МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
« » травня 2024

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна технологія інтелектуального аналізу даних для формування індивідуальної дієти»
здобувача групи ІН.м-21н Гагенка Миколи Андрійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Микола ГАГЕНКО
(підпис)

Керівник
кандидат технічних наук,
доцент В'ячеслав МОСКАЛЕНКО _____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-21н Гагенка Миколи Андрійовича

1. Тема роботи: «Інформаційна технологія інтелектуального аналізу даних для формування індивідуальної дієти»
затверджую наказом по СумДУ від *«8» березня 2024 року № 0234-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 21 травня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області та постановка завдань дослідження. 2) Огляд технологій розробки інтелектуальних систем на основі LLM. 3) Розробка моделі інформаційної системи. 4) Вибір технологій реалізації та методології розробки. 5) Розробка програмної реалізації. 6) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання « » р.
Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області та постановка завдань дослідження*		
2	*Огляд технологій розробки інтелектуальних систем на основі LLM*		
3	*Розробка моделі інформаційної системи*		
4	*Вибір технологій реалізації та методології розробки*		
5	*Розробка програмної реалізації*		
6	*Аналіз результатів*		
7	*Оформлення пояснювальної записки до кваліфікаційної роботи.*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 69 стр., 4 рис., 1 додаток, 77 використаних джерел.

Тема: Інформаційна технологія інтелектуального аналізу даних для формування індивідуальної дієти.

Мета: Розробити модель інформаційної технології інтелектуального аналізу даних для формування індивідуальної медичної дієти, створити її програмну реалізацію.

Метод: Для досягнення поставленої мети використано велику мовну модель (LLM). До цієї моделі застосовано методи оптимізації великих лінгвістичних моделей у контексті задачі формування індивідуальних дієт.

Результати: Модель, що пропонується, має спеціально розроблені шаблони запитів (prompts) які використовують методи prompt engineering, що дозволяє підвищити ефективність використання моделі LLM та досягнути задовільної точності (Precision) створення індивідуальних дієт. Побудована модель обробляє вхідні дані, які містять інформацію про обмеження та побажання користувача щодо персоналізації дієти, подані у вигляді природної мови. Модель враховує обмеження, а саме медичні рекомендації, протипоказання та особисті побажання користувача.

Ключові слова:

LLM, PROMPT, ДІЄТА, РЕКОМЕНДАЦІЯ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА

ЗМІСТ

ВСТУП

Актуальність. Існуючі системи створення індивідуальних дієт анкетують користувачів і не дозволяють вводити дані природньою мовою, або мають недостатню якість генерації. Моделі Multi-Objective Bilevel Recommender System, A DASH Diet Recommendation System, FML-Based Recommender System та подібні мають цю проблему. Моделі ChatGPT robo-diets, ChatGPT-based recommender частково її вирішують, але не забезпечують належної точності створення дієт. Тому необхідно розробити нову модель інформаційної технології створення індивідуальних дієт, яка вирішить цю задачу.

Об'єкт дослідження. Процес створення індивідуальних дієт.

Предмет дослідження. Модель інформаційної технології створення індивідуальних дієт на основі оптимальної великої лінгвістичної моделі (LLM) та набору спеціальних промтів які дозволяють обробляти вхідні параметри подані у вигляді природньої мови.

Суперечність. Існуючі моделі не забезпечують введення даних природньою мовою або не достатньо ефективні. Має бути вирішена практична задача розроблення нової, більш ефективної за попередні, моделі з можливістю вводу вхідних обмежень у форматі природньої мови для системи створення індивідуальних дієт.

Гіпотеза дослідження. Підвищення ефективності системи створення індивідуальних дієт можливе якщо:

- створити спеціальні шаблони запитів (prompts), які дозволять ефективно обробляти вхідні дані системи у неструктурованому вигляді;

- створити спеціальні шаблони запитів (prompts) які будуть враховувати та допоможуть обробити додаткові вхідні обмеження типу: локальність продуктів, бюджет, особливі медичні показання;

- експериментально перевірити, з метою підвищення ефективності системи шляхом підбору нових комбінацій, методів та моделей LLM які відрізняються від вже застосованих іншими дослідниками комбінацій;

- експериментально порівняти якість роботи системи з різними великими лінгвістичними моделями (LLM), з метою утворення закономірностей підвищення ефективності при зменшенні витрат.

Новизна результатів. На відміну від існуючих моделей ChatGPT robo-diets, ChatGPT-based recommender авторів Paweł Niszczota, Iga Rybicka, Ilias Papastratis та ін., запропонована модель враховує недоліки обраних попередниками методів prompt engineering. Нові спеціально розроблені промпти дозволяють підвищити ефективність використання моделі LLM та збільшити точність (Precision) створення дієт.

Структура роботи. Загальне ознайомлення, теоретичні та експериментальні дослідження, обробка інформації, впровадження результатів.

Впровадження. Результати роботи будуть використані для розробки інтелектуальної інформаційної системи створення індивідуальних дієт. Система буде автономною та позиціонуватиметься як персональний або медичний допоміжний засіб.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан та тенденції розвитку інформаційних технологій підбору дієти

Інформаційні технології підбору дієт представляють собою спеціалізовані системи аналітично-рекомендаційного типу. Головною метою таких систем є генерація оптимального меню харчування для кожного користувача з урахуванням його фізіологічних потреб, медичних обмежень, вподобань у стравах та інших обмежень. Їх ціль полягає в забезпеченні можливості складання персоналізованих меню дієт з певного переліку продуктів та з урахуванням індивідуальних особливостей, і цим полегшити досягнення конкретних цілей, таких як контроль ваги, дотримання лікувальної дієти або загальна підтримка здорового харчування. Типовими сферами застосування інформаційних систем підбору дієт є медицина, дієтологія, спорт. На сьогоднішній день перед цією галуззю стоїть ряд актуальних завдань, серед яких вдосконалення алгоритмів аналізу даних, забезпечення точності та безпечності рекомендацій, а також розробка нових більш продуктивних методів підбору дієти з урахуванням індивідуальних потреб кожної людини.

Існує кілька ключових напрямів розвитку інформаційних технологій, які використовуються для створення інтелектуальних систем підбору та рекомендацій дієтичних меню.

Задачі створення систем рекомендації дієт з використанням машинного навчання та гібридних підходів розглядалися в роботах [1–10]. В дослідженнях використовують random forest для класифікації харчових продуктів і надання рекомендацій щодо дієти на основі даних про здоров’я користувачів [1,10]. Застосовують SVM (Support Vector Machine) і kNN (k Nearest Neighbor) для оцінки персоналізованих потреб у поживних речовинах і пропозиції їжі [5,6]. Моделі NN (Neural Networks) використовують для прогнозування харчових уподобань користувачів і рекомендації рецептів [2,4]. Поєднують системи машинного навчання (ML) для прогнозів і rule-based логіки для рекомендацій щодо дієт [3,7].

Результати в максимальних показниках мають прийнятну точність генерації рекомендацій, в цілому використання гібридних систем перевершує за ефективністю монотехнологічні системи. Переваги ML підходу включають високу точність рекомендацій харчових планів на основі даних про здоров'я та побажань користувача, гарну адаптованість до нових користувачів, можливість обробки великих об'ємів даних про користувача та продукти, фіксування складних взаємозв'язків про дієти та здоров'я. Недоліками даного підходу можна вважати непрозора логіка рекомендацій, вірогідність некоректних рекомендацій, потреба у великих наборах даних для навчання, потреба в постійній актуалізації критичних даних для запобігання проблем «холодного старту». Загалом дослідження демонструють багатообіцяючі можливості використання ML підходів та їх гібридів для надання персоналізованих рекомендацій щодо дієт.

Задачі створення систем рекомендації дієт з використанням Crowdsourcing підходу розглядалися в роботах [11–13]. Дослідження використовують типові для Crowdsourcing підходу Human-Intelligence Tasks методи опитування для збору персоналізованої інформації про якість дієт [13], зворотній зв'язок щодо вибору дієти за принципом Traffic light [12], застосування спеціалізованих платформ що надають експертну інформацію [11], а також поєднання з іншими рекомендаційними алгоритмами. Результати показують можливість створення рекомендацій дієт які частково або повністю підкріплені експертною інформацією, або інформацією від надійних джерел. Перевагами є достатня простота технології, більш висока довіра до рекомендацій. Недоліками даного методу є те що не вся інформація проходить якісну валідацію і можливі неякісні рекомендації, якість кінцевих рекомендацій досить сильно залежить від персональних якостей найманого виконавця, постійна оплата та модерація даних для підтримки актуальності та безпечності даних. Метод доцільний для систем рекомендації дієт з високими вимогами до точності, пріоритетом експертних знань, наявністю значного пулу експертів та за умови відповідного фінансового забезпечення.

Задачі створення систем рекомендації дієт з використанням Content-based, Collaborative filtering підходів та їх модифікації розглядалися в роботах [10,14]. У дослідженнях використовуються Content-based filtering для рекомендації продуктів, подібних до харчових уподобань користувача [10] та Collaborative filtering, щоб пропонувати страви, які подобаються схожим користувачам [14]. Результати показують що системи рекомендацій, які використовують ці підходи, змогли надати користувачам індивідуальні поради щодо дієт. Методи мають переваги в тому що мають гарну персоналізацію без великих даних про користувача, можливість підлаштування рекомендацій у схожих за обмеженнями користувачів, динамічна підлаштування під зміну вподобань користувача. Однак методи вимагають складних правил про допустимі продукти, є проблеми з «холодним стартом» та «перевантаженням». Отже метод підходить для ситуацій коли різноманітність рекомендацій не є занадто важливою, також у реалізації є достатньо простим та не потребує багато ресурсів.

Задачі створення систем рекомендації дієт з використанням NLP підходу розглядалися в роботах [2,9]. Для аналізу текстів про їжу, рецепти, медичні записи використовувались методи NLP та нейронні мережі для оцінок харчових потреб та підбору рекомендацій дієт на основі користувацьких даних. Рекомендаційні системи NLP змогли надати персоналізовані дієти згідно з запитами користувачів. Під час тестування було виявлено що користувачам подобається рекоменлаційна система у вигляді діалогу з чат-ботом, немалою перевагою є наближення до розпізнавання неструктурованих вхідних даних у вигляді природньої мови для виділення побажань та обмежень з запиту користувача. Недоліками даного підходу є недосконаліть класичних методів NLP що може призводити до некоректного розпізнавання запиту і некоректних рекомендацій.

Задачі створення систем рекомендації дієт з використанням методів розпізнавання зображень розглядалися в роботах [15–17]. Основні методи, які використовуються в дослідженнях для аналізу та розпізнавання зображень харчових продуктів, включають моделі глибокого навчання, згорточні нейронні мережі (CNN), а також більш традиційні підходи до машинного навчання, метод

опорних векторів (SVM). Структура глибокого навчання під назвою DeepFood, використовує CNN для класифікації зображень їжі та регресію для оцінки об’єму [15]. Глибоке машинне навчання застосовується для розпізнавання та класифікації їжі на основі зображень [17]. Результати показують високу точність розпізнавання їжі та об’ємів порцій, а також аналізу раціону з подальшим генеруванням рекомендацій його покращення. Результати в цілому демонструють доцільність існування автоматизованої системи оцінки раціону на основі зображень їжі. Основні переваги цих використання розпізнавання зображень, включають об’єктивність і послідовність порівняно з ручним записом, менше навантаження на користувачів і масштабованість. Системи фіксують більше деталей про приготування їжі порівняно з традиційними звітами. Однак існують обмеження щодо точності в складних реальних середовищах, якість результатів напряму залежить від великих наборів даних для навчання, також деякі продукти та страви залишаються складними для розпізнавання. Підсумовуючи, галузь автоматизованої оцінки раціону на основі зображень демонструє прогрес у застосуванні комп’ютерного зору та глибокого навчання для таких завдань, як розпізнавання їжі, класифікація та оцінка об’єму порцій але потребує подальших досліджень та поєднання з іншими технологіями.

Задачі створення систем рекомендації дієт з використанням Genetic та Evolutionary алгоритмів розглядалися в роботах [18,19]. Основними методами, які досліджуються в цих роботах для створення оптимальних харчових меню на основі математичних моделей і вхідних обмежень користувача, є алгоритм bacterial foraging optimization (BFO) [19] та генетичні алгоритми [18]. Результати дослідження дали зрозуміти що підхід BFO та генетичний алгоритм можуть створювати збалансоване меню оптимізоване за такими критеріями харчування, як калорії, мікроелементи, вартість тощо. Дані алгоритми оптимізації можуть здійснювати пошук у великих просторах рішень, щоб знайти глобально оптимальні рекомендації щодо їжі. Недоліками є те що генетичні та еволюційні алгоритми мають достатньо високу обчислювальну складність, не завжди підходять для складних проблем, вимагають ефективних методів оцінювання,

потребують налагодження параметрів та стикаються з проблемою "прокляття розмірності". Отже метод підходить показує гарну точність але в купі з затратністю може бути неоптимальним рішенням в для даної задачі.

Задачі створення систем рекомендації дієт з використанням knowledge based та експертних систем розглядалися в роботах [2,3,11,20,21]. Джерела досліджують методи засновані на knowledge based та експертних системах для створення рекомендацій створення дієт, вони перетворюють експертні знання з питань харчування в бази знань із логічними правилами та фактами. Інформація про існуючі вподобання та обмеження користувача також вносяться в бази знань [21]. Результати показують що система може надати персоналізовані рекомендації щодо дієт з урахуванням медичних показань та рекомендацій [20], створений за цим підходом чат-бот надав розумні відповіді на запити щодо рекомендацій їжі [2]. Основною перевагою підходів, заснованих на знаннях, є можливість пояснення прийнятих рішень, що стоять за пропозиціями та достатньо висока точність рекомендацій заснованих на знаннях експерта. Незважаючи на свою ефективність, оскільки наука про харчування стрімко розвивається, системи цього типу значною мірою залежать від ручного внесення знань експерта, що ускладнює їх масштабування. Крім того, подібні системи не володіють гнучкістю щодо невизначеності або нових ситуацій, які не представлені в їх базі знань. Отже метод підходить для ситуацій коли перелік можливих рекомендацій не занадто великий та в поєднанні з іншими технологіями може забезпечити гарну якість відповідей.

Задачі створення систем рекомендації дієт з використанням онтологій розглядалися в роботах [22–25]. Використовують методи побудови онтологічних формальних моделей що представляють знання про харчування з правилами семантичного висновку для надання персоналізованих порад щодо дієти [22,23], використовують поєднання онтологій з додатковими методами, такими як дерева рішень [25]. Системи були перевірені рядом експериментів, та надали задовільні рекомендації з достатнім рівнем точності щодо дієт, які відповідають знанням закодованим в онтологіях, система [22] продемонструвала здатність пропонувати

відповідні заміни їжі та порції, персоналізовані для рівня ожиріння. Перевагами використання онтологій є те що вони утворюють зрозумілу структуру знань, що робить рекомендації більш прозорими та зрозумілими, також онтології дозволяють описувати семантичні зв'язки між предметами, що веде до більш точного й релевантного прогнозування. Недоліками даного підходу є складність побудови оскільки онтології потребують значних зусиль для розробки та підтримки, що може бути витратним, динамічні уподобання користувачів і профілі смаків можуть бути неправильно інтерпретовані в умовах недостачі знань. Також до недоліків можна віднести обмежену масштабованість, бо онтології можуть стати громіздкими й складними для обробки з розширенням даних, що може обмежувати можливості їх застосування. Метод підходить для систем, орієнтованих на чіткі знання, і є перспективним для рекомендацій щодо дієти заснованої на чітких медичних рекомендаціях. Проте, використання виключно попередньо визначених онтологій обмежує адаптивність та персоналізацію і потребує вдосконалення іншими методами.

Сучасні системи намагаються поєднувати декілька технологій разом із простими у використанні програмами та інтерфейсами, щоб надавати персоналізовані, інтелектуальні рекомендації щодо дієти та планування їжі з урахуванням стану здоров’я, смаків і обмежень людини. Мета полягає в тому, щоб зробити рекомендації щодо харчування більш доступними та ефективними завдяки персоналізації та інноваційним підходам до отримання вхідної інформації.

Згідно з останніми дослідженнями [26,27], великі мовні моделі (LLM) проявили значний потенціал для покращення систем рекомендацій у різних аспектах. Вони можуть діяти як zero-shot ранжувальники, використовуючи свої універсальні здатності вирішення задач та механізми self-attention для ранжування елементів враховуючи вподобання користувачів та їх історію взаємодії. LLM можуть використовувати високоякісні репрезентації текстових ознак та обширне покриття зовнішніх знань з метою підвищення якості та різноманітності рекомендацій. Зокрема, ці моделі мають можливість генерувати пояснення для

рекомендацій під час діалогової взаємодії з користувачем у вигляді природньої мови, та якщо потрібно враховувати фідбек користувача або контекстуальну інформацію.

1.2 Аналіз моделей і методів оброблення природньої мови для аналізу довідкової інформації

Природна мова, на відміну від побудованих і формальних мов, постійно змінюється та розвивається під впливом людського досвіду. Ця динамічність, звична для людей, створює труднощі при спробах обробити цю мову та передати зрозумілий контекст комп’ютеру. Обробка природної мови (Natural Language Processing, NLP) є галуззю штучного інтелекту та комп'ютерної лінгвістики, що націлена на створення систем та алгоритмів для аналізу, розуміння та генерації людської мови в її природній формі.

NLP галузь можна розділити на дві підмножини: розуміння природної мови (Natural language understanding, NLU) та генерація природної мови (Natural language generation, NLG).

NLU представляє собою частину процесу обробки природної мови, яка використовує синтаксичний і семантичний аналіз тексту та мовлення для визначення значення речення. Синтаксис стосується граматичної структури речення, тоді як семантика вказує на його передбачуване значення. Крім того, NLU встановлює відповідну онтологію: структуру даних, яка визначає зв'язки між словами та фразами. Хоча цей процес є природним для людей під час розмови, комбінація цих аналізів необхідна для того, щоб машина могла розуміти передбачуване значення різних текстів. Серед поширених завдань в рамках NLU можна виділити аналіз настроїв, розпізнавання іменованих сутностей, семантичний аналіз і машинний переклад.

NLG охоплює розробку алгоритмів та моделей для комп’ютерної генерації тексту який не відрізняється від людської мови. Генерація природної мови є процесом створення текстової відповіді людською мовою на основі введених даних. NLG також охоплює можливості підсумовування тексту, що дає змогу

створювати звіти на основі вхідних документів, зберігаючи при цьому інформаційну цілісність. Основна мета NLG полягає в тому, щоб надати машинам можливість створювати плавний, зв'язний та інформативний текст шляхом вибору та організації слів, фраз і речень таким чином, що передає певне повідомлення або ідею. Поширенними завданнями NLG є аналіз та резюмування тексту, створення діалогів і мовний переклад.

Техніки обробки природної мови (NLP) охоплюють два основні підходи: системи, що базуються на правилах, та системи машинного навчання. Системи на основі правил працюють із заздалегідь визначеними правилами та лінгвістичними знаннями для аналізу тексту, тоді як системи машинного навчання використовують статистичні алгоритми та моделі для виявлення закономірностей у даних і поступового підвищення точності. Процес обробки природної мови передбачає розділення тексту на найменші одиниці (слова, фрази, речення) та аналіз їхньої структури й значення. Цей процес включає такі основні етапи [28]:

1) Токенізація: розбиття фрагменту тексту на окремі слова чи фрази, відомі як токени. Токенізація є першим кроком у NLP, оскільки вона дозволяє комп'ютеру аналізувати та розуміти структуру тексту.

2) Розмітка частин мови (Part-of-Speech Tagging): ідентифікація граматичної ролі кожного слова чи фрази у тексті шляхом позначення їх відповідними частинами мови, такими як іменник, дієслово, прикметник тощо. Ця інформація є важливою для розуміння значення тексту.

3) Синтаксичний аналіз (Parsing): аналіз граматичної структури речення для визначення його складових та зв'язків між ними. Ідентифікація різних частин мови в реченні та встановлення їхніх синтаксичних відношень.

4) Семантичний аналіз: розуміння сенсу тексту за межами його граматичної структури. Аналіз семантичних зв'язків між словами та фразами в реченні для вилучення їхнього значення.

Завдання аналізу довідкової інформації в тексті із застосуванням NLP технології полягає у виявленні та інтерпретації ключових елементів тексту: смислові зв'язки, тема, тон мови та основні пункти, які впливають на контекст.

Аналіз може проводитися як над структурованим, так і над неструктурованим текстом, а також вимагати застосування різноманітних методів обробки тексту.

Розпізнавання інформації в тексті поділяється на такі основні завдання [29]:

1) Розпізнавання мовлення: конвертація звукового мовлення в текст дозволяє комп'ютерам розуміти та аналізувати мовні дані. Це важлива стадія у розвитку технологій, таких як голосові асистенти та системи автоматичних відповідей на запитання.

2) Розпізнавання іменованих сутностей (NER): визначення та категоризація сутностей, таких як люди, організації та місця розташування в тексті, є ключовим для численних застосувань, включаючи пошукові системи, аналіз соціальних мереж та системи підтримки клієнтів. NER може бути використаний для визначення ключових сутностей у тексті, що сприяє поліпшенню ефективності пошуку та аналізу.

3) Аналіз настроїв: визначення тону чи настрою, присутнього у фрагменті тексту, використовується для аналізу відгуків клієнтів, моніторингу соціальних мереж та управління репутацією бренду. Такий аналіз допомагає визначити, чи має відгук позитивний, негативний або нейтральний характер, що є корисним для маркетингу та управління відносинами з клієнтами.

4) Машинний переклад: процес перекладу тексту з однієї мови на іншу використовується для забезпечення глобальної комунікації та електронної комерції. Машинний переклад допомагає подолати мовні бар'єри, дозволяючи особам з різними мовними навичками спілкуватися між собою.

5) Класифікація тексту: сортування тексту за попередньо визначеними категоріями використовується для автоматичного сортування, наприклад, фільтрації спаму або категоризації новин за темами. Класифікація тексту може бути бінарною (дві взаємовиключні категорії) або багатокласовою (більше двох категорій).

6) Відповіді на запитання: створення чатботів та віртуальних асистентів, які можуть розуміти й відповідати на питання користувачів, застосовується для надання відповідей на запитання природною мовою. Системи відповідей можуть

бути відкритими або закритими, залежно від використовуваних джерел інформації для формування відповідей.

7) Резюмування тексту: створення короткого викладу довшого фрагменту тексту дозволяє конденсувати великі обсяги інформації, сприяючи швидкому розумінню основних ідей та ключових моментів. Резюме тексту може бути створено за допомогою витягування ключових фраз або генерації нового тексту, що відображає основні ідеї оригінального матеріалу.

Типові моделі та методи NLP для завдань екстрагування та аналізу корисної інформації з тексту неявно поділяються на кілька основних категорій:

1) Моделі на основі правил: цей підхід покладається на попередньо задані набори правил. Подібні системи зазвичай розробляються експертами в галузі лінгвістики та інформатики, які вручну створюють набір правил для визначення граматичної структури мови, семантичних ролей та інших лінгвістичних особливостей. Вони часто використовуються для таких завдань, як токенізація, розмітка частини мови та розпізнавання іменованих сутностей. Системи, засновані на правилах вже давно використовуються для обробки та аналізу тексту, покладаючись на створені вручну лінгвістичні правила та шаблони для вилучення інформації з тексту, але були обмежені в своїй здатності обробляти складні мовні структури [30].

2) Статистичні моделі: цей клас моделей застосовує імовірнісні алгоритми та методи машинного навчання, такі як дерева рішень, логістична регресія, методи опорних векторів та приховані марковські моделі. Вони широко використовуються для аналізу тону тексту, тематичного моделювання, а також можуть застосовуватися для екстракції ключових слів/фраз на основі статистичних частотних показників [30].

3) Нейромережеві моделі: доволі сучасний підхід заснований на глибокому навчанні з використанням різних архітектур нейронних мереж, таких як рекурентні (RNN), згорткові (CNN) та моделі трансформерів. Вони успішно застосовуються для багатьох складних завдань NLP, включно з генерацією тексту, класифікацією, відповідями на запитання та машинним перекладом [30].

4) Трансформерні моделі: це категорія архітектур нейронних мереж, які використовують механізми уваги для ефективної обробки послідовностей даних і дозволяє трансформаторам краще розуміти та моделювати контекстуальні зв’язки в мові порівняно з попередніми методами [31]. Вони досягли видатних результатів у різноманітних галузях, включаючи обробку природної мови, машинний переклад, синтез мовлення та обробку зображень. Найвідомішими прикладами трансформерних моделей є BERT, GPT та інші. Трансформерні моделі часто розглядаються окремо через їхню унікальну архітектуру, ефективність, масштабованість та значний вплив на галузь штучного інтелекту [32].

На основі архітектури трансформерів великі мовні моделі (Large language model, LLM) стали проривом у обробці та генерації природної мови. LLM - це трансформаторні нейронні мережі, які навчаються на величезній кількості текстових даних, щоб набути повного розуміння мови та здатності генерувати. Моделі мають кодер, який кодує вхідний текст у числове представлення, і декодер, який генерує вихідний текст на основі закодованих представлень [33,34].

Великі мовні моделі володіють рядом переваг, що робить їх важливими інструментами в області обробки природної мови. Зокрема, LLM можуть розуміти та генерувати текст, що має високу зв’язність та плавність, надаючи людям враження, що цей текст створений людиною. Крім того, вони здатні виконувати різноманітні завдання природної мови, такі як відповіді на запитання, узагальнення тексту та переклад мови, що сприяє їх широкому застосуванню в різних областях. Також, завдяки можливості налаштування для конкретних завдань або сфер застосування, LLM можуть досягати кращої продуктивності [33].

Також, існують певні недоліки, пов'язані з використанням LLM. Перш за все, для їх навчання та використання потрібні значні обчислювальні ресурси. Крім того, дані моделі можуть генерувати упереджені, непослідовні або галюцинаторні результати, особливо на нерозповсюджених вхідних даних, що створює певні труднощі в їх застосуванні, але існють методи подолання цього, наприклад RAG.

Крім цього модель в загальновідомому сенсі насправді не розуміє текст, що обробляє, тому це може статит причиною виходу за межі заданог контексту. В моделях загального користування, існує ризик, що вони можуть запам'ятовувати та ненавмисно виводити конфіденційну інформацію з навчальних даних або з даних які моедль отримує в процесі роботи. Нарешті, складно інтерпретувати процес прийняття рішень у моделі, що може призводити до спроб вирішення проблеми "чорного ящика" [33].

Великі мовні моделі стрімко розвиваються в останні роки, з’явибагато сучасних моделей, які демонструють надзвичайне розуміння мови та можливості генерації. Ринок моделей штучного інтелекту в 2023-2024 роках можна розділити на дві основні категорії: комерційні (пропрієтарні) та відкриті (open-source) моделі.

Комерційні моделі розробляються великими технологічними компаніями, такими як Google, Microsoft, OpenAI, Anthropic та інші. Їхній вихідний код не оприлюднюється, а для використання потрібно сплачувати ліцензійні збори або підписку. Прикладами є GPT-3 від OpenAI (175 млрд параметрів), PaLM від Google (540 млрд параметрів, мультимодальна), Jurassic-1 від AI21 Labs (178 млрд параметрів). Великі інвестиції забезпечують цим моделям надзвичайну потужність та точність. Вони часто використовуються у корпоративних рішеннях: чат-ботах, системах автоматичного письма, аналізі даних тощо [34–36].

Натомість, відкриті моделі розробляються спільнотами дослідників та ентузіастів. Їх вихідний код доступний для вільного використання та модифікації згідно з ліцензіями open-source. Популярними є BLOOM від Hugging Face (176 млрд параметрів, багатомовна), OPT від Meta AI (різні розміри до 175 млрд параметрів), GPT-Neo від EleutherAI (до 2.7 млрд параметрів), FLAN від Google (мультизадачна). Незважаючи на меншу потужність, відкриті моделі забезпечують прозорість, можливість адаптації та широко використовуються в дослідженнях, навчанні та спеціалізованих додатках. Серед них безкоштовна версія ChatGPT, створена на базі спеціально налаштованої версії моделей GPT OpenAI, стала найвідомішим інструментом, демонструючи потужність LLM у

генерації тексту, і сприяючи широкому спектру програм від обслуговування клієнтів до генерації змістовного тексту [34,36,37].

Дослідження розділу рекомендаційних систем з застосуванням LLM моделей в основному зосереджуються на трьох основних напрямах: створення рекомендаційних систем на базі LLM, підвищення ефективності самих LLM, оцінювання ефективності LLM та систем які вони утворюють. Проаналізовані роботи підсумовують, що LLM демонструють значний потенціал для вдосконалення якості рекомендацій завдяки своїм потужним можливостям обробки та генерації даних на природній мові. Однак залишаються певні невирішені питання, зокрема: відсутність теоретичних гарантій результатів та їх відтворюваності для рекомендаційних систем на основі LLM, потреба подальшого вдосконалення методів оптимізації та оцінки ефективності самих LLM, необхідність комплексного аналізу точності, упередженості та безпечності цих моделей. Вирішення зазначених проблем може суттєво посилити практичне застосування LLM для якісних систем рекомендацій.

Задачі створення рекомендаційних систем на основі large language model (LLM) розглядалися у роботах [38–60]. Використовують попереднє навчання (pre-training) LLM на великих наборах даних, налаштування підказок (prompt engineering), вбудовування крос-модальних ознак та інтеграції з графами знань. Результати досліджень демонструють високу ефективність LLM для контекстно-залежних персоналізованих рекомендацій. Недоліками даного підходу є відсутність теоретичних гарантій результатів та відсутність стійких гарантій повторюваності результатів.

Задачі підвищення ефективності великих мовних моделей (LLM) розглядалися у роботах [61–69]. Використовують байєсівську оптимізацію, перенос навчання, зміну функцій активації, дистиляцію моделей, gradient pruning, квантування. Результати досліджень продемонстрували значне підвищення ефективності LLM за різними метриками. Подальші дослідження мають враховувати точність та упередженість моделей, мають бути проведені

експерименти з комплексного застосування декількох підходів одночасно, також мають бути вдосконалені ансамблеві та адаптивні методи.

Задачі оцінки ефективності, розробки системних методик порівняння LLM, аналізу кореляції метрик якості LLM із задачами, встановлення залежності ефективності від розміру моделей розглядалися в роботах [61,70–72]. Пропонують різні методики оцінки ефективності, такі як: порівняння за уніфікованим набором тестів, застосування поведінкових зондів (behavioral probes), мета-аналіз, тестування на спеціальних даних. Отримані результати висвітлюють різні аспекти порівняльної ефективності LLM. Основними недоліками є відсутність універсальної системи оцінювання та складність аналізу упередженості. В подальших дослідженях мають бути розроблені комплексні підходи, що поєднують методики.

Для дисертаційного дослідження обрана тематика застосування prompt engineering, emotional prompting та підбору оптимальних моделей LLM для розроблення системи створення індивідуальних дієт. Задачі схожого типу розглядалися у роботах [10,27,73–75]. Використовують методи prompt engineering та великі мовні моделі: ChatGPT, GPT-3, GPT-4. Результати досліджень пропонують створені рекомендаційні системи або методи для їх створення. Недоліками даних досліджень є обмеженість використання методів оптимізації промтів для цієї задачі, не проаналізовані більш бюджетні моделі LLM які можуть бути використані для цієї задачі, кількість некоректних рекомендацій наданих системою не є нульовою. Отже, ефективність вирішення цієї задачі гіпотетично можна підвищити.

Для більш ефективного вирішення даної задачі треба провести теоретичне дослідження в рамках якого має бути розроблена модель інформаційної технології створення індивідуальних дієт на основі large language models (LLMs), мають бути розроблені спеціальні промти (prompts) які допоможуть моделі підвищити ефективність генерування рекомендацій, для розробки промтів має бути перевірена методика підвищення якості генерації - emotional prompts. Модель інформаційної технології має бути покращена шляхом експериментальної

перевірки раціональних комбінацій різних моделей LLM, та підбору достатньої складності та об’єму для промптів.

Ефективність створеної моделі інформаційної технології має бути підтверджена експериментально. Мають бути проведені експериментальні дослідження із застосуванням методу модельних досліджень. Суть експериментів є у проведені аналітичного порівняння результатів які буде генерувати модель інформаційної технології та еталонних даних з тестової вибірки. Оцінка ефективності (Precision) моделі визначається через перевірку гіпотез щодо кількості правильно та неправильно сформованих вихідних результатів, які входять у матрицю помилок (True Positive, False Positive).

1.3 Формалізована постановка задачі

Вхідними даними інформаційної системи є запити користувачів, подані у вигляді природньої мови. Типовий запит містить інформацію про обмеження та побажання користвача щодо персоналізації дієти.

Запит користувача містить певні ключові терміни, згідно з якими буде обрано медичну дієту, а також пул допустимих локальних продуктів. Для тестування та налаштування системи має бути створена тестова вибірка вхідних запитів, типу «вхідні параметри : коректна дієта».

Необхідно, щоб розроблена інформаційна інтелектуальна система могла створювати індивідуальні дієти, враховуючи вхідні обмеження, надані користувачем, а саме: медичні рекомендації, протипоказання та персональні побажання. Система повинна враховувати всі вхідні обмеження та повертати результат із задовільною точністю (Precision) генерації результатів. Вона має бути самостійним продуктом, який може бути використаний для особистого користування або як допоміжний експериментальний засіб у медичній практиці.

Для досягнення поставленої мети треба:

1) Розробити модель інформаційної технології, яка приймає дані у вигляді природньої мови замість класичних методів заповнення форм.

1.1) Розробити спеціально для цієї задачі шаблони запитів (prompts), які будуть обробляти вхідні дані.

1.2) Розробити спеціальні промти (prompts), які допоможуть моделі LLM створити рекомендацію дієтичного меню з локальних продуктів.

1.3) Створити базу даних основних медичних дієт та розробити промти для її зчитування і надання моделі LLM.

1.4) Настроїти модель інформаційної технології шляхом експериментальної перевірки раціональних комбінацій, використовуючи нові шаблони запитань (prompts), методи оптимізації промтів та різні моделі LLM.

2) Провести експериментальне порівняння продуктивності інформаційної технології з використанням великих лінгвістичних моделей (LLM) різної потужності, з метою утворення закономірностей підвищення ефективності при зменшенні витрат.

3) Розробити програмну реалізацію системи для формування індивідуальної дієти.

2 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДЛЯ ФОРМУВАННЯ ІНДИВІДУАЛЬНОЇ ДІЄТИ

2.1 Модель інформаційної технології формування індивідуальної дієти

Базуюсичь на проведеному дослідженні було з’ясовано, що великі мовні моделі, на основі трансформерів, представляють найбільший інтерес та мають значний потенціал для розвитку в галузі обробки природної мови.

Основою для побудови моделі інформаційної технології інтелектуального аналізу даних для формування індивідуальної дієти було вирішено обрати саме велику мовну модель та модифікувати процес її роботи розробленими спеціальними запитами, які дозволять обробляти вхідні параметри подані у вигляді природньої мови.

Для підвищення ефективності порівняно з попередніми дослідженнями, було вирішено застосувати метод prompt engineering для використання LLM у генерації дієтичних меню. Суть методу полягає в ітераційному підборі максимально ідеального промпту для вирішення конкретної задачі. Задача генерації дієтичних меню за медичними рекомендаціями є достатньо трудомісткою, тому доцільно розділити її на декілька етапів і для кожного етапу застосовувати відповідні модифікації промптів.

Реальні моделі, побудовані на основі трансформерів, відрізняються певними модифікаціями, обсягом даних, на яких вони були натреновані, а також розміром контексту, що впливає на необхідні обчислювальні ресурси для їх використання. Тому для кожного типу задач необхідно проводити відповідне тестування та підбирати найбільш оптимальну модель. Або, якщо є достатні ресурси, створювати власну модифікацію.

Предметна область даного дослідження пов'язана із дієтичним харчуванням та здоров'ям людини, відтак точність фінального результату має бути максимально високою і, в ідеалі, не містити хибної інформації. Однак, система може також містити запити, менш вибагливі до точності, які можна обробляти за допомогою менш потужної, а отже дешевшої, моделі. Наразі найпотужнішими

реалізаціями великих мовних моделей вважаються GPT, Gemini, Claude та подібні [36]. Використання можливостей таких моделей у практичних реалізаціях є доволі стандартизованим, що робить можливим підбір оптимальної за продуктивністю та вартістю моделі на будь-якому етапі розробки. Критично важливим критерієм для вибору моделі є її здатність надавати точні та релевантні відповіді на всі запити в межах системи.

Для функціонального моделювання інформаційної системи було створено IDEF0 діаграму, щоб візуалізувати та деталізувати можливості системи. Діаграма, відображена на рисунку 2.1, описує концептуальну архітектуру системи створення дієтичних меню.

Центральний блок моделі представляє ядро системи, де здійснюється безпосередньо процес формування дієтичних рекомендацій. Вхідні дані для цього процесу включають додаткову інформацію про дієти, обмеження щодо дієтичних вподобань, а також доступні продукти харчування. Ці дані надходять із зовнішніх джерел або з накопичених моделлю LLM знань у процесі навчання, після чого вони обробляються та інтегруються в межах системи.

Керуючі впливи на функціонування системи представлені трьома ключовими компонентами: фреймворком, методологією prompt engineering та великою мовною моделлю (LLM). Ці елементи забезпечують необхідну логіку, алгоритм і техніки для генерації релевантних дієтичних рекомендацій на основі вхідних даних. Результатом роботи системи є пропозиція конкретного дієтичного меню, яке повертається до користувача.

Таким чином, IDEF0 модель наочно демонструє структуру та взаємодію основних складових інформаційної системи, що дозволяє здійснювати ефективне функціональне моделювання та розробку даного програмного комплексу.

Рисунок 2.1 – Контекстна діаграма процесу "Система створення дієтичних меню" в нотації IDEF0

Інформаційна система, що пропонується, здійснює аналіз інформації в декілька етапів, послідовність яких показана на рисунку 2.2.

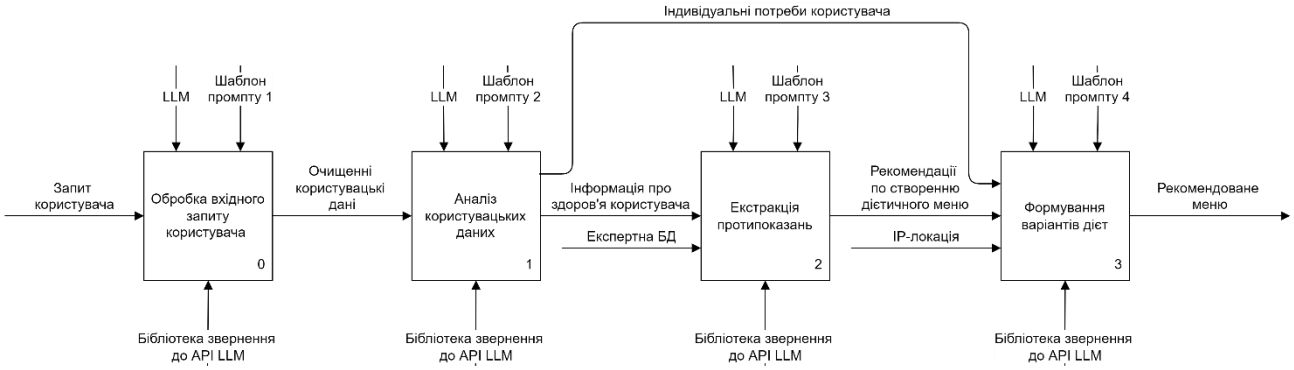

Рисунок 2.2 – Функціональна схема інформаційної технології інтелектуального аналізу даних для формування індивідуальної дієти

Запропонована модель інформаційної технології інтелектуального аналізу даних для формування індивідуальної дієти функціонує за таким алгоритмом:

1) Етап отримання вхідних даних від користувача:

a. Користувач вводить текстовий запит у форматі природньої мови, який може містити іфнормацію про його стан здоров'я, наявні алергії, харчові непереносимості, додаткові вподобання або обмеження.

b. Відбувається перевірка коректності та повноти введених даних засобами LLM.

c. За допомогою спеціалізованого промпту (шаблону запиту) вхідні дані від користувача обробляються великою мовною моделлю (LLM) і зберігаються в структурованому форматі.

2) Етап обробки даних:

a. Підбір відповідних медичних показань здійснюється шляхом порівняння моделлю LLM попередньо сформованої бази даних медичних показань з дієтичними рекомендаціями та інформацією про здоров'я конкретного користувача.

b. З бази даних типових медичних дієт вибирається інформація про рекомендації та обмеження щодо харчування, яка форматується для подальшої передачі LLM.

c. Сформовані дієтичні рекомендації перевіряються більш потужною LLM на предмет виявлення некоректних даних, після чого знайдені неточності виправляються.

d. Здійснюється локалізація продуктів шляхом додавання в промпт інформації про локацію користувача в модель LLM. LLM враховує країну проживання користувача для включення місцевих продуктів та правильної валюти для розрахунку вартості дієти.

e. Сформовані рекомендації щодо створення дієти передаються у спеціальний промпт, після чого модель LLM генерує декілька варіантів персоналізованих дієтичних рекомендацій.

3) Етап виведення результатів:

a. Користувач отримує кілька варіантів дієтичних рекомендацій з зазначеною приблизною вартістю реалізації кожного варіанту.

Процес функціонування моделі зображений на схемі алгоритму (Рис. 2.3). Схема ілюструє послідовність основних етапів обробки вхідних даних та формування рекомендацій дієтичних меню.

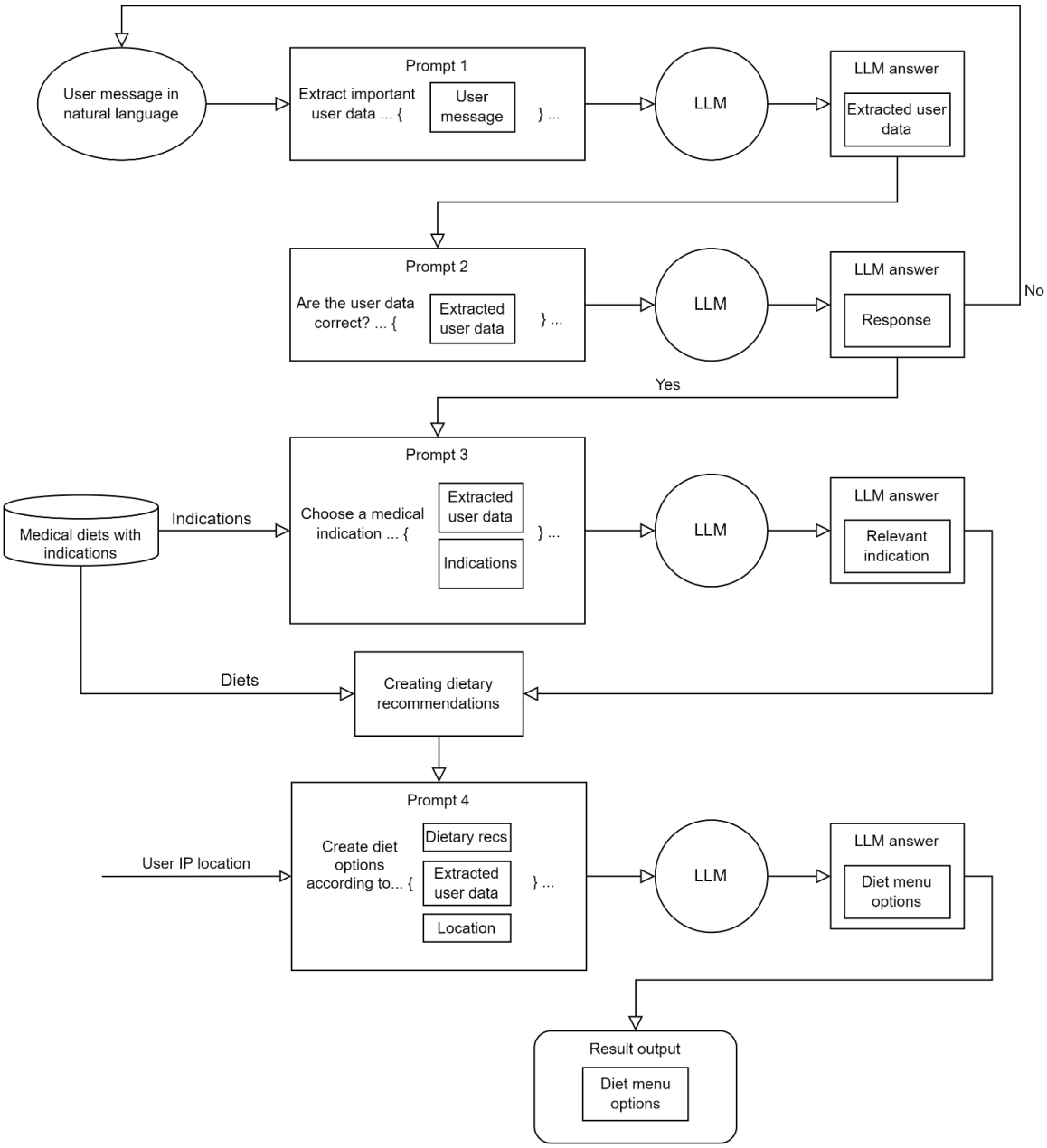

Рисунок 2.3 – Схематичне зображення алгоритму роботи інформаційної системи

Слід зазначити, що в рамках даного дослідження інформаційна система здатна екстрагувати та враховувати при гернерації меню обмежену кількість показників, а саме:

– проблема зі здоров’ям;
– алергії;
– непереносимості;
– побажання що виключити з раціону;

– побажання що додати до раціону.

Система має створювати декілька варіантів денних меню розрахованих на 5 прийомів їжі, що відповідає загальним дієтичним рекомендаціям з харчування.

В якості експертних знань про медичні дієти використовуються дієтичні столи за Певзенром [76].

2.2 Методи настроювання інтелектуальної моделі аналізу і генерації даних

Великі мовні моделі навчаються на колосально великих об'ємах даних, що дозволяє їм накопичувати обширну базу знань. Однак, різні LLM відрізняються за обсягом даних, використаних для тренування, а також за розміром контексту, що обробляється. Відповідно, деякі моделі є більш швидкодіючими, а інші демонструють вищу когнітивну здатність. Попри це, для вирішення специфічних задач або надання вузькоспрямованих відповідей наявних знань LLM може бути недостатньо.

Для вирішення цієї проблеми застосовуються методи настроювання LLM, такі як донавчання, тонке налаштування та промпт-інжиніринг. З огляду на доступність і відносно низьку вартість, промпт-інжиніринг є найбільш практичним підходом. Оскільки інтелектуальна система, що розробляється, базується на LLM, застосування методів промпт-інжинірингу є необхідним етапом її налаштування та оптимізації.

Промпт-інжиніринг являє собою техніку, що дозволяє адаптувати роботу великих мовних моделей під конкретні завдання. Суть методу полягає у створенні спеціальних інструкцій-підказок (промптів), які супроводжують вхідні дані, надані моделі. Ці промпти містять детальні вказівки щодо бажаного формату, стилю та змісту вихідного результату. Таким чином, модель отримує чіткі налаштування на виконання конкретного завдання, на відміну від генерування стандартних, неспеціалізованих відповідей.

Ефективне застосування промпт-інжинірингу вимагає глибокого розуміння можливостей та обмежень великих мовних моделей, а також детального аналізу

цільових завдань. Шляхом ітеративного налаштування та тестування промптів можна досягти значного покращення якості та релевантності результатів, отриманих від мовної моделі.

Найбільш популярними та ефективними методами промптінгу для генеративних мовних моделей є пряме промптування (zero-shot), промптування з використанням прикладів (one-, few-, multi-shot), ланцюжкове промптування (chain-of-thought), ітеративне промптування та самостійне промптування. Також практикуються не стандартні підходи такі як емоційний промптінг.

Пряме промптування передбачає надання моделі лише інструкцій без будь-яких прикладів. Інструкція може бути сформульована у вигляді запитання або визначати конкретну роль для моделі.

Промптування з використанням прикладів доповнює інструкцію конкретними прикладами, що дозволяє моделі краще фокусуватися та отримувати більш точні результати. Розрізняються такі варіанти: one-shot prompt, few-shot prompt та multi-shot prompt, кожен з яких використовує різну кількість прикладів для керування моделлю.

Ланцюжкове промптування передбачає керування штучним інтелектом мовної моделі через серію взаємопов'язаних промптів, дозволяючи моделі будувати відповіді на основі попередніх результатів. Цей підхід є прогресивною формою ланцюжкового промптування і корисний для дослідження складних тем або генерації детального контенту.

Ітеративне промптування передбачає побудову подальших запитань на основі попередніх відповідей моделі, що дозволяє поглиблювати тему або уточнювати неоднозначності початкового виводу.

Самостійне промптування є прогресивною формою ланцюжкового промптування, де сама модель ставить собі питання на основі початкового промпту, що призводить до більш детальних та глибоких відповідей.

Емоційний промптинг передбачає додавання до промпту слів які вкажуть моделі на емоційний підтекст запиту, що на практиці призводить до підвищення якості відповідей [77].

Всі вищеперелічені методи при необхідності мають бути використані для створення шаблонів запитів для досягнення поставленої мети.

Ще одним способом збільшення ефективності системи є підбір LLM, а саме використання більш потужних моделей для більш складних завдань, які цього потребують. Використання більш потужних моделей для простих задач, які не потребують занадто високої точності, є ірраціональним підходом, тому для більш простих задач мають бути використані більш економічні моделі.

Великі мовні моделі здатні опрацьовувати запити як українською, так і англійською мовами. Проте, англійська мова, як правило, використовує меншу кількість токенів порівняно з українською, що робить її більш бажаною для написання запитів-промптів. Це пов'язано з особливостями синтаксичної структури та морфологічної складності української мови.

Попри переваги англійської, в деяких випадках збереження оригінальної мови тексту може бути доцільним. Зокрема, при опрацюванні медичних рекомендацій або специфічної термінології, переклад може призвести до втрати важливої контекстної інформації. У таких ситуаціях доцільно залишати певні фрагменти тексту, особливо вузькоспеціалізовані поняття, у вихідній мовній формі.

Вибір мови запиту-промпту для LLM має враховувати як загальну ефективність опрацювання, так і збереження семантичної точності та контекстуальної доречності результатів. Застосування гнучкого підходу, що поєднує переваги обох мов, дозволяє максимізувати продуктивність та якість роботи LLM в системі.

Промпт можна вважати завершеним якщо відповідь моделі на цей промпт не містить некоректної інформації, а вцілому відповідає очікуваному результату.

Промпт можна вважати ефективно сформованим, якщо генерована моделлю відповідь на цей промпт відповідає кільком критеріям:

– відсутність фактичних помилок або некоректної інформації в тексті відповіді;

– загальна відповідність отриманого результату очікуваному, цільовому результату;

– дотримання необхідного формату, стилю та структури відповіді, визначених у самому промпті.

Таким чином, ключовими показниками успішності промпту є точність фактичного змісту відповіді, її релевантність до поставленого завдання, а також відповідність формальним вимогам, зазначеним у промпті. Лише за умови задоволення всіх цих критеріїв промпт можна вважати остаточно сформованим і здатним забезпечити бажаний результат від великої мовної моделі.

Перший промпт має на меті обробку вхідних даних, наданих користувачем. Промпт визначає роль моделі LLM як «спеціаліст з опрацювання запитів користувачів» та інформує її про необхідність витягнення ключових відомостей з повідомлення користувача. Водночас промпт застерігає про можливу некоректність інформації, наданої користувачем. Варто зазначити, що даний промпт є відносно ресурсоощадним, тому застосування моделі GPT-3.5 є раціональним вибором для його виконання.

Орієнтовний шаблон промпту англійською мовою виглядає наступним чином:

```
«You are a specialist in processing requests from a user who describes information about himself for the purpose of assigning him a therapeutic diet. But there are cases when the user's message is incorrect or contains false information, so first check whether the user's message is an adequate description of information about: the user's health problems, his allergies, intolerances, wishes, what to exclude and what to add, type of diet and other restrictions. If the user's message is inadequate, contains many grammatical, lexical, semantic errors, errors in medical terms, non-existent medical terms or invented made-up diseases, humorous diseases then fill all fields as None. If the user's message is correct, extract the relevant information from the message. User message: {user_message}».
```

Переклад на українську: «Ви спеціаліст з обробки запитів від користувача, який описує інформацію про себе з метою призначення йому лікувальної дієти.

Але бувають випадки, коли повідомлення користувача є некоректним або містить неправдиву інформацію, тому спочатку перевірте, чи є повідомлення користувача адекватним описом інформації про: проблеми зі здоров'ям користувача, його алергію, непереносимість, побажання, що виключити, а що додати, тип дієти та інші обмеження. Якщо повідомлення користувача є неадекватним, містить багато граматичних, лексичних, семантичних помилок, помилок у медичних термінах, містить неіснуючі медичні терміни або вигадані хвороби, жартівливі хвороби, то заповніть усі поля як «None». Якщо повідомлення користувача правильне, витягніть відповідну інформацію з повідомлення. Повідомлення користувача: {user_message}».

Наступний промпт повинен зробити аналіз доступних медичних рекомендацій щодо певних проблем зі здоров'ям та порівняння їх з інформацією про стан здоров'я користувача. За результатами порівняння, промпт визначає, яка з наявних медичних рекомендацій найбільше підходить для користувача. Промпт визначає роль моделі як «фахівець у визначенні відповідності інформації користувача наявним медичним показанням» та інформує її про необхідність порівняння даних користувача з медичними рекомендаціями та вибору найбільш підходящого варіанту. Також, промпт застерігає, що якщо відповідності не знайдено, необхідно повернути значення «None». Даний промпт є відносно не ресурсоємним, тому застосування моделі GPT-3.5 є раціональним вибором для його виконання.

Орієнтовний шаблон промпту англійською мовою виглядає наступним чином:

```
    «'List of medical indications': {indications_list}. You are a
specialist in determining the appropriateness of user information and
available medical indications. Please write the index of the medical
indication that corresponds to the information about the health of the
user: user_info = '{user_info[health_problems]}'. If user_info
contains rubbish not related to medicine or you did not find an exact
match in the 'List of medical indications', then indicate the index
None».
```

Переклад на українську: «'Список медичних показань': {indications_list}. Ви є фахівцем у визначенні відповідності інформації користувача та наявних медичних показань. Напишіть, будь ласка, індекс медичного показання, що відповідає інформації про стан здоров’я користувача: user_info = '{user_info[health_problems]}'. Якщо user_info містить дурниці, не пов'язані з медициною, або ви не знайшли точної відповідності в «Переліку медичних показань», то вказуйте індекс «None».

Третій промпт має на меті запит до LLM про генерування декількох варіантів дієтичних меню на день. Промпт має вказати моделі на регіон проживання користувача для того щоб модель намагалась використовуват типові для цього регіону продукти. Також регіон впливає на орієнтовне формування вартості згенерованих меню. Промпт визначає роль моделі як "кухар-дієтолог" та підкреслює важливість точності виконання цього завдання. Запит застерігає від використання неіснуючих страв, а також від включення до меню алергенних або проблемних для користувача продуктів. Також промпт повідомляє про бажану кількість опцій меню та кількість прийомів їжі на день. Враховуючи важливість та ресурсоємність даного промпту, застосування більш потужної мовної моделі, такої як GPT-4, є раціональним вибором.

Орієнтовний шаблон промпту англійською мовою виглядає наступним чином:

```
    «You are a dietary cook, your work is very important, so do it
responsibly and carefully, do not invent non-existent dishes and
follow dietary recommendations. Taking into account the dietary
recommendations of {diet_rules[diet_name]}, create for me a list of
{3} balanced daily menu options. Each daily menu on your list should
consist of {5} meals. Each daily menu dish on your list should not
contain products that may cause allergies: {user_info[allergies]} and
intolerances:{user_info[intolerances]}. User is currently located in
the country {Ukraine}. Please indicate prices for food products in
{UAH}».
```

Переклад на українську: «Ви кухар-дієтолог, ваша робота дуже важлива, тому виконуйте її відповідально і акуратно, не вигадуйте неіснуючі страви і дотримуйтеся дієтичних рекомендацій. Беручи до уваги дієтичні рекомендації {temp[diet_name]}, створіть для мене список {3} збалансованих варіантів щоденного меню. Кожне щоденне меню у вашому списку повинно складатися з {5} прийомів їжі. Кожна страва щоденного меню у вашому списку не повинна містити продуктів, які можуть викликати алергію: {user_info[allergies]} або непереносимість: {user_info[intolerances]}. Користувач зараз знаходиться в країні Україна. Ціни на продукти харчування просимо вказувати в гривнях.»

2.3 Критерій оцінювання ефективності підбору індивідуальної дієти

Для оцінювання ефективності розробленої інформаційної технології з підбору дієтичних меню необхідно провести експериментальні дослідження. Доцільним є застосування методу модельних досліджень, який дозволить максимально наблизити умови експерименту до реальних ситуацій використання системи. Ефективність створеної моделі інформаційної технології має бути підтверджена експериментально.

Розроблена інформаційна система для свторення індивідуальних дієтичних рекомендацій є достатньо складною для автоматичної перевірки оскільки на виході створюются дієтичні меню які мають бути перевірені на відповідність медичним рекомендаціям стосовно даних про здоров’я користувача, також меню не повинні містити жодних продуктів які можуть викликати додаткові проблеми зі здоров’ям користувача. З огляду на це, необхідно забезпечити ефективне мануальне тестування та оцінювання розробленоъ системи.

Методика експериментальних досліджень вимагає створення різнотипних вхідних запитів користувача, для перевірки ефективності розробленої інформаційної системи. Такий підхід дозволить протестувати систему в різноманітних сценаріях використання з варіативними вхідними параметрами.

Вхідні запити мають включати як коректні дані про стан здоров'я, харчові вподобання та обмеження користувача, так і некоректну, помилкову інформацію.

Це дасть змогу оцінити, як система реагує на некоректні вхідні дані та наскільки ефективно вона здатна їх ідентифікувати.

Для проведення оцінки ефективності розробленої інформаційної системи необхідно сформувати репрезентативну тестову вибірку вхідних даних. Існують два основні підходи до цього:

1. Ручне формування тестових даних. Цей варіант дозволяє створити набір вхідних запитів, в ідеалі написаних різними людьми, що максимально відображають різноманітні реальні сценарії використання системи. Однак, такий підхід є досить ресурсомістким та може забезпечити лише обмежену за обсягом тестову вибірку.

2. Автоматизоване формування тестових даних з використанням потужних LLM. Подібні моделі здатні генерувати великі обсяги синтетичних, але реалістичних тестових запитів, покриваючи широкий спектр можливих сценаріїв. Це дозволяє мінімізувати витрати ресурсів при забезпеченні необхідної репрезентативності тестової вибірки.

Таким чином, при формуванні тестових даних доцільно поєднувати ручне створення окремих узагальнених за певною особливістю кейсів за якими LLM зможе автоматизовано генерувати масиви синтетичних вхідних запитів. Такий комплексний підхід забезпечить повноту та різноманітність тестової вибірки, необхідної для об'єктивної оцінки ефективності розробленої інформаційної системи.

Типові вхідні дані для тестування розробленої інформаційної системи з підбору дієтичних рекомендацій можуть бути оформлені у вигляді текстових повідомлень, які в своєму контексті містять важливу для системи інформацію. Наприклад повідомлення «Привіт, у мене проблеми зі здоров'ям, зайва вага, проблеми з печінкою. У мене алергія на деревні горіхи. Хочу додати, що я не люблю лимон, але люблю банани. Також з минулого року помітив що у мене непереносимість лактози.» містить ключову інформацію про:

- проблеми із зайвою вагою та хворобою печінки;
- алергію на деревні горіхи;

– непереносимість лактози;

– харчові вподобання (не любить лимони, любить банани).

Відповідно до цієї інформації, інформаційна система має сформувати індивідуальне дієтичне меню, яке чітко враховуватиме зазначені обмеження та побажання.

Для всебічної перевірки ефективності розробленої системи потрібно провести значну кількість експериментів, але в межах дослідницької роботи та в умовах обмежених ресурсів дослідження, припустимо, що набору з 50 різнотипових тестових вхідних запитів буде достатньо.

Тестування має здійснюватися шляхом перевірки результату, який видає система на основі вхідного запиту на наявність некоректних даних. Зокрема, необхідно перевіряти:

1) Наявність у згенерованому дієтичному меню заборонених для користувача продуктів (алергени, продукти з непереносимістю або похідні від них).

2) Відповідність дієтичних рекомендацій наявній базі медичних показанням та обмеженням користувача.

Результати тестування мають бути внесені до таблиці 2.1. Це дозволить систематизувати дані та провести аналіз ефективності роботи розробленої інформаційної системи.

Таблиця 2.1 – Макет таблиці результатів експериментів

Номер експе-рименту	Вхідний запит	Коректно об-роблено пові-домлення	Вірно обрана медична дієта	Відповідність медичним реко-мендаціям	Відсутні за-боронені продукти у меню	TP/FP

Таблиця 2.1 передбачає фіксацію таких параметрів:

– номер експерименту;

– текст вхідного запиту користувача;

– наявність у згенерованому меню заборонених для користувача продуктів, може бути 1 (Так) або 0 (Ні);

– відповідність згенерованого меню відповідним до показанння користувача медичним рекомендаціям, може бути 1 (Так) або 0 (Ні);

– визначення, чи є результат True Positive (TP) чи False Positive (FP), якщо третій стовпець містить 0 і четвертий 1 то TP/FP = TP, інакше TP/FP = FP.

Оцінка ефективності та точності (Precision) моделі інформаційної технології може бути визначена через перевірку гіпотез щодо кількості правильно та неправильно сформованих дієтичних меню. Ці показники формують матрицю помилок (True Positive, False Positive).

Показник True Positive (TP) відображає випадки, коли система правильно визначила, що згенероване дієтичне меню є коректним та відповідає потребам користувача.

Показник False Positive (FP) відображає випадки, коли система помилилася, вважаючи, що згенероване дієтичне меню є коректним, тоді як насправді це не так.

Показник Precision відображає частку правильно згенерованих дієтичних рекомендацій серед усіх згенерованих меню, які система визначила як коректні. Іншими словами, Precision показує, наскільки часто система правильно створила коректні дієтичні меню. Розрахунок Precision здійснюється за формулою (2.1).

$$\text{Precision} = \frac{TruePositive}{TruePositive + FalsePositive} \qquad (2.1)$$

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ФОРМУВАННЯ ІНДИВІДУАЛЬНОЇ ДІЄТИ

3.1 Формування вхідних даних для алгоритму підбору індивідуальної дієти

Формування вхідних даних для алгоритму підбору індивідуальної дієти здійснюється шляхом складання спеціальних запитів-промптів до LLM системи ChatGPT. Вимоги до створення тестових вхідних даних включають:

1) Охоплення всіх типів медичних дієт, доступних у системі, відповідно до можливих показань користувача.

2) Включення вхідних даних, які містять коректну медичну інформацію, але для яких точної дієти немає в системі.

3) Наявність у тестовому наборі некоректних запитів для перевірки реакції системи, а саме:

- вигадані медичні терміни;
- медичні терміни з серйозними орфографічними помилками;
- запити, що містять нісенітницю або жарти.

4) Частина запитів має містити медичні терміни з незначними помилками.

5) Випадкове включення у вхідні запити додаткових уточнень щодо алергій, непереносимостей та особистих побажань користувача.

6) Структура запитів має бути максимально наближена до звичайних повідомлень, написаних природною мовою.

Для генерації коректних запитів використано декілька промптів.

Перший промпт пропонує моделі ChatGPT обрати випадковим чином хвороби з переліку медичних показань до наявних дієт та повернути результат у форматі Python коду. Приблизний вигляд промпту:

```
    «Choose one or two diseases from each line, return it like a
Python list. \n Indication List= {‘indications for diet 1':' acute
diseases of the upper parts of the alimentary canal',…}».
```

Модель повертає список захворювань, який в свою чергу входить до складу наступного промпту, який вже пропонує моделі сформувати повідомлення

користувачів у форматі природньої мови, які б містили ці хвороби. Приблизний вигляд промпту:

```
«Diseases_List = {‘indications for diet 1': ['esophagitis', 'gastritis'],…}».
For each line of diseases in the list, generate a message in the format of natural human language. The purpose of the messages should be as an appeal to the system with a request for help in creating a diet menu or simply reporting health information in order to create a diet. The format is free and maximum variety of message structure is encouraged. Diseases should be mentioned in the context of the message, as if the user were talking about himself. Return the resulting list of messages to Python list format.».
```

Результатом повертається згенерований список який містить тестові повідомлення користувачів. Але для більш реальної перевірки треба випадковим чином розширити ці повідомлення додаванням до них інформації про алергії та/або непереносимості користувачів. Тому створюється ще один промпт, який пропонує моделі контекстуально додати до наявних повідомлень інформацію про алергії та непереносимості. Приблизний вигляд промпту:

```
«I give you a list of messages: " messages = ["Hi there! I've been diagnosed with esophagitis and gastritis. Can you assist me in creating a suitable diet plan?",…]" in which contextually (as part of the message) and randomly selecting from the available ones you must add information about allergies and/or intolerances. List of existing allergies: " allergies = ["Peanuts", "Tree nuts",…]. List of existing intolerances: " food_intolerances = {"Dairy Products": ["milk",…],}.».
```

В результаті отримано список коректних повідомлень користувачів, який може бути використаний для експериментальної перевірки системи. Типовий приклад згенерованих коректних вхідних повідомлень:

```
«correct_messages = ["Hi there! I've been diagnosed with esophagitis and gastritis. Can you assist me in creating a suitable diet plan? I'm also allergic to peanuts and soy.",…]».
```

Для генерації частково коректних тестових вхідних запитів користувача також використано спеціальний промпт. Промпт пропонує моделі ChatGPT

переглянути вже існуючі коректні запити та додати до кожного незначні орфографічні помилки до медичних термінів, алергенів та іншої користувацької інформації. Пропонується повернути результат у форматі Python коду. Приблизний вигляд промпту:

```
«Hello, here are some messages from users intended for testing the system. Your task is to add minor typos to these messages: especially in medical terms or descriptions of allergies or food intolerances. The main thing is that these typos are very minor, for example, one or two letters are mixed up. Return the messages you have already changed in the form of Python array code. \n correct_messages = ["Hi there! I've been diagnosed with esophagitis and gastritis. Can you assist me in creating a suitable diet plan? I'm also allergic to peanuts and soy.",…]».
```

В результаті модель повертає список частково коректних повідомлень користувачів, які будуть використані для експериментальної перевірки реакції системи в подібній ситуації. Приклад згенерованих частково коректних вхідних повідомлень:

```
«partially_correct_messages = ["Hi there! I've been diagnosed with esophagytis and gastritis. Can you assist me in creating a suitable diet plan? I'm allergic to peanits (коректний варіант: peanuts) and soy.",…]»
```

Для генерації повністю некоректних тестових вхідних запитів користувача використано наступний спеціальний промпт. Промпт пропонує моделі ChatGPT переглянути вже існуючі коректні запити, та зробити їх некоректними. Моделі пропонується створити повністю некоректні повідомлення які будуть містити: вигадані медичні терміни, медичні терміни з серйозними орфографічними помилками, повідомлення з нісенітницею або жартами. Також моделі пропонується повернути результат форматований як Python код. Приблизний вигляд промпту:

```
«Hello, here are some messages from users intended for testing the system. Your task is to create completely incorrect messages that will contain: invented medical terms, medical terms with serious
```

```
spelling errors, messages that contain nonsense or jokes. The main
thing is that these errors are very significant, for testing the
system failure mechanism. Return the messages you have already changed
in the form of Python array code. \n correct_messages = ["Hi there!
I've been diagnosed with esophagitis and gastritis. Can you assist me
in creating a suitable diet plan? I'm also allergic to peanuts and
soy.",…]»
```

Модель повертає результат у вигляді згенерованого списку некоректних повідомлень користувачів, які будуть використані для експериментальної перевірки реакції системи в подібній критичній ситуації. Система не має обробляти подібні запити. Приклад згенерованих некоректних вхідних повідомлень:

```
    «incorrect_messages = ["Hiya, I've been diagnosed with
flibberjabberitis and gobbledygookitis (не є реальними захворюваннями:
flibberjabberitis, gobbledygookitis). Can you help me create a diet
plan? I'm allergic to marshmallows and moon cheese (жарт: moon
cheese).",…]».
```

3.2 Короткий опис програмного забезпечення

На вхід системи інформаційної технології для створення дієтичних меню необхідно подавати окрім вхідного повідомлення користувача ще й дані IP адреси для визначення приблизної локації перебування. Докладно алгоритм роботи інформаційної системи описано в пункті 2.1.

Реалізація даної інформаційної системи потребує використання бібліотек зворотнього зв’язку до API LLM, токенізації тексту, а також бібліотек для роботи з промптами. Для простої веб реалізації графічного інтерфейсу системи пропонується використання бібліотеки Streamlit.

Запропонований варіант графічного веб-інтерфейсу системи наведено на рисунку 3.1. Також користувач може отримати меню як PDF файл.

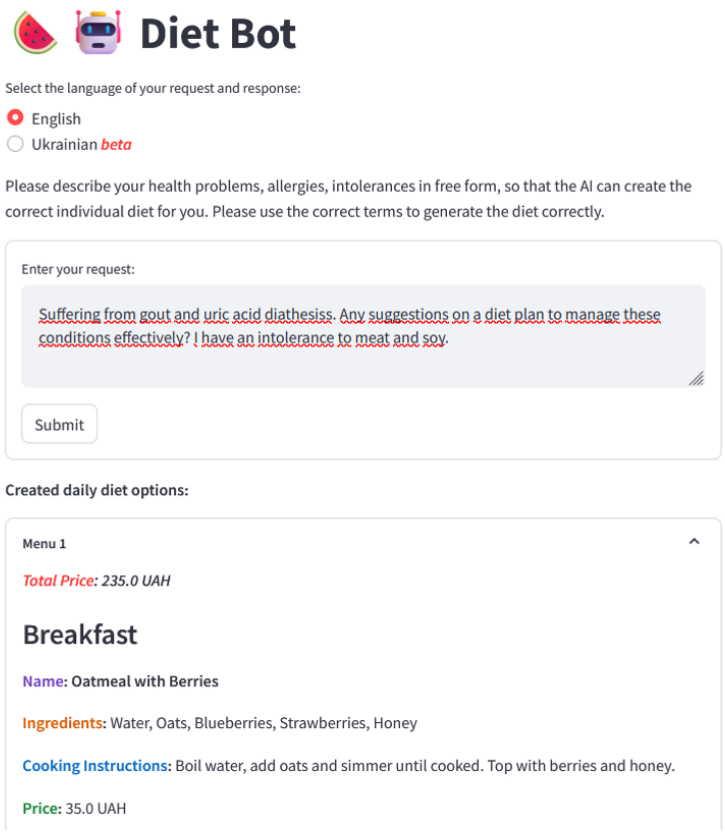

Рисунок 3.1 – Варіант графічного інтерфейсу системи

Для експериментів пропонується перевірити як система у різних сценаріях обробляє тестові клієнтські запити. Для визначення ефективності запропонованого підходу варто дослідити якість створених системою дієтичних меню та порівняти з результатами досліджених існуючих систем генерації дієт за допомогою LLM.

Розробка програмного коду здійснювалась в середовищі Jupyter Notebook, Google colab, JetBrains PyCharm на мові програмування python 3. Розроблене програмне забезпечення потребує використання сторонніх бібліотек та фреймворку, опис яких наведено в таблиці 3.1.

Вхідні дані надходять до системи і з використанням бібліотек pydantic та langchain утворюють промпти для LLM. Для простоти написання коду роботи з API LLM пропонується використовувати технологію chains фреймворку LangChain.

Таблиця 3.1 – Використані бібліотеки

Назва бібліотеки	Опис
LangChain	Це бібліотека, що спеціалізується на обробці природної мови (NLP) та машинному навчанні. Вона включає в себе набір інструментів для обробки тексту, перекладу, генерації тексту, а також для аналізу та взаємодії з мовними моделями (LLM). Включає ряд пакетів: – Інструменти промптінгу (langchain.prompts); – Парсери відповідей LLM (langchain_core.output_parsers); – Контроль помилок (langchain.schema).
reportlab	ReportLab є потужною бібліотекою для створення PDF документів у Python. Вона дозволяє розробникам генерувати документи з різними форматами, включаючи текст, зображення, таблиці, графіки та інше. ReportLab підтримує широкий спектр форматів PDF, та ін.
openai	OpenAI є бібліотекою для взаємодії з API від OpenAI, що включає в себе моделі машинного навчання для генерації тексту, відповіді на питання, перекладу тексту та інше. Ця бібліотека дозволяє розробникам легко інтегрувати можливості штучного інтелекту в свої додатки.
streamlit	Streamlit є бібліотекою для швидкого створення інтерактивних веб-додатків для даних. Вона дозволяє розробникам легко візуалізувати дані та інтерактивно взаємодіяти з ними через веб-інтерфейс. Streamlit підтримує різні типи даних та візуалізацій, включаючи графіки, таблиці, мапи та інше.
pydantic	Pydantic є бібліотекою для валідації даних та створення моделей у Python. Вона дозволяє легко перетворювати дані в об'єкти Python з аннотаціями типів, що покращує структуру коду та допомагає у виявленні помилок. Дозволяє створювати типізовані шаблони для генерації відповідей LLM у форматі JSON.
typing	Модуль Typing в Python дозволяє використовувати аннотації типів для статичної типізації. Разом з Pydantic допомагає контролювати генерацію типізованих відповідей LLM.

Продовження таблиці 3.1.

Назва бібліотеки	Опис
tiktoken	Tiktoken є бібліотекою для підрахунку токенів у тексті, що корисно для роботи з API, які обмежують кількість токенів у запитах. Це дозволяє контролювати використання ресурсів API LLM та оптимізувати використання.
googletrans	Googletrans є бібліотекою для перекладу тексту за допомогою Google Translate API. Вона дозволяє легко перекладати текст між різними мовами, використовуючи можливості Google Translate.

Програмний код наведено в додатку А. Призначення основних функцій програми приведено в таблиці 3.2.

Таблиця 3.2 – Основні функції програмного забезпечення

Функція	Короткий опис
get_model	Функція ініціалізує та повертає об'єкт chatopenai, використовуючи наданий ключ API та модель.
get_user_info	Функція витягує інформацію користувача, що стосується проблем зі здоров'ям, алергій, непереносимості, дієтичних вподобань та інших обмежень з користувацького повідомлення за допомогою моделі chatopenai. Вона повертає об'єкт типу userinfo, що містить витягнуту інформацію.
get_indication_info	Функція витягує медичні показання на основі інформації про здоров'я користувача та списку показань. Вона повертає об'єкт indication, що містить індекс збігу та пояснення.
get_diet_options	Функція генерує список збалансованих щоденних меню з урахуванням дієтичних рекомендацій та вподобань користувача. Вона виключає інгредієнти, які можуть викликати алергії або непереносимість, зазначені користувачем. Вона повертає об'єкт dailymenulist, що містить згенеровані варіанти меню.

Продовження таблиці 3.2.

Функція	Короткий опис
translate_dict, translate_text	Функції рекурсивно перекладають текст за допомогою API Google Translate. Переклад потрібен для економії токенів.
main	Головна функція, яка керує всім процесом. Спочатку вона витягує інформацію користувача за допомогою get_user_info, отримує показання за допомогою get_indication_info, генерує варіанти дієтичного меню за допомогою get_diet_options, і, за необхідності, перекладає вихідні дані на українську мову в залежності від вказаної мови. На останок, вона повертає згенеровані або перекладені варіанти меню.

Таким чином, програмна реалізація дозволяє протестувати розроблений алгоритм. Простий графічний веб-інтерфейс забезпечує зручне тестування програми та є початковим варіантом кінцевого вигляду програмного продукту. В подальшому, розроблена програмна реалізація може бути розвинена до повнофункціональної системи формування персоналізованих дієтичних меню.

3.3 Результати експериментів

Проведемо експерименти застосовуючи створені набори тестових запитів. Першими опрацюємо коректні запити, результати занесемо до таблиці 3.3.

Таблиця 3.3 – Результати експериментів для коректних даних

№	**Вхідний запит**	**Коректно оброблено повідомлення**	**Вірно обрана медична дієта**	**Відповідність медичним рекомендаціям**	**Відсутність заборонених продуктів в меню**	**TP/FP**
1	"Hi there! I've been...	1	1	1	1	TP
2	"Dealing with peptic...	1	0,5	1	1	TP
3	"I've been managing ...	1	0,5	1	1	TP
4	"Struggling with acu...	1	0,5	1	1	TP
5	"I've been diagnosed...	1	1	0,5	1	TP
6	"Dealing with acute ...	1	1	1	1	TP

Продовження таблиці 3.3.

№	Вхідний запит	Коректно оброблено повідомлення	Вірно обрана медична дієта	Відповідність медичним рекомендаціям	Відсутність заборонених продуктів в меню	TP/ FP
7	"I'm coping with chr...	1	0,5	1	1	TP
8	"Recently diagnosed ...	1	1	1	0,5	TP
9	"I'm facing acute ch...	1	1	1	1	TP
10	"Suffering from gout...	1	1	0	0	FP
11	"Dealing with acute ...	1	1	1	1	TP
12	"I've been diagnosed...	1	1	1	1	TP
13	"Managing diabetes a...	1	1	1	1	TP
14	"Struggling with car...	1	1	1	1	TP
15	"Recovering from hea...	1	1	1	0,5	TP
16	"Facing atherosclero...	1	1	1	1	TP
17	"Diagnosed with tube...	1	1	1	1	TP
18	"Dealing with nervou...	1	1	0	1	FP
19	"Currently battling ...	1	1	1	1	TP
20	"Struggling with pho...	1	1	1	1	TP
21	"In the recovery per...	1	1	1	1	TP
22	"Just underwent surg...	1	1	1	1	TP

На етапі проектування системи було вирішено використовувати модель GPT-3.5 для обробки вхідних повідомлень користувачів, перевірки їх коректності та вибору дієтичних рекомендацій на основі вихідних показників, тоді як модель GPT-4 передбачалося застосовувати лише для генерації дієтичних меню. Однак, у процесі експериментальних досліджень було виявлено, що точність вибору дієтичних рекомендацій за допомогою моделі GPT-3.5 була незадовільно низькою. У зв'язку з цим було прийнято рішення задіяти потужнішу модель GPT-4 також і для виконання цього завдання, та продовжити тестування вже з нею.

Для коректних даних кількість TP = 20, FP = 2, Precision = 0,909 (табл. 3.3). Позначкою «1» позначено повністю коректні результати системи. «0,5» позначено відповіді, які також можна вважати коректними, однак вони містять незначні неточності які не впливають на загальну якість результатів. Такі результати частково отримані в місцях де дієтичні рекомендації були дуже схожими або взаємозамінними. Також на оцінку вплинули не зовсім коректно сформовані

вхідні дані, наприклад дуже загальні алергени, або дуже непоширені харчові непереносимості. Позначкою «0» позначено повністю некоректні відповіді.

Далі перевіримо реакцію системи на частково некоректні запити, результати занесемо до таблиці 3.4.

Таблиця 3.4 – Результати експериментів для частково коректних даних

№	Вхідний запит	Коректно оброблено повідомлення	Вірно обрана медична дієта	Відповідність медичним рекомендаціям	Відсутність заборонених продуктів в меню	TP/FP
23	"Hi there! I've been...	1	1	1	1	TP
24	"Dealing with peptic...	1	0,5	1	1	TP
25	"I've been managing ...	1	1	1	1	TP
26	"Struggling with acu...	1	0,5	1	1	TP
27	"I've been diagnosed...	1	1	1	1	TP
28	"Dealing with acut e...	1	1	1	1	TP
29	"I'm coping with chr...	1	0,5	1	1	TP
30	"Recently diagnosed ...	1	1	1	0,5	TP
31	"I'm facing acut cho...	1	1	1	1	TP
32	"Suffering from gout...	1	1	0	0	FP
33	"Dealing with acut a...	1	1	1	1	TP
34	"I've been diagnosed...	1	0,5	1	1	TP
35	"Managing diabetes a...	1	1	1	1	TP
36	"Struggling with car...	1	1	0,5	1	TP

Для частково коректних даних кількість TP = 13, FP = 1, Precision = 0,928. Позначкою «1» позначено повністю коректні результати системи, в більшості співпадають з попередніми результатами, тому що інформаційна система виправляє деякі неточності або вони є незначними для її роботи. Аналогічно, «0,5» позначено відповіді, які можна вважати коректними, але вони містять незначні неточності які не впливають на загальну якість результатів. Оскільки дані для цього експерименту є пошкодженими даними для першого, то результати є дуже схожими. Позначкою «0» позначено повністю некоректні відповіді, відповідають першому експерименту.

Також перевіримо як відповідає система на некоректні запити, результати занесемо до таблиці 3.5.

Таблиця 3.5 – Результати експериментів для некоректних даних

№	Вхідний запит	Коректно оброблено повідомлення	Вірно обрана медична дієта	Відповідність медичним рекомендаціям	Відсутність заборонених продуктів в меню	TP/ FP
37	"Hiya, I've been dia...	1	*	*	*	TP
38	"I'm having trouble ...	0,5	*	*	*	TP
39	"I've been juggling ...	1	*	*	*	TP
40	"Struggling with acu...	1	*	*	*	TP
41	"I've been diagnosed...	0	*	*	*	FP
42	"Dealing with acute ...	1	*	*	*	TP
43	"I'm coping with chr...	0,5	*	*	*	TP
44	"Recently diagnosed ...	0,5	*	*	*	TP
45	"I'm facing acute ch...	1	*	*	*	TP
46	"Suffering from goat...	0	*	*	*	FP
47	"Dealing with acute ...	0	*	*	*	FP
48	"I've been diagnosed...	0,5	*	*	*	TP
49	"Managing diabetes a...	0,5	*	*	*	TP
50	"Struggling with car...	0,5	*	*	*	TP

Для некоректних даних кількість TP = 11, FP = 3, Precision = 0,786. Для оцінки даного типу запитів достатньо перевірки чи прийме система некоректні дані за коректні. Як «1» позначаються відхилені запити. «0,5» вирішено позначити запити які система, попри їх некоректність, виправила і відповіла на них, а також запити з яких система взяла тільки коректну інформації прибравши все зайве. В цьому експерименті позначок «0» більше, оскільки система іноді змішує коректні дані з некоректними та видає їх за істину.

Загальний підрахунок точності (Precision) системи наведено в таблиці 3.6. Слід зазначити, що показник відсутності заборонених продуктів у згенерованій дієті залежить від того, які дані були отримані системою. Система має проблему з недопущенням складних алергенів, таких як хімічні елементи, непоширені алергени та подразники, а також загальні категорії алергічних продуктів, що потребують уточнення індивідуальним діагнозом.

Таблиця 3.6 – Загальна точність системи

TP	**44**
FP	**6**
Precision	**0,88**

Порівняно з попередніми розробками систем на основі великих мовних моделей для створення дієтичних меню з урахуванням алергенів та харчових непереносимостей [26,27], розроблена система дозволяє обробляти вхідні запити користувача у форматі природної мови та застосовує техніки prompt engineering для додаткової перевірки наявності некоректних даних при генерації дієтичних меню на основі медичних рекомендацій. Розроблена система демонструє задовільний показник точності (Precision) генерації, для предметної області — генерації медичних дієт з використанням експертних знань про медичні дієти. Подібно до інших систем, вона все ще стикається з проблемами точності генерації дієт, оскільки це пов'язано з загальними недоліками точності моделей LLM. Тому застосування розробленої системи інтелектуального аналізу даних для створення індивідуальних меню, базованих на експертній базі медичних рекомендацій, наразі доцільне лише як допоміжного рекомендаційного інструменту, де остаточне рішення приймається користувачем або експертом з огляду на можливість наявності помилок у згенерованих відповідях.

Під час експериментальної перевірки системи було виявлено, що для задач, які вимагають точних відповідей, зокрема пов'язаних з медициною, рекомендаціями щодо харчування тощо, найбільш ефективними є найпотужніші варіанти великих мовних моделей. У рамках дослідження було використано дві моделі: менш потужну та більш дешеву GPT-3.5, а також більш потужну та дорожчу GPT-4. Результати продемонстрували, що найякісніші відповіді забезпечила модель GPT-4. Отже, для побудови систем, призначених для побудови дієтичних меню, пропонується використовувати потужнішу модель GPT-4 із застосуванням методу prompt engineering з акцентом, у промптах, на самоперевірку моделі.

ВИСНОВКИ

Проведено аналітичний огляд сучасного стану та тенденцій розвитку інформаційних технологій підбору дієти, і в результаті встановлено, що питання розроблення нової, більш ефективної за попередні, моделі з можливістю вводу вхідних обмежень у форматі природньої мови для системи створення індивідуальних дієт є невирішеним та актуальним.

Модель, що пропонується, враховує недоліки prompt engineering попередніх досліджень, використовує спеціально спроектовані шаблони prompts та застосовує методики prompt engineering, що дозволяє ефективно використовувати моделі LLM та отримати прийнятну Precision (точність) для поставленої задачі.

Результати експериментальної перевірки підтверджують працездатність розробленого алгоритму, проте цілком безпомилкових результатів не було отримано, що може свідчити про неоптимальність створених промптів та загальні недоліки великих мовних моделей (LLM).

Розроблена програмна реалізація інформаційної технології інтелектуального аналізу даних для формування індивідуальної дієти. Побудована система обробляє запити користувачів, подані у вигляді природньої мови. Запити містять інформацію про обмеження та побажання користвача щодо персоналізації дієти. Система враховує обмеження, а саме медичні рекомендації, протипоказання та особисті побажання користувача.

Результати роботи можуть бути використані в якості експериментального онлайн сервісу створення індивідуальних дієт. Також, за умови подальшої розробки, робота може бути цікава для застосування у практичній медицині як допоміжний засіб для створення дієтичних меню.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Jayalakshmi Ms.P. et al. Food Recommendation System using One-Stage Algorithm // Int J Res Appl Sci Eng Technol. International Journal for Research in Applied Science and Engineering Technology (IJRASET), 2023. Vol. 11, № 5. P. 2931–2942.
2. Fadhil A. Can a Chatbot Determine My Diet?: Addressing Challenges of Chatbot Application for Meal Recommendation. 2018.
3. Islam T. et al. Human Behavior-based Personalized Meal Recommendation and Menu Planning Social System. 2023.
4. Taneja K., Segal R., Goodwin R. Monte Carlo Tree Search for Recipe Generation using GPT-2. 2024.
5. Roy M., Das S., Protity A.T. OBESEYE: Interpretable Diet Recommender for Obesity Management using Machine Learning and Explainable AI. 2023.
6. Moz S.H. et al. Precision cardiodiet: transforming cardiac care with artificial intelligence-driven dietary recommendations // Radioelectronic and Computer Systems. 2023. Vol. 0, № 4. P. 20–31.
7. Lambay M.A., Mohideen S.P. A Hybrid Approach Based Diet Recommendation System using ML and Big Data Analytics. 2022.
8. (1) (PDF) Diet Recommendation System Using Machine Learning [Electronic resource]. URL: https://www.researchgate.net/publication/376517685_Diet_Recommendation_System_Using_Machine_Learning?channel=doi&linkId=657b9ff3ea5f7f020570b2a9&showFulltext=true (accessed: 15.02.2024).
9. Salinari A. et al. The Application of Digital Technologies and Artificial Intelligence in Healthcare: An Overview on Nutrition Assessment // Diseases 2023, Vol. 11, Page 97. Multidisciplinary Digital Publishing Institute, 2023. Vol. 11, № 3. P. 97.
10. Sookrah R., Dhowtal J.D., Devi Nagowah S. A DASH Diet Recommendation System for Hypertensive Patients Using Machine Learning // 2019 7th

International Conference on Information and Communication Technology (ICoICT). IEEE, 2019. P. 1–6.

11. Balloccu S. et al. Ask the experts: sourcing high-quality datasets for nutritional counselling through Human-AI collaboration. 2024.
12. Turner-McGrievy G.M. et al. Crowdsourcing for self-monitoring: Using the Traffic Light Diet and crowdsourcing to provide dietary feedback // Digit Health. SAGE Publications, 2016. Vol. 2. P. 205520761665721.
13. Hosio S.J. et al. Crowdsourcing Personalized Weight Loss Diets // Computer (Long Beach Calif). IEEE Computer Society, 2020. Vol. 53, № 1. P. 63–71.
14. Silva V.C. et al. Recommender System Based on Collaborative Filtering for Personalized Dietary Advice: A Cross-Sectional Analysis of the ELSA-Brasil Study // Int J Environ Res Public Health. MDPI, 2022. Vol. 19, № 22. P. 14934.
15. Jiang L. et al. DeepFood: Food Image Analysis and Dietary Assessment via Deep Model // IEEE Access. Institute of Electrical and Electronics Engineers Inc., 2020. Vol. 8. P. 47477–47489.
16. Tahir G.A., Loo C.K. A Comprehensive Survey of Image-Based Food Recognition and Volume Estimation Methods for Dietary Assessment. // Healthcare (Basel). Multidisciplinary Digital Publishing Institute (MDPI), 2021. Vol. 9, № 12. P. 1676.
17. Dalakleidi K. V et al. Applying Image-Based Food-Recognition Systems on Dietary Assessment: A Systematic Review. // Adv Nutr. Elsevier, 2022. Vol. 13, № 6. P. 2590–2619.
18. Marrero A., Segredo E., Leon C. On the automatic planning of healthy and balanced menus // Proceedings of the Genetic and Evolutionary Computation Conference Companion. New York, NY, USA: ACM, 2019. P. 71–72.
19. Hernandez-Ocana B. et al. Bacterial Foraging Optimization Algorithm for Menu Planning // IEEE Access. Institute of Electrical and Electronics Engineers Inc., 2018. Vol. 6. P. 8619–8629.

20. Ahmed I.M., Mahmoud A.M. Development of an Expert System for Diabetic Type-2 Diet // Int J Comput Appl. Foundation of Computer Science, 2020. Vol. 107, № 1. P. 13–21.
21. Kovásznai G. Developing an expert system for diet recommendation // SACI 2011 - 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings. 2011. P. 505–509.
22. Aditya N., Baizal Z.K.A., Dharayani R. Healthy Food Recommender System for Obesity Using Ontology and Semantic Web Rule Language // Building of Informatics, Technology and Science (BITS). Forum Kerjasama Pendidikan Tinggi (FKPT), 2023. Vol. 4, № 4.
23. S* S., K A. Ontology-Based Diet Recommendation System // International Journal of Recent Technology and Engineering (IJRTE). Blue Eyes Intelligence Engineering and Sciences Publication, 2019. Vol. 8, № 3. P. 5811–5815.
24. (1) (PDF) DFRS: Diet food recommendation system for diabetic patients based on ontology [Electronic resource]. URL: https://www.researchgate.net/publication/283690541_DFRS_Diet_food_recommendation_system_for_diabetic_patients_based_on_ontology (accessed: 15.02.2024).
25. Chen R.-C., Huang C.-Y., Ting Y.-H. A Chronic Disease Diet Recommendation System Based on Domain Ontology and Decision Tree // Journal of Advanced Computational Intelligence and Intelligent Informatics. Fuji Technology Press Ltd., 2017. Vol. 21, № 3. P. 474–482.
26. Papastratis I. et al. Can ChatGPT provide appropriate meal plans for NCD patients? // Nutrition. Elsevier, 2024. Vol. 121. P. 112291.
27. Niszczota P., Rybicka I. The credibility of dietary advice formulated by ChatGPT: Robo-diets for people with food allergies. // Nutrition. Elsevier, 2023. Vol. 112. P. 112076.
28. What is natural language processing? | Definition from TechTarget [Electronic resource]. URL: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP (accessed: 27.03.2024).

29. Aman’s AI Journal • Natural Language Processing • NLP Tasks [Electronic resource]. URL: https://aman.ai/primers/ai/nlp-tasks/ (accessed: 27.03.2024).
30. From Rule-Based Systems to Transformers: A Journey through the Evolution of Natural Language Processing | by Seyed Saeid Masoumzadeh, PhD | Medium [Electronic resource]. URL: https://medium.com/@masoumzadeh/from-rule-based-systems-to-transformers-a-journey-through-the-evolution-of-natural-language-9131915e06e1 (accessed: 27.03.2024).
31. Vaswani A. et al. Attention Is All You Need // Adv Neural Inf Process Syst. Neural information processing systems foundation, 2017. Vol. 2017-December. P. 5999–6009.
32. Rahali A., Akhloufi M.A. End-to-End Transformer-Based Models in Textual-Based NLP // AI 2023, Vol. 4, Pages 54-110. Multidisciplinary Digital Publishing Institute, 2023. Vol. 4, № 1. P. 54–110.
33. Large Language Model (LLM): GPT ChatGPT BERT XLNet T5 RoBERTa — Computing for All [Electronic resource]. URL: https://computing4all.com/large-language-model-llm/#Strengths_and_Limitations_of_LLMs (accessed: 28.03.2024).
34. Kalyan K.S. A survey of GPT-3 family large language models including ChatGPT and GPT-4 // Natural Language Processing Journal. Elsevier, 2024. Vol. 6. P. 100048.
35. Introducing Gemini: Google’s most capable AI model yet [Electronic resource]. URL: https://blog.google/technology/ai/google-gemini-ai/#availability (accessed: 28.03.2024).
36. The best large language models (LLMs) in 2024 [Electronic resource]. URL: https://zapier.com/blog/best-llm/ (accessed: 28.03.2024).
37. What is Llama 2 and why does it matter? [Electronic resource]. URL: https://zapier.com/blog/llama-meta/ (accessed: 28.03.2024).
38. Lin G., Zhang Y. Sparks of Artificial General Recommender (AGR): Experiments with ChatGPT // Algorithms. Multidisciplinary Digital Publishing Institute (MDPI), 2023. Vol. 16, № 9. P. 432.

39. Lai J. et al. Large Language Models in Law: A Survey. 2023.
40. Wang H., Na T. Rethinking E-Commerce Search. 2023.
41. Liao J. et al. LLaRA: Aligning Large Language Models with Sequential Recommenders. 2023.
42. Li X. et al. E4SRec: An Elegant Effective Efficient Extensible Solution of Large Language Models for Sequential Recommendation. ACM, 2023. P. 11.
43. Huang C.-Y. et al. Inspo: Writing Stories with a Flock of AIs and Humans. 2023.
44. Qiu J. et al. ControlRec: Bridging the Semantic Gap between Language Model and Personalized Recommendation // Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX). 2023. Vol. 1.
45. Meskó B., Topol E.J. The imperative for regulatory oversight of large language models (or generative AI) in healthcare // NPJ Digit Med. Nature Research, 2023. Vol. 6, № 1.
46. Sorin V. et al. Large language model (ChatGPT) as a support tool for breast tumor board // NPJ Breast Cancer. Nature Research, 2023. Vol. 9, № 1.
47. Demszky D. et al. Using large language models in psychology // Nature Reviews Psychology. Nature Publishing Group, 2023. Vol. 2, № 11. P. 688–701.
48. Gong Y. et al. An Unified Search and Recommendation Foundation Model for Cold-Start Scenario. Association for Computing Machinery (ACM), 2023. P. 4595–4601.
49. Wong G.K.W., Li S.Y.K. An Exploratory Study of Helping Undergraduate Students Solve Literature Review Problems Using Litstudy and NLP // Educ Sci (Basel). Multidisciplinary Digital Publishing Institute (MDPI), 2023. Vol. 13, № 10.
50. Yin B. et al. Heterogeneous Knowledge Fusion: A Novel Approach for Personalized Recommendation via LLM // Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023. Association for Computing Machinery, Inc, 2023. P. 599–601.

51. Mysore S., McCallum A., Zamani H. Large Language Model Augmented Narrative Driven Recommendations // Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023. Association for Computing Machinery, Inc, 2023. P. 777–783.
52. Harte J. et al. Leveraging Large Language Models for Sequential Recommendation // Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023. Association for Computing Machinery, Inc, 2023. P. 1096–1102.
53. Kaebnick G.E. et al. Editors' Statement on the Responsible Use of Generative AI Technologies in Scholarly Journal Publishing // Ethics Hum Res. John Wiley and Sons Inc, 2023. Vol. 45, № 5. P. 39–43.
54. Liao L., Yang G.H., Shah C. Proactive Conversational Agents in the Post-ChatGPT World // SIGIR 2023 - Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, Inc, 2023. P. 3452–3455.
55. Fraser H. et al. Comparison of Diagnostic and Triage Accuracy of Ada Health and WebMD Symptom Checkers, ChatGPT, and Physicians for Patients in an Emergency Department: Clinical Data Analysis Study // JMIR Mhealth Uhealth. JMIR Publications Inc., 2023. Vol. 11, № 1.
56. Bernabei M. et al. Students' use of large language models in engineering education: A case study on technology acceptance, perceptions, efficacy, and detection chances // Computers and Education: Artificial Intelligence. Elsevier B.V., 2023. Vol. 5.
57. Yamazaki T. et al. Building a hospitable and reliable dialogue system for android robots: a scenario-based approach with large language models // Advanced Robotics. Robotics Society of Japan, 2023. Vol. 37, № 21. P. 1364–1381.
58. Varkevisser R.D.M. et al. Cardiovascular risk management in people with type 1 diabetes: Performance using three guidelines // BMJ Open Diabetes Res Care. BMJ Publishing Group, 2022. Vol. 10, № 4.

59. Hou Y. et al. Large Language Models are Zero-Shot Rankers for Recommender Systems. 2023.
60. Fan W. et al. Recommender Systems in the Era of Large Language Models (LLMs) // IEEE Trans Knowl Data Eng. 2023. P. 1.
61. Naveed H. et al. A Comprehensive Overview of Large Language Models. 2023.
62. Singh I. et al. ProgPrompt: program generation for situated robot task planning using large language models // Auton Robots. Springer, 2023. Vol. 47, № 8. P. 999–1012.
63. Arora S. et al. Ask Me Anything: A simple strategy for prompting language models. 2022.
64. Rooein D., Curry A.C., Hovy D. Know Your Audience: Do LLMs Adapt to Different Age and Education Levels? 2023.
65. Ma X. et al. Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses. 2023. Vol. 1.
66. Li L., Zhang Y., Chen L. Prompt Distillation for Efficient LLM-based Recommendation. Association for Computing Machinery (ACM), 2023. P. 1348–1357.
67. Bao K. et al. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation // Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023. Association for Computing Machinery, Inc, 2023. P. 1007–1014.
68. de Zarzà I. et al. Emergent Cooperation and Strategy Adaptation in Multi-Agent Systems: An Extended Coevolutionary Theory with LLMs // Electronics (Switzerland). MDPI, 2023. Vol. 12, № 12.
69. Meskó B. Prompt Engineering as an Important Emerging Skill for Medical Professionals: Tutorial // J Med Internet Res. JMIR Publications Inc., 2023. Vol. 25, № 1.
70. Decker H. et al. Large Language Model-Based Chatbot vs Surgeon-Generated Informed Consent Documentation for Common Procedures // JAMA Netw Open. 2023. Vol. 6, № 10. P. e2336997.

71. Zhang J. et al. Is ChatGPT Fair for Recommendation? Evaluating Fairness in Large Language Model Recommendation // Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023. Association for Computing Machinery, Inc, 2023. P. 993–999.
72. Lim B. et al. Evaluating the efficacy of major language models in providing guidance for hand trauma nerve laceration patients: a case study on Google's AI BARD, Bing AI, and ChatGPT // Plast Aesthet Res. OAE Publishing Inc., 2023. Vol. 10.
73. Papastratis I. et al. Can ChatGPT provide appropriate meal plans for NCD patients? // Nutrition. Elsevier, 2023. P. 112291.
74. Building a Food Recommendation System | by Luís Rita | Towards Data Science [Electronic resource]. URL: https://towardsdatascience.com/building-a-food-recommendation-system-90788f78691a (accessed: 04.12.2023).
75. Pochmann V.O., Von Zuben F.J. Multi-Objective Bilevel Recommender System for Food Diets // 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2022. P. 1–8.
76. Pevzner Diets [Electronic resource]. URL: https://w.wiki/A49r (accessed: 22.04.2024).
77. Emotional prompts enhance language models, study finds - TechTalks [Electronic resource]. URL: https://bdtechtalks.com/2023/11/06/llm-emotion-prompting/ (accessed: 05.04.2024).

ДОДАТОК А

main.py

```
import streamlit as st
from openai import AuthenticationError
from functions import main
from pdf_generator import generate_pdf
import time

# Title and language selection
st.title('🍉🤖 Diet Bot')
language = st.radio("Select the language of your request and response:", ["English", "Ukrainian ***:red[beta]***"])

# Sidebar for OpenAI API key input
openai_api_key = st.sidebar.text_input('OpenAI API Key', type='password')

# Initialization result
if 'result' not in st.session_state:
    st.session_state['result'] = ''


# Function to generate response
def generate_response(input_text, api_key, language):
    with st.spinner('Wait for it...'):
        st.session_state['result'] = main(input_text, api_key, language)


# Main content
st.write(
    'Please describe your health problems, allergies, intolerances '
```

```
        'in free form, so that the AI can create the correct
individual diet for you. '
        'Please use the correct terms to generate the diet
correctly.'
    )

    # Form for user input
    with st.form('my_form'):
        text = st.text_area('Enter your request:',
                            placeholder='Example: Hello, I have a
cold, please help me create a dietary menu for me. Please note that I
am allergic to tree nuts and peanuts. I also have lactose
intolerance.')
        submitted = st.form_submit_button('Submit')

        # Warning for missing OpenAI API key
        if not openai_api_key.startswith('sk-'):
            st.warning('Please enter your OpenAI API key!', icon='⚠')

        # Process submitted form
        if submitted and openai_api_key.startswith('sk-'):
            if text:
                try:
                    generate_response(text, openai_api_key, language)
                except AuthenticationError:
                    st.error('Please enter correct OpenAI API key!',
icon='⚠')
            else:
                st.info('Please enter your request.', icon='⚠')

    # Display result if available
    if st.session_state['result']:
        st.write("**Created daily diet options:**")
        result = st.session_state['result']
```

```
        for i, menu in enumerate(result['menus']):
            # Create tab for each menu option
            with st.expander(f"**Menu {i + 1}**"):
                st.write(f"***:red[Total Price]:
{menu['total_price']} {menu['currency']}***")
                for meal in menu['meals']:
                    st.subheader(f"{meal['description']}")
                    for dish in meal['dishes']:
                        st.write(f"**:violet[Name]:
{dish['name']}**")
                        st.write(f"**:orange[Ingredients]:** {',
'.join(dish['ingredients'])}")
                        st.write(f"**:blue[Cooking Instructions]:**
{dish['cooking_instructions']}")
                        st.write(f"**:green[Price]:** {dish['price']}
{menu['currency']}")
                    # Generate PDF button
                if st.button(f"Generate PDF for Menu {i + 1}",
key=f"generate-pdf-menu_{i + 1}"):
                    with st.spinner('Generating PDF...'):
                        time.sleep(1)
                        generate_pdf(menu, f"diet_plan_{i + 1}.pdf")
                    st.success(f"Generated diet_plan_{i + 1}.pdf!")
                    with open(f"diet_plan_{i + 1}.pdf", "rb") as f:
                        st.download_button(f"Download pdf for Menu {i
+ 1}", f, f"diet_plan_{i + 1}.pdf",
                                           key=f"download-pdf-menu_{i
+ 1}")
```

functions.py

```
from typing import List, Dict, Optional
from langchain.prompts import PromptTemplate
from langchain_core.output_parsers import JsonOutputParser
from langchain_core.pydantic_v1 import BaseModel, Field
```

```
    from langchain_openai import ChatOpenAI
    from langchain.output_parsers import OutputFixingParser
    from langchain.schema import OutputParserException
    import googletrans
    import diets
    import streamlit as st


    # Define your models
    class UserInfo(BaseModel):
        health_problems: str = Field(
            description="If included in the message: User's health
problems and any information from the message.")
        allergies: str = Field(description="If included in the
message: User's allergies.")
        intolerances: str = Field(description="If included in the
message: User's food intolerances.")
        exclude: str = Field(description="If included in the message:
What to exclude from the diet.")
        add: str = Field(description="If included in the message:
What to add from the diet.")
        calories: str = Field(default=2400,
                              description="If included in the
message: Any calorie information. Default is 2400.")
        type_of_diet: str = Field(description="If included in the
message: Any information about the desired type of diet.")
        meals_number: str = Field(default=3,
                                  description="If included in the
message: Any information number of meals per day. Default is 3.")
        other_restrictions: str = Field(
            description="If included in the message: Any information
about religious, cultural, or ethical dietary restrictions.")


    class Indication(BaseModel):
```

```
        indeication_index: int = Field(description="Index of similar
indications.")
        explanation: str = Field(description="Why you think so?")


    class Dish(BaseModel):
        name: str = Field(description="Name of the dish")
        ingredients: List[str] = Field(description="List of
ingredients for the dish")
        cooking_instructions: str = Field(description="Cooking
instructions for the dish")
        price: float = Field(description="Approximate price of the
dish")


    class Meal(BaseModel):
        meal_number: int = Field(description="Number of the meal")
        description: str = Field(description="Description of the
meal")
        dishes: List[Dish] = Field(description="List of dishes for
the meal")
        total_price: float = Field(description="Total approximate
price of the meal")


    class DailyMenu(BaseModel):
        meals: List[Meal] = Field(description="List of meals for the
day")
        meals_number: Optional[int] = Field(description="Number of
meals for the day")
        total_price: float = Field(description="Total approximate
price of all meals for the day")
        currency: str = Field(description="Local currency")
```

```
    class DailyMenuList(BaseModel):
        menus: List[DailyMenu] = Field(description="List of daily
menu options")



    # Define your functions
    def get_model(api_key, model) -> ChatOpenAI:
        return ChatOpenAI(openai_api_key=api_key, model=model,
temperature=0)



    def get_user_info(user_message: str, model: ChatOpenAI) ->
UserInfo:
        query = (
            f"Extract information about the user's health problems,
his allergies, intolerances, "
            f"wishes what exclude, wishes what add, calorie
information, type of diet and other_restrictions from the message:
{user_message}."
        )
        parser = JsonOutputParser(pydantic_object=UserInfo)
        prompt = PromptTemplate(
            template="Answer the user
query.\n{format_instructions}\n{query}\n",
            input_variables=["query"],
            partial_variables={"format_instructions":
parser.get_format_instructions()},
        )
        chain = prompt | model
        output = chain.invoke({"query": query})
        try:
            parsed_output = parser.parse(output.content)
        except OutputParserException as e:
            fix_parser = OutputFixingParser.from_llm(parser=parser,
llm=model)
```

```
            parsed_output = fix_parser.parse(output.content)
        return parsed_output



    def get_indication_info(user_info: UserInfo, indications_list:
List[tuple], model: ChatOpenAI) -> Indication:
        query = (
            f"'List of medical indications': {indications_list}. "
            f"Please write the index of the medical indication that
corresponds to the information about the health of the user:
'{user_info['health_problems']}'. "
            "If user_info contains rubbish not related to medicine or
you did not find an exact match in the 'List of medical indications',
then indicate the index 99."
        )
        parser = JsonOutputParser(pydantic_object=Indication)
        prompt = PromptTemplate(
            template="Answer the user
query.\n{format_instructions}\n{query}\n",
            input_variables=["query"],
            partial_variables={"format_instructions":
parser.get_format_instructions()},
        )
        chain = prompt | model
        output = chain.invoke({"query": query})
        try:
            parsed_output = parser.parse(output.content)
        except OutputParserException as e:
            fix_parser = OutputFixingParser.from_llm(parser=parser,
llm=model)
            parsed_output = fix_parser.parse(output.content)
        return parsed_output
```

```
    def get_diet_options(temp: Dict, user_info: UserInfo, model:
ChatOpenAI) -> DailyMenuList:
        query = (
            f"{temp} \n"
            "You are a dietary cook, your work is very important, so
do it responsibly and carefully, "
            "do not invent non-existent dishes and follow dietary
recommendations. "
            f"Taking into account the dietary recommendations of
'{temp['diet_name']}', "
            f"create for me a list of {3} balanced daily menu
options. "
            f"Each daily menu on your list should consist of {5}
meals."
            f"Each daily menu dish on your list should not contain
products that may cause allergies: '{user_info['allergies']}' and
intolerances: '{user_info['intolerances']}'."
            "User is currently located in the country Ukraine. Please
indicate prices for food products in UAH."
        )
        parser = JsonOutputParser(pydantic_object=DailyMenuList)
        prompt = PromptTemplate(
            template="Answer the user
query.\n{format_instructions}\n{query}\n",
            input_variables=["query"],
            partial_variables={"format_instructions":
parser.get_format_instructions()},
        )
        chain = prompt | model
        output = chain.invoke({"query": query})
        try:
            parsed_output = parser.parse(output.content)
        except OutputParserException as e:
            fix_parser = OutputFixingParser.from_llm(parser=parser,
llm=model)
```

```
            parsed_output = fix_parser.parse(output.content)
        return parsed_output


    def translate_text(text: str, dest_language: str = 'uk') -> str:
        translator = googletrans.Translator()
        result = translator.translate(text, dest=dest_language)
        return result.text


    def translate_dict(data: dict, dest_language: str = 'uk') ->
dict:
        if isinstance(data, dict):
            return {k: translate_dict(v, dest_language) for k, v in
data.items()}
        elif isinstance(data, list):
            return [translate_dict(item, dest_language) for item in
data]
        elif isinstance(data, str):
            return translate_text(data, dest_language)
        else:
            return data


    def main(user_message, api_key, language):
        # Main code
        model_3_5 = get_model(api_key, "gpt-3.5-turbo-0125")
        model_4 = get_model(api_key, "gpt-4-0125-preview")
        diet_data_list = diets.get_diet_data_list()
        diet_data_list_en = diets.get_diet_data_list_en()
        user_info = get_user_info(user_message, model_3_5)
        indications_list = [(i, diet['indications']) for i, diet in
enumerate(diet_data_list_en)]
        indications_list.append((99, "If it doesn't exactly match any
other."))
```

```
        indication_info = get_indication_info(user_info,
indications_list, model_3_5)
        indeication_index = int(indication_info['indeication_index'])
        temp = diet_data_list[indeication_index].copy()
        # temp.pop('diet_name')
        temp.pop('indications')
        temp.pop('purpose')
        temp.pop('eating_regime')
        diet_options = get_diet_options(temp, user_info, model_4)
        if language == "Ukrainian ***:red[beta]***":
            translated_diet_options = translate_dict(diet_options)
            return translated_diet_options
        else:
            return diet_options



    if __name__ == "__main__":
        # Hello, I have a cold, please help me create a dietary menu
for me. Please note that I am allergic to tree nuts and peanuts. I
also have lactose intolerance.
        user_message = ("Hello, I have a cold, please help me create
a dietary menu for me. "
                        "Please note that I am allergic to tree nuts
and peanuts. "
                        "I also have lactose intolerance.")
        # Call the main function
        user_info = main(user_message, st.secrets["OPENAI_API_KEY"])
        print(user_info)
```

diets.py

```
    diet_data_list_en = [{'diet_name': 'Diet № 1',
                          'indications': 'acute diseases of the upper
parts of the alimentary canal, which last for 1-3 months, or chronic
diseases that have worsened (esophagitis), gastritis with increased or
```

normal acidity, duodenitis, peptic ulcer disease of the stomach and duodenum in the phase of moderate exacerbation.',
'purpose': 'Promoting the healing of stomach and duodenal ulcers, elimination of inflammatory processes in the upper parts of the alimentary canal.',
'excluded_foods': 'foods that remain in the stomach for a long time and irritate it (animal fats, vegetable fiber), stimulants of gastric secretion, spicy foods, spices, strong broths, mushrooms, legumes, coffee, chocolate, cocoa.',
'restricted_foods': 'table salt, lactic acid products.',
'eating_regime': 'The diet involves eating 6-7 times a day every 2-3 hours. ',
'recommended_foods': 'slimy soups from cereals (oat, buckwheat, rice, semolina), milk, cream, fresh low-fat sour cream, beef, veal, steamed fish cutlets, meatballs, mashed porridge, boiled eggs, pickled berries and fruits, diluted juices from them, bread '},…