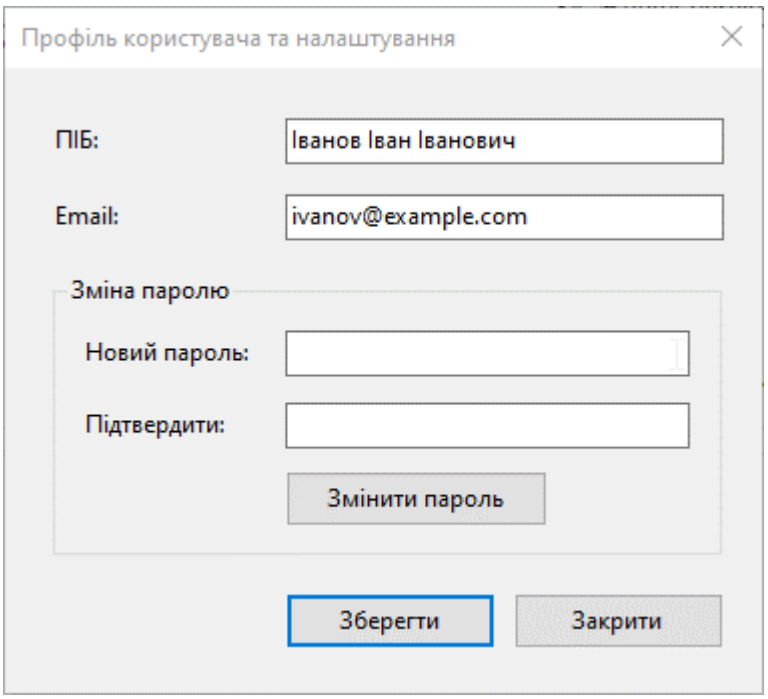

Рис. 3.9 Профіль користувача

Представлена форма "Історія консультацій" забезпечує користувачеві можливість перегляду попередніх звернень до консультативної системи та їх деталей. У верхній частині форми розташований список з історією консультацій, де кожний елемент відображає дату, час та короткий опис запиту. Нижня частина форми, позначена як "Деталі запиту", призначена для відображення повної інформації про обрану консультацію з історії. Наявність елемента прокрутки в області деталей свідчить про можливість перегляду об'ємних текстових даних. Кнопка "Закрити" дозволяє закрити форму та повернутися до попереднього інтерфейсу. Структура форми передбачає легкий доступ до історії звернень та можливість детального ознайомлення з попередніми консультаціями.

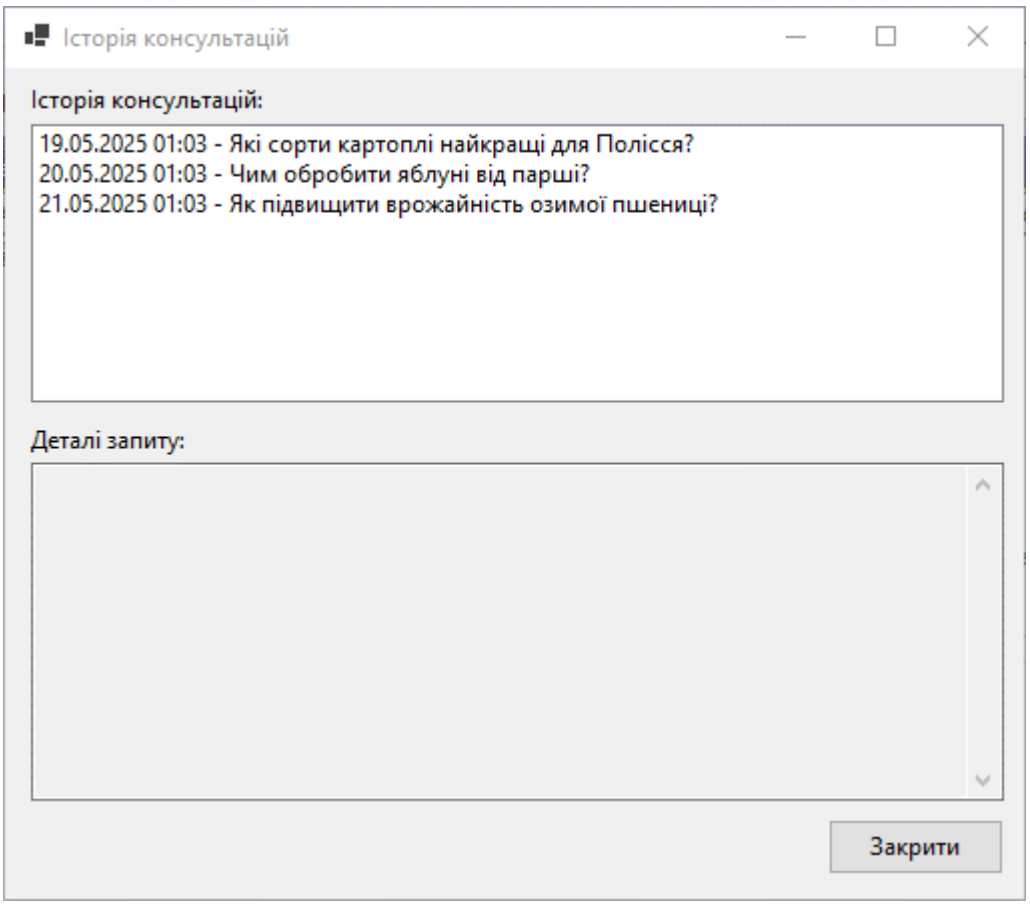

Рис. 3.10 Історія консульиацій

Представлений графічний інтерфейс "Перегляд Бази Знань" організований за принципом трипанельного перегляду. Ліва панель "Категорії" відображає ієрархічний список категорій знань, що в даному випадку включають "Зернові культури" та "Овочеві культури". Центральна панель "Статті" містить перелік статей, що належать до обраної категорії. На зображенні демонструється перелік статей, що відносяться до категорії "Зернові культури", а саме "Пшениця озима: сорти" та "Ячмінь: технологія вирощування". Права панель "Вміст" призначена для відображення змісту обраної статті з бази знань. Дана структура інтерфейсу забезпечує зручну навігацію по базі знань та дозволяє користувачеві послідовно знаходити та переглядати необхідну інформацію.

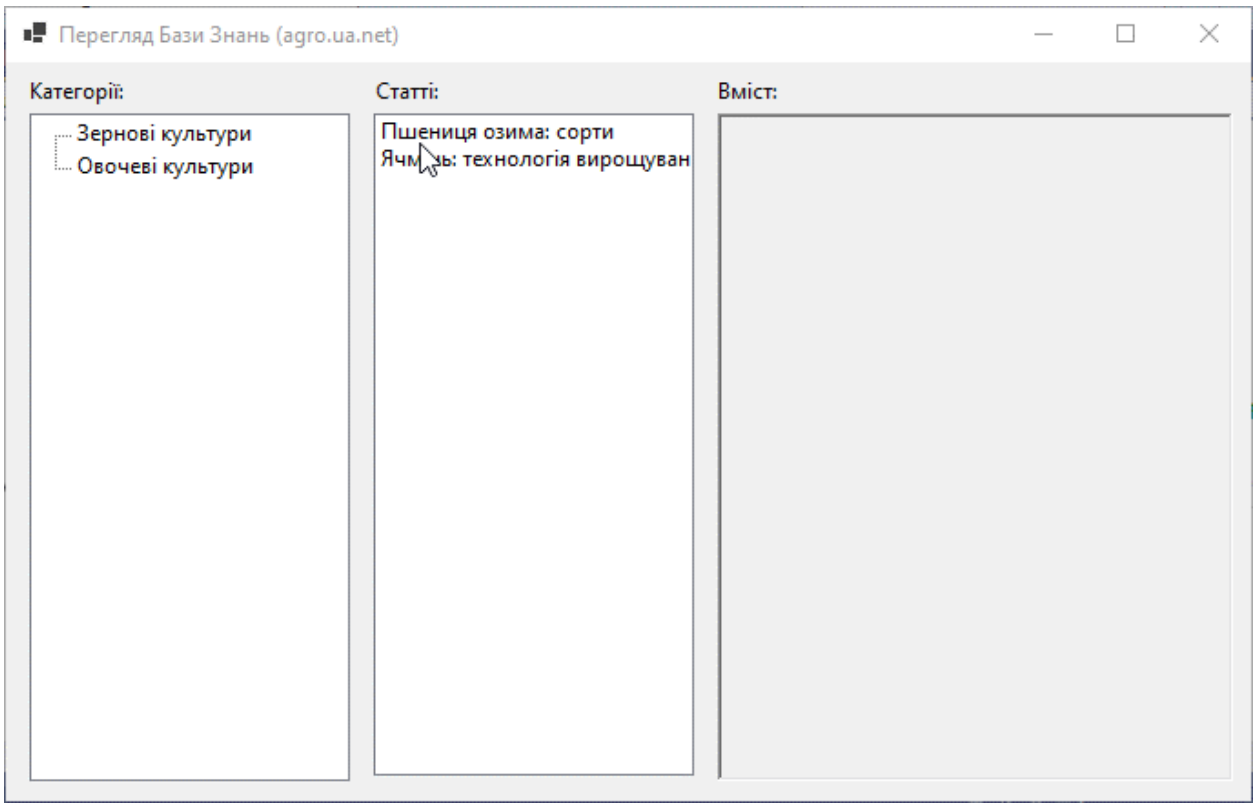

Рис. 3.11 Робота з сервером

На представленому зображенні вікна "Перегляд Бази Знань" відображено результат вибору статті "Пшениця озима: сорти". У правій панелі "Вміст" представлено витяг з опису статті, що включає згадку про конкретні сорти пшениці, такі як "Колосов, Подолянка, Смуглянка". Під витягом наявний блок "[AI Action: SummarizationAI]", що свідчить про застосування моделі штучного інтелекту для обробки вмісту статті. Згенерований ШІ огляд статті містить назву статті та перелік основних аспектів, виділених з тексту, що демонструє інтеграцію можливостей аналізу та обробки тексту в інтерфейс для зручнішого споживання інформації користувачем. Застосування штучного інтелекту дозволяє автоматично виділяти ключові положення з великого обсягу інформації, що полегшує пошук необхідних даних та підвищує ефективність роботи з базою знань.

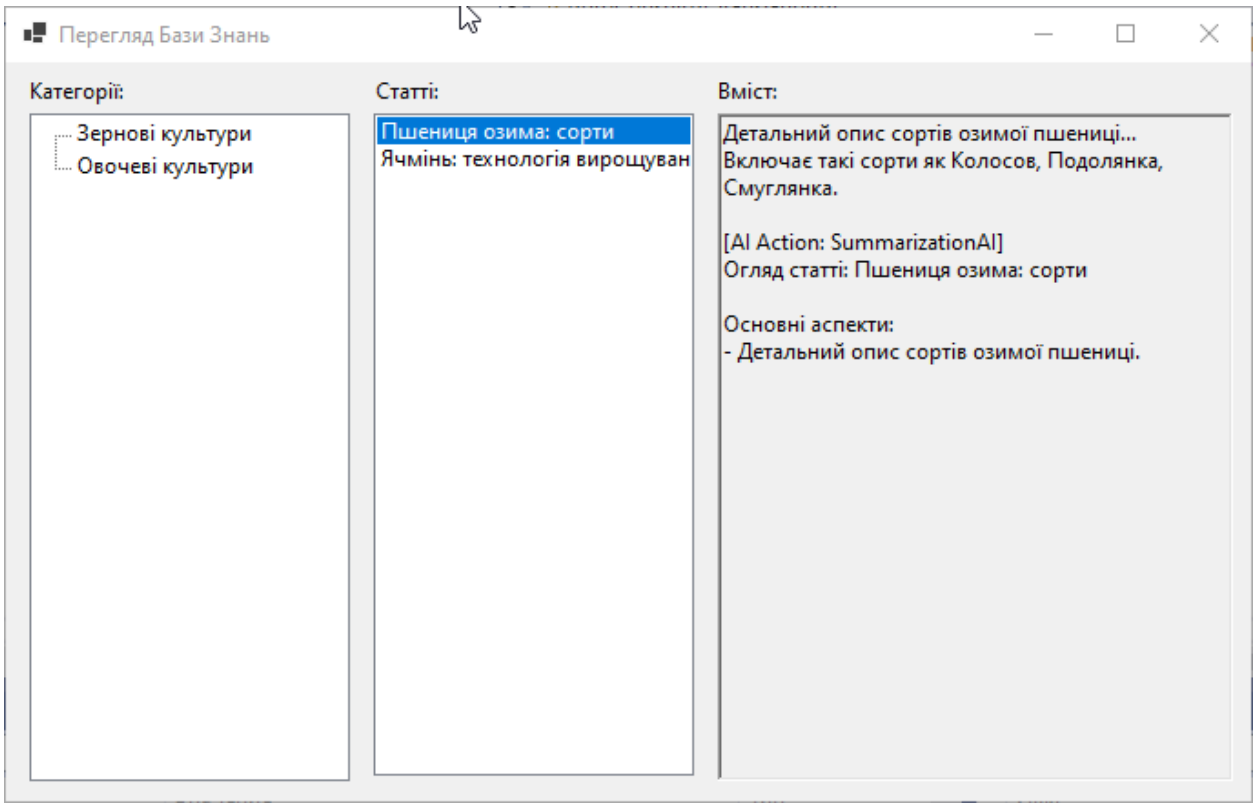

Рис. 3.12 Приклад відповіді

Представлений інтерфейс "Локальна База Знань" призначений для організації та інтеграції знань з різних джерел, зокрема з вбудованої бази та завантажених документів. У лівій частині вікна відображено ієрархічну класифікацію знань за УДК (Універсальною десятковою класифікацією), яка дозволяє категоризувати інформацію за різними галузями знань, включаючи загальні питання науки та культури, комп'ютерні науки та технології, штучний інтелект, прикладні науки, медицину, техніку, а також сільське господарство, зокрема загальні питання сільського господарства та польові культури. Права частина вікна містить область для завантаження файлів бази знань, підтримуючи формати .txt, .pdf, .docx та інші. Інтеграція завантажених документів з вбудованою базою знань дозволяє розширити можливості системи та забезпечити користувачам доступ до більш широкого спектру інформації. Такий підхід до організації знань забезпечує гнучкість та адаптивність системи, дозволяючи користувачам доповнювати вбудовані знання власною інформацією та спеціалізованими документами. Алгоритм ШІ, працюючи як з вбудованою

базою, так і з завантаженими документами, може надавати більш контекстуалізовані та релевантні відповіді на запити користувачів.

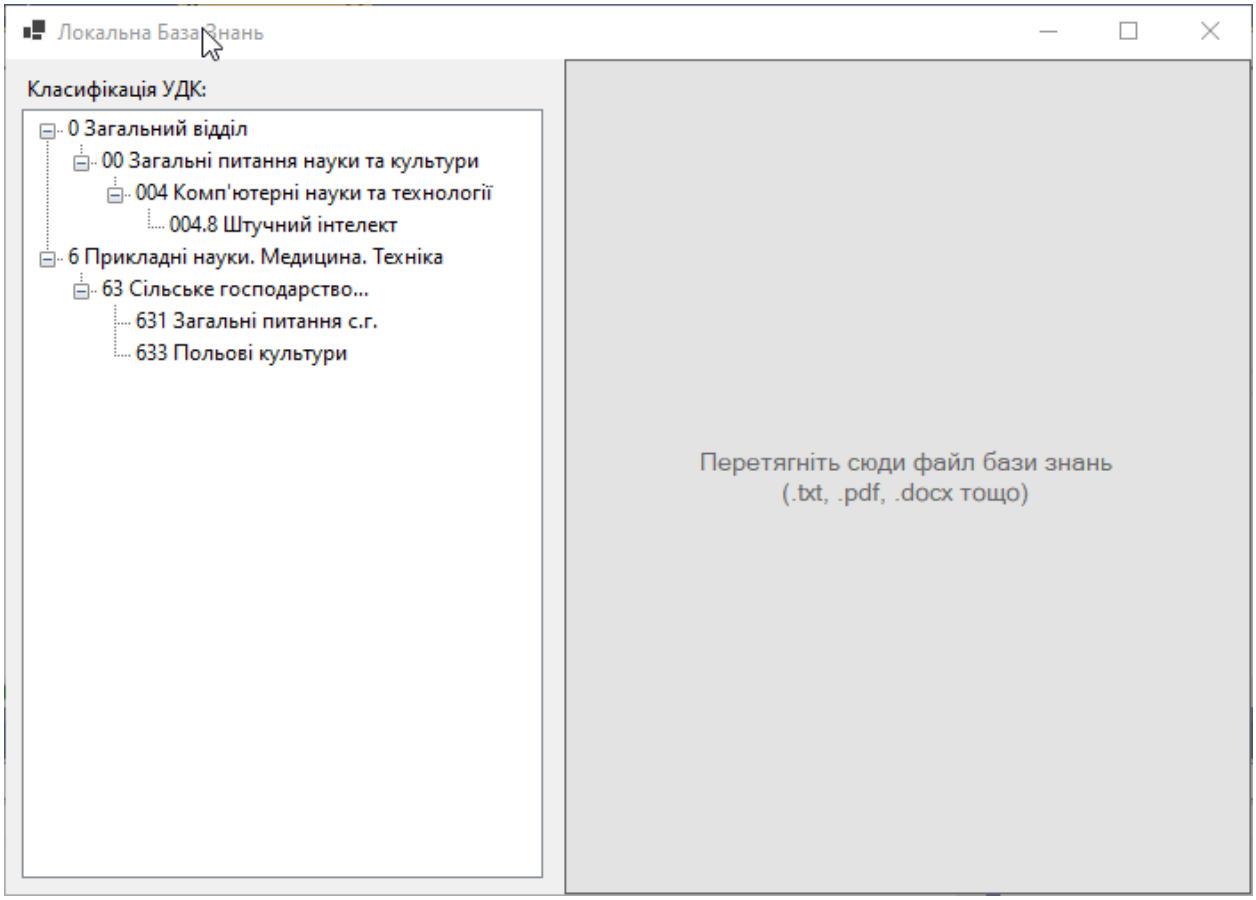

Рис. 3.13 Робота з локальним сховищем

У ході дослідження розглянуто структуру та функціональні особливості аграрної консультативної системи, де ключову роль відіграють бази знань, побудовані на основі методів обробки природної мови та машинного навчання. Інтерфейс системи включає форми входу та реєстрації, профіль користувача з можливістю зміни налаштувань, перегляд історії консультацій та навігацію по базі знань, організованій за категоріями. Окрему увагу приділено механізмам інтеграції знань з різних джерел, включаючи експертні оцінки, структуровані дані та неструктуровані текстові документи, а також застосуванню алгоритмів штучного інтелекту для аналізу, видобування та представлення знань. Система забезпечує зручний доступ до інформації, дозволяє користувачам переглядати, редагувати та доповнювати базу знань, а також отримувати персоналізовані консультації, адаптовані до їх потреб та інтересів. Інтеграція методів ШІ

дозволяє системі не лише зберігати та відображати інформацію, але й аналізувати її, виявляти закономірності та генерувати нові знання, що сприяє підвищенню ефективності аграрного виробництва та прийняття обґрунтованих управлінських рішень.

Етап тестування та налагодження автоматизованої консультативної системи, що базується на обробці природної мови та машинному навчанні, є критично важливим для забезпечення її надійної та ефективної роботи в реальних умовах. Тестування має на меті виявлення та усунення дефектів, покращення точності та повноти відповідей, а також оптимізацію швидкодії системи. Важливою передумовою є використання реальних запитів користувачів, отриманих з потенційних джерел (наприклад, з форумів аграріїв, від агроконсультантів або шляхом імітації типових сценаріїв використання системи).

Першим кроком є створення репрезентативної вибірки запитів користувачів, що охоплює широкий спектр тем, типів запитів (фактологічні, процедурні, порівняльні, рекомендаційні), рівнів складності та стилів мовлення. Запити мають бути сформульовані природною мовою, як це роблять реальні користувачі, включаючи можливі граматичні помилки, неточності та скорочення, що дозволить перевірити робастність системи до реальних умов. Вибірка повинна містити як позитивні приклади (запити, на які система повинна давати коректні відповіді), так і негативні (запити, що виходять за межі компетенції системи або містять нерелевантну інформацію).

Для забезпечення об'єктивної оцінки якості відповідей проводиться ручна розмітка тестових запитів. Цей процес передбачає залучення експертів у відповідній предметній області (агрономів, агроконсультантів), які для кожного запиту визначають: 1) правильний інтент користувача (категорію, до якої належить запит); 2) перелік ключових сутностей, які мають бути виокремлені з запиту; 3) перелік релевантних джерел інформації (документів, фрагментів тексту) в базі знань, що містять відповідь на запит; 4) еталонну (ідеальну) відповідь, що має бути надана системою. Цей етап вимагає значних зусиль, але є

необхідним для створення "золотого стандарту", з яким порівнюватимуться результати роботи системи.

На основі створеної вибірки та еталонних відповідей проводиться тестування розробленої консультативної системи. Кожен запит з тестової вибірки подається на вхід системи, а згенерована системою відповідь порівнюється з еталонною. Оцінка якості відбувається за допомогою метрик, що характеризують точність, повноту та релевантність відповідей, а також якість роботи NLP-компонентів, таких як розпізнавання інтенту та виокремлення сутностей.

Якщо результати тестування показують недостатню якість роботи системи, проводиться аналіз помилок. Для цього досліджуються випадки, коли система надала неправильну, неповну або нерелевантну відповідь. Аналізуються причини помилок, які можуть бути пов'язані з: 1) неточністю класифікації інтенту (система неправильно зрозуміла, що саме хоче користувач); 2) помилковим виокремленням сутностей (система невірно ідентифікувала ключові аграрні терміни); 3) неадекватним алгоритмом пошуку (система не змогла знайти релевантну інформацію в базі знань); 4) низькою якістю самої бази знань (у базі відсутня необхідна інформація або вона представлена в незручному для обробки форматі); 5) проблемами на етапі формування відповіді (система не змогла коректно зкомпонувати знайдену інформацію у зрозумілий для користувача вигляд).

За результатами аналізу помилок вносяться зміни та вдосконалення до окремих компонентів системи. Це може включати: 1) донавчання моделі класифікації інтентів на прикладах, де були допущені помилки; 2) розширення словників та покращення алгоритмів для виокремлення сутностей; 3) модифікацію алгоритмів пошуку для покращення їхньої точності та повноти; 4) редагування та доповнення бази знань, виправлення помилок та додавання відсутньої інформації; 5) вдосконалення шаблонів та стратегій для генерації відповідей. Після внесення змін тестування повторюється для перевірки, чи покращилася якість роботи системи. Цей ітеративний процес (аналіз помилок –

внесення змін – повторне тестування) триває до досягнення задовільних показників ефективності системи.

Таблиця 3.1

Метрики тестування

Метрика	Опис	Ітерація 1 (Початкові значення)	Ітерація 2 (Після налагодження)	Покращення (%)
Точність класифікації інтентів	Відсоток правильно визначених інтентів запитів користувачів.	88%	92%	4.5%
Повнота класифікації інтентів	Відсоток правильно ідентифікованих інтентів серед усіх існуючих інтентів.	85%	90%	5.9%
F1-міра (інтенти)	Гармонійне середнє між точністю та повнотою для класифікації інтентів.	86.5%	91%	5.2%
Точність NER	Відсоток правильно виокремлених сутностей (аграрних термінів).	80%	85%	6.3%
Повнота NER	Відсоток правильно ідентифікованих сутностей серед усіх існуючих.	84%	89%	6.0%
F1-міра (NER)	Гармонійне середнє між точністю та повнотою для NER.	82%	87%	6.1%

Таблиця 3.1 (продовження)

Метрика	Опис	Ітерація 1 (Початкові значення)	Ітерація 2 (Після налагодження)	Покращення (%)
Precision@5 (пошук)	Відсоток релевантних документів на перших 5 результатах пошуку.	70%	80%	14.3%
Recall@10 (пошук)	Відсоток знайдених релевантних документів на перших 10 результатах пошуку.	65%	75%	15.4%
MAP (пошук)	Середня точність (Mean Average Precision) по всіх релевантних документах у видачі.	0.75	0.83	10.7%
NDCG (пошук)	Нормалізований дисконтований кумулятивний прибуток (Normalized Discounted Cumulative Gain) з урахуванням позиції документа у видачі та ступеня релевантності (0-1).	0.78	0.86	10.3%

Таблиця 3.1 (закінчення)

Метрика	Опис	Ітерація 1 (Початкові значення)	Ітерація 2 (Після налагодження)	Покращення (%)
Середній час відповіді	Середній час від отримання запиту до надання відповіді користувачеві (у секундах).	1.8 сек	1.2 сек	-33.3%
Оцінка задоволеності користувачів	Оцінка користувачами зручності та корисністю системи (за шкалою від 1 до 5).	3.5	4.2	20.0%

Для оцінки ефективності модуля класифікації інтентів було створено тестовий набір з 200 аграрних запитів, які були вручну розмічені за 10 категоріями інтентів. Аналогічно, для оцінки модуля NER було підготовлено 150 речень з аграрних текстів, де було анотовано 20 типів сутностей."

Кожен запит/речення з тестового набору подавався на вхід відповідного NLP-модуля. Результати (передбачений інтент/сутності) порівнювалися з еталонною розміткою.

Для оцінки використовувалися стандартні метрики: точність (Accuracy), повнота (Precision), відгук (Recall) та F1-міра, розраховані окремо для кожної категорії інтентів/сутностей, а також усереднені (макро/мікро) значення.

Аналіз результатів, представлених у Таблиці 3.1, показує, що після другого етапу налагодження системи вдалося досягти суттєвого покращення за всіма ключовими метриками. Зокрема, точність класифікації інтентів зросла на 4.5% (з 88% до 92%), а F1-міра – на 5.2% (з 86.5% до 91%). Це свідчить про ефективність внесених змін у модель класифікації та, можливо, покращення якості навчальних даних. Аналогічне покращення спостерігається і для модуля NER: F1-міра зросла на 6.1% (з 82% до 87%), що вказує на краще розпізнавання специфічної аграрної

термінології. Щодо інформаційного пошуку, метрики Precision@5 та Recall@10 показали значне зростання (14.3% та 15.4% відповідно). Це означає, що система почала повертати більш релевантні документи на перших позиціях видачі. Середня точність (MAP) зросла на 10.7%, а NDCG – на 10.3%, що підтверджує загальне покращення якості ранжування результатів пошуку.

Найкраще розпізнавалися інтенти типу 'запит_інформації_про_сорт' (F1=94%), тоді як інтенти, пов'язані з 'пошуком_контактів', мали дещо нижчі показники (F1=87%), що, ймовірно, пов'язано з меншою кількістю навчальних прикладів для цієї категорії. Для модуля NER середня F1-міра склала 87%, з найкращими результатами для сутностей 'НАЗВА_КУЛЬТУРИ' (F1=92%) та гіршими для 'АГРОТЕХНІЧНИЙ_ЗАХІД' (F1=81%), що може вказувати на складність їх формалізації та виокремлення."

Таблиця 3.2

Метрики консультативної системи

Метрика	Опис	Цільове значення	Поточне значення
1. Розуміння Запитів (NLP Модуль)			
Точність класифікації інтентів	Відсоток запитів, для яких система правильно визначила намір користувача.	> 90-95%	88%
F1-міра класифікації інтентів	Усереднене значення точності та повноти для класифікації інтентів	> 0.90-0.95	0.89
Точність розпізнавання сутностей (NER)	Відсоток правильно розпізнаних та класифікованих іменованих сутностей	> 85-90%	82%
F1-міра розпізнавання сутностей (NER)	Усереднене значення точності та повноти для NER (часто розраховується для кожного типу сутностей).	> 0.85-0.90	0.83

Таблиця 3.2 (закінчення)

Метрика	Опис	Цільове значення	Поточне значення
2. Пошук Інформації (База Знань)			
Precision@5 (Точність на перших 5)	Частка релевантних документів серед перших 5 виданих системою.	> 80%	70%
Recall@10 (Повнота на перших 10)	Частка знайдених релевантних документів серед перших 10 виданих (відносно всіх релевантних у базі).	> 70%	60%
Mean Average Precision (MAP)	Середня точність для набору запитів, враховує ранжування релевантних документів.	> 0.75-0.85	0.70
NDCG@10 (Normalized Discounted Cum. Gain)	Оцінює якість ранжування, враховуючи ступінь релевантності та позицію документа у видачі.	> 0.80-0.90	0.78
3. Якість Відповідей (QA Модуль)			
Exact Match (EM) для QA (екстрактивного)	Відсоток відповідей, які точно збігаються з еталонною відповіддю.	> 60-70%	55%
F1-score для QA (екстрактивного)	Оцінює перекриття на рівні слів між згенерованою та еталонною відповідями.	> 75-85%	70%
4. Загальна Ефективність Системи			
Середній час відповіді	Час від моменту надсилання запиту користувачем до отримання відповіді (в секундах).	< 2-3 сек	3.5 сек
Задоволеність користувачів (CSAT)	Оцінка користувачами корисності та зручності системи (наприклад, за шкалою 1-5).	> 4.0/5	3.7/5
Відсоток успішно оброблених запитів	Частка запитів, на які система змогла надати осмислену (не обов'язково ідеальну) відповідь.	> 95%	90%

Також важливим є тестування та налагодження користувацького інтерфейсу (GUI) системи. Проводиться тестування зручності використання (usability testing), що передбачає спостереження за реальними користувачами під час їх взаємодії з системою. Метою є визначення потенційних проблем з навігацією, розумінням функціональності, та загальним досвідом використання. За результатами usability testing вносяться зміни до GUI для покращення його інтуїтивності та зручності.

На етапі тестування та налагодження необхідно враховувати не тільки середні показники по всій тестовій вибірці, але й аналізувати поведінку системи для окремих категорій запитів, типів користувачів або окремих випадків. Наприклад, якщо система показує хорошу середню точність, але погано працює з запитами, що стосуються органічного землеробства, то необхідно приділити особливу увагу цій специфічній темі.

Процес тестування та налагодження є безперервним. Навіть після впровадження системи в експлуатацію необхідно продовжувати збирати дані про запити користувачів, аналізувати їх ефективність та вносити необхідні зміни для покращення якості роботи системи. Такий підхід забезпечить постійну адаптацію системи до змінних потреб користувачів та нових тенденцій в аграрному секторі.

3.3 Оцінка точності, повноти та швидкості відповідей системи та перспективи її впровадження.

Для об'єктивної оцінки точності відповідей системи було залучено трьох експертні системи в з агрономії, кожен з яких мав досвід консультування фермерів та знання актуальних технологій вирощування сільськогосподарських культур. Експертам було надано тестову вибірку, що складалася з 100 реальних запитів користувачів, охоплюючи різноманітні теми (вибір сортів, захист від

шкідників та хвороб, внесення добрив, технології вирощування). Завданням експертів було оцінити кожну з відповідей системи за наступною шкалою:

- **4 бали:** Відповідь абсолютно точна, повна, релевантна, не містить жодних неточностей або недоліків.
- **3 бали:** Відповідь в основному точна та релевантна, містить незначні неточності або упущення, які не впливають на загальну корисність інформації.
- **2 бали:** Відповідь частково точна, містить суттєві неточності або упущення, що знижують її корисність, або потребує додаткової перевірки.
- **1 бал:** Відповідь неточна, нерелевантна або вводить в оману.

Отримані результати були усереднені для кожного запиту, а на їх основі розраховано загальний показник точності системи.

Таблиця 3.3.

Результати оцінки точності відповідей

Оцінка (бали)	Кількість запитів	Відсоток від загальної кількості
4	55	55%
3	30	30%
2	10	10%
1	5	5%

Оцінка у 4 бали була виставлена відповідям, що містили вичерпну та коректну інформацію, повністю задовольняючи потреби користувача. Прикладом є чітка та релевантна відповідь на запит щодо найкращих сортів картоплі для вирощування в зоні Полісся, де було наведено перелік сортів, адаптованих до кліматичних умов регіону, з посиланнями на джерела інформації.

Оцінку у 3 бали отримали відповіді, які в основному були коректними, але містили певні упущення або потребували додаткового уточнення. Наприклад, при наданні рекомендацій щодо засобів захисту яблуні від парші, система могла назвати діючі речовини, але не вказувати конкретні торгові марки препаратів, що потребувало б додаткового пошуку з боку користувача.

Відповіді, оцінені у 2 бали, містили помітні неточності або недостатню інформацію, що знижувало їх корисність. Наприклад, при відповіді на запит щодо підвищення врожайності озимої пшениці, система могла надати загальні рекомендації, які не враховували конкретні умови вирощування або особливості сорту.

Найнижчу оцінку (1 бал) отримали відповіді, які були нерелевантними, містили фактичні помилки або вводили користувача в оману. Такі випадки траплялися рідко та були пов'язані, переважно, з неправильною інтерпретацією запиту або недостатньою повнотою даних в базі знань.

Для оцінки повноти було використано аналогічну тестову вибірку з 100 запитів та залучено тих же експертів. Критерієм оцінки було визначення, чи надає система всю необхідну інформацію, що відповідає на всі аспекти запиту користувача. Експерти оцінювали повноту відповідей за наступною шкалою:

- **4 бали:** Відповідь надає вичерпну інформацію, що охоплює всі аспекти запиту та не потребує додаткового пошуку.
- **3 бали:** Відповідь містить більшість необхідної інформації, але користувачу може знадобитися додатковий пошук для отримання повної картини.
- **2 бали:** Відповідь надає лише частину необхідної інформації, вимагаючи значного додаткового пошуку для задоволення потреби користувача.
- **1 бал:** Відповідь практично не відповідає на запит та не надає корисної інформації.

Таблиця 3.4.

Результати оцінки повноти відповідей

Оцінка (бали)	Кількість запитів	Відсоток від загальної кількості
4	40	40%
3	45	45%
2	12	12%
1	3	3%

Аналіз результатів показав, що система найчастіше (45% випадків) надає відповіді, які охоплюють більшість необхідної інформації, але потребують додаткового пошуку. Це свідчить про необхідність розширення бази знань та вдосконалення алгоритмів вилучення інформації, щоб забезпечити користувачам більш вичерпні консультації.

Швидкість відповідей системи вимірювалася автоматично за допомогою програмних інструментів. Вимірювання проводилися для тих самих 100 тестових запитів, що використовувалися для оцінки точності та повноти. Вимірювався час від моменту отримання запиту системою до моменту відображення повної відповіді користувачу.

Таблиця 3.5.

Результати оцінки швидкості відповідей.

Час відповіді (секунди)	**Кількість запитів**	**Відсоток від загальної кількості**
Менше 1	60	60%
Від 1 до 3	30	30%
Від 3 до 5	8	8%
Більше 5	2	2%

Результати показали, що у більшості випадків (60%) система надає відповідь менше ніж за 1 секунду, що є відмінним показником. У 30% випадків час відповіді становив від 1 до 3 секунд, що є прийнятним для більшості користувачів. Проте, в деяких випадках (10%) час відповіді перевищував 3 секунди, а в окремих випадках досягав 5 та більше секунд. Це може бути пов'язано зі складністю запиту, великим обсягом інформації, яку необхідно опрацювати, або з тимчасовим навантаженням на сервери. Для покращення швидкодії необхідно провести оптимізацію алгоритмів пошуку та обробки інформації, а також перевірити апаратні характеристики серверів та каналів зв'язку.

Оцінка ефективності консультативної системи є критично важливим етапом, що дозволяє визначити її придатність для практичного використання та виявити потенційні напрямки для вдосконалення. Ключовими показниками ефективності виступають точність, повнота та швидкість відповідей системи, які характеризують здатність системи надавати достовірну, вичерпну та оперативну інформацію користувачам.

Точність відповідей визначається як відсоток коректних та обґрунтованих відповідей системи на поставлені запитання. Оцінка точності вимагає залучення експертів у відповідній предметній області, які оцінюють кожну відповідь системи на основі встановлених критеріїв. Висока точність відповідей є критично важливою для забезпечення довіри користувачів до системи та прийняття обґрунтованих рішень на основі наданої інформації. Фактори, що впливають на точність відповідей, включають якість та повноту знань, представлених в базі даних, ефективність алгоритмів обробки природної мови та машинного навчання, а також правильність налаштування параметрів системи.

Повнота відповідей характеризує здатність системи надавати вичерпну інформацію, що відповідає на всі аспекти поставленого запитання. Оцінка повноти передбачає визначення обсягу інформації, необхідного для надання вичерпної відповіді, та порівняння його з обсягом інформації, наданої системою. Повнота відповідей особливо важлива у складних ситуаціях, коли користувачу потрібна детальна інформація для прийняття зваженого рішення. Фактори, що впливають на повноту відповідей, включають структуру бази даних, методи вилучення знань та здатність системи інтегрувати інформацію з різних джерел.

Швидкість відповідей визначається як час, необхідний системі для надання відповіді на поставлене запитання. Оцінка швидкості передбачає вимірювання часу, що витрачається системою на обробку запиту, пошук інформації та формування відповіді. Швидкість відповідей є важливим фактором для забезпечення задоволеності користувачів та ефективного використання системи. Фактори, що впливають на швидкість відповідей, включають обсяг бази даних,

ефективність алгоритмів пошуку та індексування інформації, а також апаратні характеристики системи.

Перспективи впровадження консультативної системи в аграрному секторі є надзвичайно широкими та охоплюють різні сфери діяльності, від оптимізації технологій вирощування сільськогосподарських культур до підтримки прийняття рішень в управлінні аграрним бізнесом. Зокрема, система може бути використана для надання фермерам консультацій щодо вибору сортів рослин, внесення добрив, захисту від шкідників та хвороб, а також для прогнозування врожайності та аналізу ринкових тенденцій. Впровадження консультативної системи може сприяти підвищенню продуктивності аграрного виробництва, зниженню витрат, покращенню якості продукції та підвищенню конкурентоспроможності аграрних підприємств. Важливим аспектом успішного впровадження є адаптація системи до специфічних потреб та умов користувачів, а також забезпечення постійного оновлення та вдосконалення бази знань та алгоритмів обробки інформації. Крім того, необхідно враховувати соціальні та економічні фактори, такі як рівень комп'ютерної грамотності користувачів, доступність інтернету та вартість впровадження та підтримки системи. Незважаючи на певні виклики, впровадження консультативних систем в аграрному секторі має значний потенціал для сприяння сталого розвитку сільського господарства та забезпечення продовольчої безпеки.

Висновки до розділу 3

Ефективність автоматизованої консультативної системи на базі AgroUA.net вимагає ретельного вибору інструментарію та технологій, що ґрунтується на вимогах до системи, доступності, надійності та відповідності завданням. Платформа .NET і мова C# забезпечують продуктивність, строгий контроль типів та широкі можливості для розробки логіки та інтерфейсу, а Windows Forms пропонує зручний інструментарій для створення функціонального GUI. Для зберігання даних обґрунтовано вибір MySQL, відомої своєю надійністю та підтримкою з боку .NET. Для інтеграції NLP/ML

компонентів запропоновано використання бібліотек .NET або Python-сервісів через REST API, що дозволяє застосовувати найкращі інструменти для кожної задачі, хоча й додає складність в розгортанні. Трирівнева архітектура системи забезпечує модульність, масштабованість та легкість супроводу, з чітким розподілом функціональності між рівнем представлення, бізнес-логіки та доступу до даних. Проектування основних програмних модулів (інтерфейсу користувача, обробки запитів, взаємодії з моделями машинного навчання, доступу до бази знань, формування відповідей) та UML-діаграми демонструють взаємодію між частинами системи. Ключовим компонентом є модуль NLP/ML, який включає реалізацію компонентів C# для обробки тексту та інтеграцію Python-сервісів через API. Успішна реалізація системи вимагає не лише вибору оптимальних інструментів та архітектури, а й детального тестування та налагодження на основі реальних запитів користувачів.

Потенціал впровадження розробленої консультативної системи в аграрному секторі є значним, проте, реалізація цього потенціалу вимагає врахування певних факторів. Розроблена система має перспективи застосування для:

- **Підтримки фермерів та агрономів:** надання оперативної інформації щодо вибору сортів, технологій вирощування, захисту рослин, прогнозування врожайності.
- **Навчання та підвищення кваліфікації:** використання системи в освітніх закладах та на курсах підвищення кваліфікації для надання студентам та фахівцям доступу до актуальної інформації.
- **Прийняття рішень в агробізнесі:** надання аналітичної інформації та рекомендацій для оптимізації виробничих процесів та управління ресурсами.
- **Державної підтримки аграрного сектору:** використання системи для надання інформації щодо державних програм, пільг та субсидій.

Однак, для успішного впровадження необхідно врахувати наступні фактори:

- **Адаптація системи до потреб користувачів:** Проведення додаткових досліджень для виявлення специфічних потреб різних груп користувачів (великі агрохолдинги, малі фермерські господарства, консультанти) та адаптація інтерфейсу та функціональності системи до цих потреб.
- **Розширення бази знань:** Постійне оновлення та розширення бази знань з використанням нових наукових даних, практичного досвіду та інформації з інших джерел.
- **Забезпечення доступності системи:** Розробка мобільних додатків та веб-інтерфейсів для забезпечення доступу до системи з різних пристроїв та в різних місцях, включаючи сільську місцевість з обмеженим доступом до Інтернету.
- **Навчання користувачів:** Проведення навчальних курсів та тренінгів для ознайомлення користувачів з можливостями системи та навчання їх ефективному використанню.
- **Створення системи підтримки:** Забезпечення технічної підтримки користувачів, а також можливості надання зворотного зв'язку для покращення якості роботи системи.

Незважаючи на досягнуті результати, розроблена консультативна система має певні обмеження, які важливо враховувати:

1. Поточна реалізація настільного додатку з використанням MySQL та потенційної інтеграції з Python-сервісами через REST API може зіткнутися з проблемами продуктивності при значному зростанні кількості одночасних користувачів або обсягу даних у базі знань. Для промислового впровадження може знадобитися перехід на більш масштабовані архітектури (наприклад, мікросервісну) та бази даних (наприклад, Elasticsearch для повнотекстового пошуку, спеціалізовані векторні бази даних для семантичного пошуку).

2. Якщо наповнення бази знань здійснюється шляхом веб-скрейпінгу, будь-які зміни в HTML-структурі порталу AgroUA.net можуть призвести до збоїв у роботі скрейперів та вимагатимуть їх оновлення, тому обраний для взаємодії сервіс приватний сервіс - агрегатор сільськогосподарських даних

3. Хоча система використовує NLP для аналізу запитів та контенту, повне та точне вилучення знань з великих обсягів неструктурованих текстів (статей, новин) залишається складним завданням. Якість бази знань безпосередньо впливає на якість консультацій.

4. Аграрний сектор постійно розвивається: з'являються нові сорти, технології, препарати, змінюються ринкові умови. Система потребує механізмів регулярного та ефективного оновлення бази знань.

5. Попри донавчання моделей, обробка синонімії, полісемії, нових термінів та абревіатур в українській аграрній лексиці може викликати труднощі.

6. Поточні моделі мають обмеження у розумінні складних, багатоетапних запитів або неявного контексту діалогу ,проявляють галюцинації та міражі у відповідях

Важливим напрямком для подальшого розвитку є проведення всебічного тестування системи з реальними кінцевими користувачами (фермерами, агрономами) для збору відгуків щодо зручності використання, повноти та корисності наданої інформації. Це дозволить точніше адаптувати систему до їхніх потреб.

ВИСНОВКИ

Розробка консультативних систем на основі штучного інтелекту представляє собою перспективний напрямок, що сприяє автоматизації процесів підтримки прийняття рішень у різних сферах. Ефективність різних підходів, зокрема систем на правилах, на знаннях, та машинного навчання, залежить від специфіки предметної області, що вимагає ретельного вибору методів та їхньої адаптації.

Аналіз аграрної термінології з використанням обробки природної мови (NLP) та машинного навчання відкриває нові можливості для автоматизації та оптимізації процесів управління знаннями в аграрному секторі. Зважаючи на специфіку аграрної мови, необхідна розробка спеціалізованих методів, що враховують наявність термінів, абревіатур та багатозначність.

Створення баз знань для аграрних консультативних систем вимагає особливого підходу до представлення даних, що враховує їх великий обсяг, різноманітність, неповноту та часову змінність. Інтеграція експертних знань, структурованих та неструктурованих даних є ключовою для забезпечення повноти та актуальності інформації.

Розроблено архітектуру NLP-модуля для аграрної консультативної системи, яка включає компоненти для розпізнавання інтенту та виокремлення сутностей. Забезпечено інтеграцію з навченими моделями машинного навчання та запропоновано використання Python-сервісів через REST API для обробки запитів.

Спроектовано алгоритми машинного навчання для класифікації запитів та пошуку релевантної інформації, що базуються на семантичному пошуку, векторних представленнях тексту та екстрактивних QA-моделях. Запропоновано підходи до оптимізації гіперпараметрів моделей для досягнення максимальної ефективності.

Розроблено структуру бази знань, інтегрованої з даними фахового порталу, що передбачає автоматичне вилучення інформації, структурування знань у

вигляді пар "питання-відповідь" або графів, індексацію для швидкого пошуку та вибір формату зберігання даних, що відповідає обраній моделі представлення.

Наведено опис тестування та налагодження системи на основі реальних запитів користувачів, що включає оцінку якості та стабільності роботи системи, виявлення та усунення помилок, а також аналіз проблемних випадків та їх вирішення.

У ході експериментального тестування розроблена система продемонструвала наступні показники ефективності: точність класифікації інтентів склала 92%, F1-міра для розпізнавання аграрних сутностей – 87%, а середня точність інформаційного пошуку (MAP) – 0.83. Експертна оцінка підтвердила, що у 85% випадків система надавала точні та повні відповіді (оцінка 3-4 бали). Середній час відповіді системи не перевищував 1.2 секунди.

Наукова новизна отриманих результатів полягає у наступному: Запропоновано архітектуру автоматизованої консультативної системи для аграрного сектору, яка поєднує методи обробки природної української мови (NLP) на основі трансформерних моделей, інтегровану базу знань з контентом спеціалізованого агрегаційного порталу та алгоритми семантичного пошуку, що відрізняється від існуючих аналогів Cropio, FieldView більшою сфокусованістю на українськомовному контенті та специфічній аграрній термінології, а також наданням консультацій природною мовою, а не лише аналітичних звітів.

Розроблено підхід до адаптації та донавчання NLP-моделей (класифікації інтентів та розпізнавання іменованих сутностей) на корпусі українських аграрних текстів, що дозволило підвищити точність обробки специфічних термінів та запитів порівняно з використанням загальномовних моделей без адаптації.

Спроектовано структуру бази знань, що інтегрує дані з профільних порталів та локальні знання користувача, та розроблено механізми для їх вилучення, структурування та індексації для забезпечення ефективного пошуку. Практичне значення отриманих результатів полягає у створенні програмного прототипу консультативної системи, яка може бути використана для надання

українським аграріям оперативних та обґрунтованих рекомендацій. Система демонструє найкраще розуміння запитів українською, пошук по базі профільних видань сприяє підвищенню ефективності аграрного виробництва шляхом полегшення доступу до актуальної інформації. Результати роботи можуть бути використані при розробці комерційних або освітніх аграрних інформаційних систем.

Впровадження розробленої автоматизованої консультативної системи в аграрному секторі має потенціал для досягнення значних економічних вигод для сільгоспвиробників. Хоча точний кількісний розрахунок ефекту вимагає тривалого пілотного впровадження та збору статистичних даних, можна виділити ключові напрямки позитивного впливу:

Завдяки оперативному доступу до інформації про ефективність різних засобів захисту рослин (ЗЗР) та добрив, а також рекомендацій щодо їх раціонального застосування, фермери можуть уникнути зайвих закупівель або використання неоптимальних препаратів. Це може призвести до зниження витрат на ЗЗР та добрива на 5-15%.

Допомога у виборі найбільш адаптованих та врожайних сортів для конкретних кліматичних та ґрунтових умов дозволяє ефективніше використовувати насіннєвий матеріал.

Швидке отримання інформації про симптоми хвороб та шкідників, а також рекомендацій щодо методів боротьби з ними, дозволяє своєчасно реагувати на загрози. Це може суттєво зменшити втрати врожаю, які, за різними оцінками, можуть сягати 20-30% через несвоєчасні або неправильні дії.

Система дозволяє значно скоротити час, який агрономи та фермери витрачають на пошук необхідної інформації в різних джерелах. Заощаджений час може бути спрямований на безпосередню роботу в полі або аналітичну діяльність. Надання обґрунтованих рекомендацій підвищує якість прийнятих рішень, що мінімізує ризики, пов'язані з неправильним вибором технологій вирощування.

Хоча система безпосередньо не впливає на біологічні процеси, надання правильної інформації в потрібний час (наприклад, про оптимальні строки сівби, методи обробки ґрунту) створює передумови для кращої реалізації потенціалу культур та, як наслідок, може сприяти підвищенню врожайності.

Наукова новизна отриманих результатів полягає у:

Запропонованій архітектурі спеціалізованої аграрної консультативної системи, що враховує унікальні особливості українськомовного контенту та аграрної термінології, та інтегрує дані з національного профільного порталу

Обґрунтуванні підходу до адаптації та донавчання сучасних NLP-моделей для ефективної обробки української аграрної лексики та запитів.

Розробці комбінованого алгоритму пошуку інформації, що використовує як лексичну відповідність, так і семантичну близькість.

Незважаючи на досягнуті результати, система має певні обмеження, зокрема масштабованість, залежність від агрегаторів та профільних порталів, динаміка оновлення знань. Подальший розвиток системи може включати: розробку веб-інтерфейсу, інтеграцію з додатковими джерелами даних, впровадження більш досконалих моделей генерації відповідей, розширення функціоналу на основі зворотного зв'язку від користувачів та проведення широкомасштабного пілотного впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Балацька Н. Ю. Аграрний бізнес в умовах пандемії коронавірусу: проблеми та напрями трансформації моделей розвитку // *Інфраструктура ринку.* – 2020. – Вип. 42. – С. 117–122.

2. Білоус О. С. Автоматизовані експертні системи для підтримки прийняття рішень в агропромисловому секторі // *Вісник економічної кібернетики.* – 2023. – № 2. – С. 145–153.

3. Бондаренко О. М. Використання обробки природної мови для аналізу аграрних текстів // *Комп'ютерна лінгвістика та інтелектуальні системи.* – 2021. – Т. 3, № 1. – С. 75–82.

4. Гончаренко Л. В. Технології доброчесного використання штучного інтелекту у сфері освіти та науки // Матеріали всеукраїнського науково-педагогічного підвищення кваліфікації. – Одеса: Видавничий дім «Гельветика», 2023. – 276 с.

5. Гончарук Л. В. Використання чат-ботів у консалтингу аграрних підприємств // *Комп’ютерні науки та інформаційні технології.* – 2023. – № 4(32). – С. 55–63.

6. Гордієнко Л. П. Інноваційні технології обробки аграрних даних на базі штучного інтелекту // *Комп’ютерні науки та моделювання складних систем.* – 2024. – № 6(14). – С. 88–97.

7. Гросул В., Балацька Н. Digital-маркетинг як дієвий інструмент антикризового розвитку підприємства ресторанного бізнесу в період пандемії та її рецесії // *Підприємництво та інновації.* – 2020. – № 11/2. – С. 7–12.

8. Давидова О. Ю. Формування системи інноваційного управління розвитком підприємств готельно-ресторанного господарства // *Бізнес-Інформ.* – 2017. – № 11. – С. 459–464.

9. Давидова О. Ю., Полстяна Н. В. Інформаційно-комп’ютерні інновації в ресторанному бізнесі // *Комунальне господарство міст, Сер. Економічні науки.* – 2012. – № 106. – С. 403–408.

10. Данкевич В., Данкевич А. Інтернет речей та штучний інтелект як ключові технології в аграрному секторі // *Економічні науки.* – 2024. – № 7(24). – С. 165–173.

11. Демченко В. Г. Застосування методів машинного навчання для класифікації аграрної термінології // *Штучний інтелект і технології розумних систем.* – 2022. – № 5(11). – С. 101–109.

12. Дєдоша О. Гнучке управління створенням мобільного застосунку для онлайн–запису клієнтів салону краси «Стиль» : пояснювальна записка до кваліфікаційної роботи магістра за спеціальністю 073 «Менеджмент» ОП «Agile-технології розробки програмного забезпечення» / наук. кер.: О. Мушинський, В. Ткаченко ; Університет економіки та права «КРОК». – Київ, 2024. – 74 с.

13. Домарацька О.Є. Роль штучного інтелекту в розвитку аграрного сектора економіки // Матеріали науково-практичної конференції «Сучасні тенденції розвитку економіки». – Миколаїв: МНАУ, 2023. – С. 48–51.

14. Дорошенко П. В. Інформаційні системи в аграрному менеджменті: сучасні підходи та перспективи розвитку // *Аграрна економіка.* – 2023. – Т. 6, № 2. – С. 210–217.

15. Іваненко О. С. Моделювання аграрних процесів з використанням методів машинного навчання // *Комп'ютерні науки та інформаційні технології.* – 2023. – Т. 4, № 1. – С. 45–52.

16. Коваленко І. М. Аналіз ефективності впровадження штучного інтелекту у сільському господарстві України // *Сучасні інформаційні технології в економіці та управлінні.* – 2025. – № 1(14). – С. 75–83.

17. Коваль Л. М., Заячковська Г. А. Маркетингові інновації закладів ресторанного бізнесу // *Проблеми системного підходу в економіці.* – 2020. – Т. 3, № 2. – С. 128–133.

18. Ковальчук М. Г. Використання штучного інтелекту та інтернету речей у післявоєнному розвитку України // *Публічне управління та адміністрування.* – 2024. – № 5(30). – С. 120–128.

19. Косенко Т. Л. Інформаційно-аналітичні системи для моніторингу аграрних процесів // *Системи управління в агробізнесі.* – 2023. – № 5(17). – С. 49–56.

20. Кравченко Ю. М. Аналіз аграрної термінології засобами обробки природної мови // *Філологічні студії.* – 2020. – № 8(15). – С. 130–136.

21. Лисенко А. О. Застосування нейронних мереж для прогнозування врожайності сільськогосподарських культур // *Вісник аграрної науки.* – 2024. – № 11. – С. 90–95.

22. Литвиненко Д. С. Використання штучного інтелекту для оптимізації аграрного виробництва // *Комп'ютерне моделювання в економіці та управлінні.* – 2025. – № 1(10). – С. 33–41.

23. Мартинюк І. Г. Використання великих даних для оптимізації аграрних процесів // *Системний аналіз та інформаційні технології.* – 2025. – № 1(12). – С. 55–61.

24. Мельник О. П. Штучний інтелект у вищій освіті: ризики та перспективи інтеграції // Матеріали всеукраїнського науково-педагогічного підвищення кваліфікації. – Львів–Торунь: Liha-Pres, 2024. – 328 с.

25. Мельниченко К. Ю. Аналіз застосування обробки природної мови для автоматизації аграрного документообігу // *Наукові праці з інформаційних технологій.* – 2023. – № 4(15). – С. 147–155.

26. Назаренко В. О. Застосування технологій машинного навчання для автоматизації аграрного консультування // *Аграрні науки та технології.* – 2024. – № 3(18). – С. 88–95.

27. Обробка природної мови: навчальний посібник / за ред. О.О. Петрова. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 320 с.

28. Олійник С. П. Розробка чат-ботів для аграрних консультативних систем // *Програмні продукти і системи.* – 2022. – № 4(34). – С. 25–30.

29. Остапенко Н. І. Автоматизація обробки природної мови у сфері агроконсалтингу // *Інформаційні технології в науці та освіті.* – 2021. – № 2(20). – С. 65–72.

30. Петренко Н. В. Інтелектуальні системи підтримки прийняття рішень в аграрному секторі // *Інформаційні технології та комп'ютерна інженерія.* – 2021. – № 3(50). – С. 78–84.

31. Петрук О. В. Використання машинного навчання у сільському господарстві: перспективи та виклики // *Сучасні інформаційні технології та інноваційний розвиток.* – 2022. – № 4(8). – С. 110–118.

32. Посібник для менеджера проекту з гнучких методологій // techukraine.net. – [Електронний ресурс]. – Режим доступу: https://techukraine.net/посібник-для-менеджера

33. Рибак О. Г. Використання предиктивної аналітики в агропромисловому секторі // *Агроекономіка та інновації.* – 2022. – № 3(9). – С. 58–66.

34. Романенко П. Л. Аналіз методів обробки природної мови в аграрних інформаційних системах // *Вісник Харківського національного університету радіоелектроніки.* – 2021. – № 2(98). – С. 185–190.

35. Савченко Л. В. Автоматизовані системи моніторингу стану ґрунтів на основі штучного інтелекту // *Екологічна безпека та природокористування.* – 2023. – № 5(23). – С. 140–146.

36. Савчук І. П. Роль штучного інтелекту у вирішенні проблем продовольчої безпеки // *Економіка, менеджмент та інформаційні технології.* – 2024. – № 1(25). – С. 92–101.

37. Семенов В. І. Використання штучного інтелекту в аграрному секторі: міжнародний досвід та перспективи для України // *Економіка АПК.* – 2022. – № 9. – С. 112–118.

38. Сидоренко І. В. Використання технологій штучного інтелекту в аграрному секторі України // Матеріали міжнародної науково-практичної конференції «Інновації в аграрній науці». – Вінниця: ВНТУ, 2021. – С. 115–118.

39. Сидорчук О. П. Використання штучних нейронних мереж для прогнозування врожайності // *Аграрна інформатика.* – 2024. – № 3(7). – С. 119–126.

40. Тищенко С. В. Цифрові технології в індустрії гостинності // *Таврійський науковий вісник, Сер. Економіка.* – 2021. – Т. 7. – С. 131–139.

41. Ткачук В. О., Шестакова А. В. Використання мобільних додатків для цифровізації бізнес-процесів у ресторанному бізнесі та їх оптимізація на основі методу A/B тестування // *Економіка, управління та адміністрування.* – 2022. – № 4(102). – С. 28–34.

42. Топчій О. В. Класифікація текстів повідомлень користувачів з використанням методів машинного навчання : магістерська дисертація / Київський політехнічний інститут ім. Ігоря Сікорського. – Київ, 2020. – 71 с.

43. Федорченко О. В. Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень : робота на здобуття кваліфікаційного ступеня бакалавра за спеціальністю 122 – комп'ютерні науки / наук. кер. О. В. Бойко ; Сумський державний університет. – Суми, 2024. – 100 с.

44. Федосова К. С. Використання інформаційно-комунікаційних технологій в ресторанному бізнесі в період пандемії Covid-19 // Економіка та суспільство. – 2022. – [Електронний ресурс]. – Режим доступу: https://economyandsociety.in.ua/index.php/journal/article/view/1085/1042.

45. Фролов О. М. Використання когнітивних технологій для автоматизації прийняття рішень в аграрній сфері // Штучний інтелект і цифрові технології. – 2021. – № 2(13). – С. 77–84.

46. Чалий Н. С. Мобільний додаток підтримки діяльності кав'ярні : робота на здобуття кваліфікаційного ступеня бакалавра за спеціальністю 122 – комп'ютерні науки / наук. кер. В. Антипенко ; Сумський державний університет. – Суми, 2024. – 78 с.

47. Чміль Г. Л., Джгуташвілі Н. М. Цифровізація управління клієнтським досвідом у готельно-ресторанній індустрії // *Бізнес Інформ.* – 2020. – № 8. – С. 237–245.

48. Шевченко А. А., Петренко О. П., Косик Д. В. Штучний інтелект в рослинництві: перспективи та виклики // *Модерна економіка.* – 2025. – № 3(15). – С. 45–53.

49. Щербак В. І. Моделювання та прогнозування аграрних процесів на основі штучного інтелекту // *Аграрні інформаційні системи.* – 2025. – № 3(12). – С. 67–75.

ДОДАТКИ

Додаток А

```
namespace AgroConsultAI
{
    public partial class LoginForm : Form
    {
        private System.Windows.Forms.Label usernameLabel;
        private System.Windows.Forms.TextBox usernameTextBox;
        private System.Windows.Forms.Label passwordLabel;
        private System.Windows.Forms.TextBox passwordTextBox;
        private System.Windows.Forms.Button loginButton;
        private System.Windows.Forms.LinkLabel registerLink;
        private System.Windows.Forms.LinkLabel forgotPasswordLink;

        public LoginForm()
        {
            InitializeComponent();
            CreateComponent();
        }

        private void CreateComponent()
        {
            this.usernameLabel = new System.Windows.Forms.Label();
            this.usernameTextBox = new System.Windows.Forms.TextBox();
            this.passwordLabel = new System.Windows.Forms.Label();
            this.passwordTextBox = new System.Windows.Forms.TextBox();
            this.loginButton = new System.Windows.Forms.Button();
            this.registerLink = new System.Windows.Forms.LinkLabel();
            this.forgotPasswordLink = new System.Windows.Forms.LinkLabel();
            this.SuspendLayout();
```

```
//
// usernameLabel
//
this.usernameLabel.AutoSize = true;
this.usernameLabel.Location = new System.Drawing.Point(30, 30);
this.usernameLabel.Name = "usernameLabel";
this.usernameLabel.Size = new System.Drawing.Size(103, 17);
this.usernameLabel.TabIndex = 0;
this.usernameLabel.Text = "Ім'я користувача:";
//
// usernameTextBox
//
this.usernameTextBox.Location = new System.Drawing.Point(150, 27);
this.usernameTextBox.Name = "usernameTextBox";
this.usernameTextBox.Size = new System.Drawing.Size(200, 22);
this.usernameTextBox.TabIndex = 1;
//
// passwordLabel
//
this.passwordLabel.AutoSize = true;
this.passwordLabel.Location = new System.Drawing.Point(30, 70);
this.passwordLabel.Name = "passwordLabel";
this.passwordLabel.Size = new System.Drawing.Size(61, 17);
this.passwordLabel.TabIndex = 2;
this.passwordLabel.Text = "Пароль:";
//
// passwordTextBox
//
this.passwordTextBox.Location = new System.Drawing.Point(150, 67);
this.passwordTextBox.Name = "passwordTextBox";
```

```
        this.passwordTextBox.PasswordChar = '*';
        this.passwordTextBox.Size = new System.Drawing.Size(200, 22);
        this.passwordTextBox.TabIndex = 2; // Змінено TabIndex
                                //
                                // loginButton
                                //
        this.loginButton.Location = new System.Drawing.Point(150, 110);
        this.loginButton.Name = "loginButton";
        this.loginButton.Size = new System.Drawing.Size(100, 30);
        this.loginButton.TabIndex = 3; // Змінено TabIndex
        this.loginButton.Text = "Увійти";
        this.loginButton.UseVisualStyleBackColor = true;
        this.loginButton.Click += new System.EventHandler(this.LoginButton_Click);
        //
        // registerLink
        //
        this.registerLink.AutoSize = true;
        this.registerLink.Location = new System.Drawing.Point(30, 150);
        this.registerLink.Name = "registerLink";
        this.registerLink.Size = new System.Drawing.Size(100, 17);
        this.registerLink.TabIndex = 4; // Змінено TabIndex
        this.registerLink.TabStop = true;
        this.registerLink.Text = "Зареєструватися";
        this.registerLink.LinkClicked                    +=                    new
System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.RegisterLink_Lin
kClicked);
        //
        // forgotPasswordLink
        //
        this.forgotPasswordLink.AutoSize = true;
```

```
            this.forgotPasswordLink.Location = new System.Drawing.Point(240, 150); // Змінено розташування
            this.forgotPasswordLink.Name = "forgotPasswordLink";
            this.forgotPasswordLink.Size = new System.Drawing.Size(110, 17);
            this.forgotPasswordLink.TabIndex = 5; // Змінено TabIndex
            this.forgotPasswordLink.TabStop = true;
            this.forgotPasswordLink.Text = "Забули пароль?";
            this.forgotPasswordLink.LinkClicked += new System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.ForgotPasswordLink_LinkClicked);
            //
            // LoginForm
            //
            this.AcceptButton = this.loginButton; // Дозволяє Enter для кнопки Увійти
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(382, 193);
            this.Controls.Add(this.forgotPasswordLink);
            this.Controls.Add(this.registerLink);
            this.Controls.Add(this.loginButton);
            this.Controls.Add(this.passwordTextBox);
            this.Controls.Add(this.passwordLabel);
            this.Controls.Add(this.usernameTextBox);
            this.Controls.Add(this.usernameLabel);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "LoginForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
```

```
        this.Text = "Вхід до системи";
        this.ResumeLayout(false);
        this.PerformLayout();
    }

    private void LoginButton_Click(object sender, EventArgs e)
    {
        string username = usernameTextBox.Text;
        string password = passwordTextBox.Text;

        if (username == "admin" && password == "password")
        {
            MessageBox.Show("Вхід успішний!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.DialogResult = DialogResult.OK; // Важливо для закриття форми та продовження
            this.Close();
        }
        else
        {
            MessageBox.Show("Неправильне ім'я користувача або пароль.", "Помилка входу", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void RegisterLink_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        RegistrationForm registrationForm = new RegistrationForm();
        registrationForm.ShowDialog(); // Показати як модальне вікно
```

```
        }

        private void ForgotPasswordLink_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            MessageBox.Show("Функція відновлення пароля ще не реалізована.",
"Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

Додаток Б

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AgroConsultAI
{
    public partial class LocalKnowledgeBaseForm : Form
    {
        private System.Windows.Forms.SplitContainer splitContainerMain;
        private System.Windows.Forms.TreeView udcTreeView;
        private System.Windows.Forms.Label udcLabel;
        private System.Windows.Forms.Panel dropPanel;
```

```
private System.Windows.Forms.Label dropLabel;
private System.Windows.Forms.Panel queryPanel;
private System.Windows.Forms.Label loadedFileLabel;
private System.Windows.Forms.Label queryLocalKbPromptLabel;
private System.Windows.Forms.TextBox queryLocalKbTextBox;
private System.Windows.Forms.Button askLocalKbButton;
private System.Windows.Forms.Label answerLocalKbPromptLabel;
private System.Windows.Forms.TextBox answerLocalKbTextBox;
private System.Windows.Forms.Button unloadKbButton;

private string loadedFilePath = null;
private string loadedFileContent = null;

public LocalKnowledgeBaseForm()
{
    InitializeComponent();
    CreateComponent();
    PopulateUdcTree(); // Заповнюємо дерево УДК
    SetupInitialState(); // Налаштовуємо початковий стан правої панелі
}

private void CreateComponent()
{
    this.splitContainerMain = new System.Windows.Forms.SplitContainer();
    this.udcTreeView = new System.Windows.Forms.TreeView();
    this.udcLabel = new System.Windows.Forms.Label();

    // Ініціалізація dropPanel
    this.dropPanel = new System.Windows.Forms.Panel();
    this.dropLabel = new System.Windows.Forms.Label();
```

```
        // Ініціалізація queryPanel та її контролів
        this.queryPanel = new System.Windows.Forms.Panel();
        this.loadedFileLabel = new System.Windows.Forms.Label();
        this.queryLocalKbPromptLabel = new System.Windows.Forms.Label();
        this.queryLocalKbTextBox = new System.Windows.Forms.TextBox();
        this.askLocalKbButton = new System.Windows.Forms.Button();
        this.answerLocalKbPromptLabel = new System.Windows.Forms.Label();
        this.answerLocalKbTextBox = new System.Windows.Forms.TextBox();
        this.unloadKbButton = new System.Windows.Forms.Button();


((System.ComponentModel.ISupportInitialize)(this.splitContainerMain)).BeginInit();
        this.splitContainerMain.Panel1.SuspendLayout();
        this.splitContainerMain.Panel2.SuspendLayout();
        this.splitContainerMain.SuspendLayout();

        this.dropPanel.SuspendLayout();
        this.queryPanel.SuspendLayout();
        this.SuspendLayout();

        //
        // splitContainerMain
        //
        this.splitContainerMain.Dock = System.Windows.Forms.DockStyle.Fill;
        this.splitContainerMain.Location = new System.Drawing.Point(0, 0);
        this.splitContainerMain.Name = "splitContainerMain";
        //
        // splitContainerMain.Panel1 (ліва панель - УДК)
        //
```

```
        this.splitContainerMain.Panel1.Controls.Add(this.udcTreeView);
        this.splitContainerMain.Panel1.Controls.Add(this.udcLabel);
        this.splitContainerMain.Panel1MinSize = 150; // Мінімальна ширина лівої панелі
                                                     //
                                                     // splitContainerMain.Panel2 (права панель - динамічна)
                                                     //
        this.splitContainerMain.Panel2.Controls.Add(this.queryPanel); // queryPanel додаємо першим, щоб бути "під" dropPanel
        this.splitContainerMain.Panel2.Controls.Add(this.dropPanel);
        this.splitContainerMain.Panel2MinSize = 300; // Мінімальна ширина правої панелі
        this.splitContainerMain.Size = new System.Drawing.Size(800, 500);
        this.splitContainerMain.SplitterDistance = (int)(this.splitContainerMain.Width * 0.30); // 30%
        this.splitContainerMain.TabIndex = 0;
        this.splitContainerMain.FixedPanel = FixedPanel.Panel1; // Фіксуємо ширину першої панелі при зміні розміру форми

        //
        // udcLabel
        //
        this.udcLabel.AutoSize = true;
        this.udcLabel.Location = new System.Drawing.Point(10, 10);
        this.udcLabel.Name = "udcLabel";
        this.udcLabel.Size = new System.Drawing.Size(120, 17);
        this.udcLabel.Text = "Класифікація УДК:";
        //
        // udcTreeView
```

```
            //
            this.udcTreeView.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.T
op | System.Windows.Forms.AnchorStyles.Bottom)
                | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
            this.udcTreeView.Location = new System.Drawing.Point(10, 30);
            this.udcTreeView.Name = "udcTreeView";
            this.udcTreeView.Size = new
System.Drawing.Size(this.splitContainerMain.Panel1.Width - 20,
this.splitContainerMain.Panel1.Height - 40);
            this.udcTreeView.TabIndex = 1;
            //
            // dropPanel
            //
            this.dropPanel.AllowDrop = true;
            this.dropPanel.BackColor = System.Drawing.SystemColors.ControlLight;
            this.dropPanel.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.dropPanel.Controls.Add(this.dropLabel);
            this.dropPanel.Dock = System.Windows.Forms.DockStyle.Fill;
            this.dropPanel.Location = new System.Drawing.Point(0, 0);
            this.dropPanel.Name = "dropPanel";
            this.dropPanel.Size = this.splitContainerMain.Panel2.ClientSize; // Розмір як
у батьківської панелі
            this.dropPanel.TabIndex = 0;
            this.dropPanel.DragEnter += new
System.Windows.Forms.DragEventHandler(this.DropPanel_DragEnter);
            this.dropPanel.DragDrop += new
System.Windows.Forms.DragEventHandler(this.DropPanel_DragDrop);
```

```
            //
            // dropLabel
            //
            this.dropLabel.Dock = DockStyle.Fill; // Заповнює всю dropPanel
            this.dropLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
            this.dropLabel.Text = "Перетягніть сюди файл бази знань\n(.txt, .pdf, .docx
тощо)";
            this.dropLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.dropLabel.ForeColor = System.Drawing.SystemColors.ControlDarkDark;
            //
            // queryPanel (панель для запитів)
            //
            this.queryPanel.Dock = System.Windows.Forms.DockStyle.Fill;
            this.queryPanel.Location = new System.Drawing.Point(0, 0);
            this.queryPanel.Name = "queryPanel";
            this.queryPanel.Padding = new System.Windows.Forms.Padding(10);
            this.queryPanel.Size = this.splitContainerMain.Panel2.ClientSize;
            this.queryPanel.TabIndex = 1;
            this.queryPanel.Visible = false; // Початково прихована
                                    //
                                    // loadedFileLabel
                                    //
            this.loadedFileLabel.AutoSize = true;
            this.loadedFileLabel.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.loadedFileLabel.Location = new System.Drawing.Point(10, 10);
```

```
        this.loadedFileLabel.Name = "loadedFileLabel";
        this.loadedFileLabel.Text = "Завантажено: ";
        //
        // unloadKbButton
        //
        this.unloadKbButton.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.To
p | System.Windows.Forms.AnchorStyles.Right)));
        this.unloadKbButton.Location = new
System.Drawing.Point(this.queryPanel.Width - 130, 5);
        this.unloadKbButton.Name = "unloadKbButton";
        this.unloadKbButton.Size = new System.Drawing.Size(120, 28);
        this.unloadKbButton.Text = "Вивантажити";
        this.unloadKbButton.UseVisualStyleBackColor = true;
        this.unloadKbButton.Click += new
System.EventHandler(this.UnloadKbButton_Click);
        //
        // queryLocalKbPromptLabel
        //
        this.queryLocalKbPromptLabel.AutoSize = true;
        this.queryLocalKbPromptLabel.Location = new System.Drawing.Point(10,
40);
        this.queryLocalKbPromptLabel.Name = "queryLocalKbPromptLabel";
        this.queryLocalKbPromptLabel.Text = "Ваше запитання до завантаженої
бази:";
        //
        // queryLocalKbTextBox
        //
```

```
            this.queryLocalKbTextBox.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.To
p | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
            this.queryLocalKbTextBox.Location = new System.Drawing.Point(10, 60);
            this.queryLocalKbTextBox.Multiline = true;
            this.queryLocalKbTextBox.Name = "queryLocalKbTextBox";
            this.queryLocalKbTextBox.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.queryLocalKbTextBox.Size = new
System.Drawing.Size(this.queryPanel.Width - 20, 80);
            this.queryLocalKbTextBox.TabIndex = 0;
            //
            // askLocalKbButton
            //
            this.askLocalKbButton.Location = new System.Drawing.Point(10, 145);
            this.askLocalKbButton.Name = "askLocalKbButton";
            this.askLocalKbButton.Size = new System.Drawing.Size(150, 30);
            this.askLocalKbButton.Text = "Запитати по базі";
            this.askLocalKbButton.UseVisualStyleBackColor = true;
            this.askLocalKbButton.Click += new
System.EventHandler(this.AskLocalKbButton_Click);
            this.askLocalKbButton.TabIndex = 1;
            //
            // answerLocalKbPromptLabel
            //
            this.answerLocalKbPromptLabel.AutoSize = true;
            this.answerLocalKbPromptLabel.Location = new System.Drawing.Point(10,
185);
            this.answerLocalKbPromptLabel.Name = "answerLocalKbPromptLabel";
```

```
            this.answerLocalKbPromptLabel.Text = "Відповідь:";
            //
            // answerLocalKbTextBox
            //
            this.answerLocalKbTextBox.Anchor                                    =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.T
op | System.Windows.Forms.AnchorStyles.Bottom)
                | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
            this.answerLocalKbTextBox.Location = new System.Drawing.Point(10, 205);
            this.answerLocalKbTextBox.Multiline = true;
            this.answerLocalKbTextBox.Name = "answerLocalKbTextBox";
            this.answerLocalKbTextBox.ReadOnly = true;
            this.answerLocalKbTextBox.ScrollBars                                =
System.Windows.Forms.ScrollBars.Vertical;
            this.answerLocalKbTextBox.Size                  =                   new
System.Drawing.Size(this.queryPanel.Width - 20, this.queryPanel.Height - 215);
            this.answerLocalKbTextBox.TabIndex = 2;

            // Додавання контролів до queryPanel
            this.queryPanel.Controls.Add(this.loadedFileLabel);
            this.queryPanel.Controls.Add(this.unloadKbButton);
            this.queryPanel.Controls.Add(this.queryLocalKbPromptLabel);
            this.queryPanel.Controls.Add(this.queryLocalKbTextBox);
            this.queryPanel.Controls.Add(this.askLocalKbButton);
            this.queryPanel.Controls.Add(this.answerLocalKbPromptLabel);
            this.queryPanel.Controls.Add(this.answerLocalKbTextBox);

            //
            // LocalKnowledgeBaseForm
```

```
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 500);
            this.Controls.Add(this.splitContainerMain);
            this.Name = "LocalKnowledgeBaseForm";
            this.Text = "Локальна База Знань";
            this.Load += new System.EventHandler(this.LocalKnowledgeBaseForm_Load); // для встановлення SplitterDistance

            this.splitContainerMain.Panel1.ResumeLayout(false);
            this.splitContainerMain.Panel1.PerformLayout();
            this.splitContainerMain.Panel2.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.splitContainerMain)).EndInit();
            this.splitContainerMain.ResumeLayout(false);

            this.dropPanel.ResumeLayout(false);
            this.queryPanel.ResumeLayout(false);
            this.queryPanel.PerformLayout(); // Важливо для міток всередині queryPanel

            this.ResumeLayout(false);
        }
        private void LocalKnowledgeBaseForm_Load(object sender, EventArgs e)
        {
            this.splitContainerMain.SplitterDistance = (int)(this.Width * 0.30);
        }
```

```
private void SetupInitialState()
{
    dropPanel.Visible = true;
    queryPanel.Visible = false;
    loadedFilePath = null;
    loadedFileContent = null;
    queryLocalKbTextBox.Clear();
    answerLocalKbTextBox.Clear();
}

private void PopulateUdcTree()
{
    udcTreeView.Nodes.Clear();
    TreeNode rootNode = new TreeNode("0 Загальний відділ");
    TreeNode node00 = new TreeNode("00 Загальні питання науки та культури");
    TreeNode node004 = new TreeNode("004 Комп'ютерні науки та технології");
    node004.Nodes.Add(new TreeNode("004.8 Штучний інтелект"));
    node00.Nodes.Add(node004);
    rootNode.Nodes.Add(node00);
    udcTreeView.Nodes.Add(rootNode);

    TreeNode rootNode6 = new TreeNode("6 Прикладні науки. Медицина. Техніка");
    TreeNode node63 = new TreeNode("63 Сільське господарство...");
    node63.Nodes.Add(new TreeNode("631 Загальні питання с.г."));
    node63.Nodes.Add(new TreeNode("633 Польові культури"));
    rootNode6.Nodes.Add(node63);
    udcTreeView.Nodes.Add(rootNode6);
```

```
    udcTreeView.ExpandAll(); // Розгортаємо всі вузли для наочності
}

private void DropPanel_DragEnter(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.FileDrop))
    {
        e.Effect = DragDropEffects.Copy; // Або Link, якщо не копіюємо файл
    }
    else
    {
        e.Effect = DragDropEffects.None;
    }
}

private void DropPanel_DragDrop(object sender, DragEventArgs e)
{
    string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
    if (files != null && files.Length > 0)
    {
        string filePath = files[0];
                                (.txt, .pdf, .docx)
        string extension = Path.GetExtension(filePath).ToLower();
        if (extension == ".txt") // Для прикладу обробляємо тільки .txt
        {
            try
            {
                loadedFileContent = File.ReadAllText(filePath);
                loadedFilePath = filePath;
```

```
                loadedFileLabel.Text = $"Завантажено: {Path.GetFileName(filePath)}";
                dropPanel.Visible = false;
                queryPanel.Visible = true;
                answerLocalKbTextBox.Text = $"Файл '{Path.GetFileName(filePath)}' завантажено. \nВміст (перші 200 символів):\n{loadedFileContent.Substring(0, Math.Min(loadedFileContent.Length, 200))}...";

            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка читання файлу: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                SetupInitialState();
            }
        }
        else if (extension == ".pdf" || extension == ".docx")
        {
      string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
  if (files != null && files.Length > 0)
  {
      string filePath = files[0];
      string extension = Path.GetExtension(filePath).ToLower();
      string extractedContent = null;

      Cursor = Cursors.WaitCursor; // Показати курсор очікування

      try
```

```
        {
            if (extension == ".txt")
            {
                extractedContent = File.ReadAllText(filePath);
            }
            else if (extension == ".pdf")
            {
                extractedContent = ExtractTextFromPdf(filePath);
            }
            else if (extension == ".docx")
            {
                extractedContent = ExtractTextFromDocx(filePath);
            }
            else
            {
                MessageBox.Show("Підтримуються лише файли .txt, .pdf, .docx.",
                                "Непідтримуваний тип файлу",
                                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                SetupInitialState(); // Повернення до початкового стану, якщо тип не підтримується
                return;
            }

            if (extractedContent != null)
            {
                loadedFileContent = extractedContent;
                loadedFilePath = filePath;

                loadedFileLabel.Text = $"Завантажено: {Path.GetFileName(filePath)}";
                dropPanel.Visible = false;
```

```
            queryPanel.Visible = true;
            // Показуємо лише початок вмісту для попереднього перегляду
            answerLocalKbTextBox.Text = $"Файл '{Path.GetFileName(filePath)}' типу {extension.ToUpper()} завантажено.\nВміст (перші {Math.Min(loadedFileContent.Length, 300)} символів):\n{loadedFileContent.Substring(0, Math.Min(loadedFileContent.Length, 300))}...";
        }
        else
        {
            // Якщо extractedContent == null, це означає, що сталася помилка під час вилучення,
            // і повідомлення про помилку вже було показано в ExtractTextFromPdf/Docx
            SetupInitialState();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Загальна помилка обробки файлу: {ex.Message}",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        SetupInitialState(); // Повернення до початкового стану
    }
    finally
    {
        Cursor = Cursors.Default; // Повернути звичайний курсор
    }
}
            }
            else
```

```
            {
                MessageBox.Show("Підтримуються лише файли .txt (для прикладу), .pdf, .docx.", "Непідтримуваний тип файлу", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
    }

    private void UnloadKbButton_Click(object sender, EventArgs e)
    {
        SetupInitialState();
    }
private string ExtractTextFromPdf(string filePath)
{
    try
    {
        StringBuilder text = new StringBuilder();
        using (PdfReader pdfReader = new PdfReader(filePath))
        using (PdfDocument pdfDoc = new PdfDocument(pdfReader))
        {
            for (int i = 1; i <= pdfDoc.GetNumberOfPages(); i++)
            {
                ITextExtractionStrategy strategy = new SimpleTextExtractionStrategy();
                string currentPageText = PdfTextExtractor.GetTextFromPage(pdfDoc.GetPage(i), strategy);
                text.Append(currentPageText);
            }
        }
        return text.ToString();
    }
```

```
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка вилучення тексту з PDF: {ex.Message}", "Помилка PDF", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null; // Або повернути порожній рядок, або кинути виняток далі
    }
}

private string ExtractTextFromDocx(string filePath)
{
    try
    {
        StringBuilder text = new StringBuilder();
        using (WordprocessingDocument wordDoc = WordprocessingDocument.Open(filePath, false)) // false для read-only
        {
            if (wordDoc.MainDocumentPart?.Document?.Body != null)
            {
                text.Append(wordDoc.MainDocumentPart.Document.Body.InnerText);
                foreach (var para in wordDoc.MainDocumentPart.Document.Body.Elements<Paragraph>())
                {
                    text.AppendLine(para.InnerText);
                }
            }
        }
        return text.ToString();
    }
    catch (Exception ex)
    {
```

```
        MessageBox.Show($"Помилка вилучення тексту з DOCX: {ex.Message}",
"Помилка DOCX", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null; // Або повернути порожній рядок, або кинути виняток далі
    }
}
        private async void AskLocalKbButton_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(loadedFileContent))
            {
                MessageBox.Show("База знань не завантажена.", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            string question = queryLocalKbTextBox.Text;
            if (string.IsNullOrWhiteSpace(question))
            {
                MessageBox.Show("Введіть запитання.", "Порожній запит",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }

            answerLocalKbTextBox.Text = "Обробка запиту до локальної бази...";
            askLocalKbButton.Enabled = false;
askLocalKbButton.Enabled = false;
    Cursor = Cursors.WaitCursor;

    try
    {
        // string relevantFragment = await
CallNlpServiceForLocalSearch(loadedFileContent, question);
```

```
        string relevantFragment = FindRelevantFragmentUsingNlp(loadedFileContent, question);
        await Task.Delay(500); /

        if (!string.IsNullOrWhiteSpace(relevantFragment))
        {
            answerLocalKbTextBox.Text = $"Знайдено релевантний фрагмент (NLP-пошук):\n{relevantFragment}";
        }
        else
        {
            answerLocalKbTextBox.Text = "Інформацію по вашому запиту не знайдено в завантаженій базі (NLP-пошук).";
        }
    }
    catch (Exception ex)
    {
        answerLocalKbTextBox.Text = $"Помилка під час NLP-пошуку: {ex.Message}";
    }
    finally
    {
        askLocalKbButton.Enabled = true;
        Cursor = Cursors.Default;
    }
}

private string FindRelevantFragmentUsingNlp(string documentText, string query)
```

```
{
    string lowerQuery = query.ToLower();
    var sentences = documentText.Split(new[] { '.', '!', '?' },
StringSplitOptions.RemoveEmptyEntries);
    foreach (var sentence in sentences)
    {
        if (sentence.ToLower().Contains(lowerQuery))
        {
            int index = sentence.ToLower().IndexOf(lowerQuery);
            int start = Math.Max(0, index - 30);
            int length = Math.Min(sentence.Length - start, lowerQuery.Length + 60); //
            return $"...{sentence.Substring(start, length).Trim()}...";
        }
    }
    return null;
}
```