НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

До захисту допущено:
Завідувач кафедри
_____ Олена ЧУМАЧЕНКО
«___»_____2023 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту» спеціальності 122 «Комп'ютерні науки»

на тему: «Використання великих мовних моделей в персоналізованій превентивній медицині XXI століття»

Виконала:
студентка IV курсу, групи КІ-93
Супрунюк Юлія Володимирівна _____

Керівник:
к.ф-.м..н., доцент Тимошенко Ю.О. _____

Консультант з нормоконтролю:
фахівець першої категорії, Гончарук М.М. _____

Консультант з економічного розділу:
доцент, к.е.н., Рощина Н.В. _____

Рецензент:
професор, д.т.н., Дичка І. А. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ, 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____Олена ЧУМАЧЕНКО
«___» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Супрунюк Юлія Володимирівна

1. Тема роботи «Використання великих мовних моделей в персоналізованій превентивній медицині XXI століття», керівник роботи к.ф-.м..н., доцент Тимошенко Юрій Олександрович, затверджені наказом по університету від «30» травня 2023 р. № 2065-с
2. Термін подання студентом роботи 08.06.2023
3. Вихідні дані до роботи – архітектура додатку медичних рекомендацій.
4. Зміст роботи – 1. Дослідження предметної області; 2. – Теоретичні аспекти великих мовних моделей; 3. Аналіз можливостей використання великих мовних моделей в персоналізованій превентивній медицині; 4. Функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) – 1. Презентація до захисту роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, к.е.н., Рощина Н.В.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Підготовка даних до роботи	27.02.2023	Виконано
2.	Вивчення літератури за темою роботи	15.03.2023	Виконано
3.	Підготовка першого розділу	30.03.2023	Виконано
4.	Підготовка другого розділу	27.04.2023	Виконано
5.	Підготовка третього розділу	20.05.2023	Виконано
6.	Підготовка економічної частини	30.05.2023	Виконано
7.	Підготовка презентації доповіді	03.06.2023	Виконано
8.	Оформлення дипломної роботи	05.06.2023	Виконано

Студент Юлія СУПРУНЮК

Керівник Юрій ТИМОШЕНКО

АНОТАЦІЯ

до бакалаврської дипломної роботи Супрунюк Юлії Володимирівни
на тему: «Використання великих мовних моделей в персоналізованій превентивній медицині XXI століття»

Дана робота присвячена дослідженню можливостей використання великих мовних моделей для покращення персоналізованої превентивної медицини. У роботі розглядається поняття превентивної медицини, її значення та основні принципи.

Крім того, досліджується роль великих мовних моделей у покращенні превентивної медицини. Представлено детальний огляд таких моделей, як GPT (Generative Pre-trained Transformer) і Med-PaLM. Описано архітектуру та принципи роботи цих моделей. У роботі також описано архітектуру додатку, який буде допомагати користувачам слідкувати за станом свого здоров'я та надавати персоналізовані рекомендації на основі показників здоров'я. Додаток буде використовувати GPT для аналізу інформації, отриманої від користувачів, та, буде здатний виявляти потенційні ризики, а також рекомендувати заходи з профілактики і покращення здоров'я.

Дана робота та її результати буде корисна для людей, які піклуються про своє здоров’я та медичних установ, щоб полегшити роботу з пацієнтами. Потенційний додаток може значно полегшити процес моніторингу стану здоров'я та надавати цінні рекомендації користувачам для підтримки їх загального благополуччя та профілактики захворювань.

Загальний обсяг роботи: 84 с., 10 рис., 8 таблиць, 25 джерело.

Ключові слова: превентивна медицина, великі мовні моделі, GPT, Med-PaLM, персоналізація, здоров'я, рекомендації, профілактика.

ABSTRACT

for the bachelor’s thesis of Supruniuk Yuliia Volodymyrivna
on the topic: "The Use of Large Language Models in Personalized Preventive Medicine of the 21st century"

This work is devoted to the study of the possibilities of using large language models to improve personalized preventive medicine. The paper discusses the concept of preventive medicine, its meaning and basic principles.

In addition, the role of large-scale language models in improving preventive medicine is explored. Presented a detailed overview of such models as GPT (Generative Pre-trained Transformer) and Med-PaLM. Described the architecture and principles of these models.

The paper also describes the architecture of an application that will help users monitor their health and provide personalized recommendations based on health indicators. The application will use GPT to analyze the information received from users and will be able to identify potential risks and recommend measures to prevent and improve health.

This work and its results will be useful for people who care about their health and medical institutions to facilitate work with patients. A potential application could greatly facilitate the process of health monitoring and provide valuable recommendations to users to support their overall well-being and disease prevention.

Total volume of work: 84 p., 10 figures, 8 tables, 25 sources.

Keywords: preventive medicine, big language models, GPT, Med-PaLM, personalization, health, recommendations, prevention.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ППППМ – Прогностична, превентивна та персоналізована медицина;

ВММ – Велика мовна модель;

GPT – Generative Pre-trained Transformer;

BERT – Bidirectional Encoder Representations from Transformers;

RNN – Recurrent Neural Network;

LSTM – Long short-term memory;

ШІ – Штучний інтелект;

ВСТУП

Захворювання та медичні проблеми суттєво впливають на якість життя і його тривалість. Велика частина захворювань може бути попереджена або виявлена на ранніх стадіях, однак, в багатьох країнах превентивна медицина не отримує достатньої уваги та фінансування порівняно з куративною медициною, яка спрямована на лікування хвороб. Це може призводити до втрати можливостей у запобіганні хвороб та зниження ефективності медичної системи.

Крім того, зростаюча потреба у медичних послугах, старіння населення та збільшення захворюваності призводять до того, що кількість лікарів не вистачає для задоволення потреб пацієнтів. Це може призвести до затримок у наданні медичної допомоги, зменшення якості обслуговування та загрози здоров'ю громадськості. Брак лікарів також може обмежувати розвиток превентивної медицини, оскільки не вистачає фахівців, які зосереджуються на запобіганні хвороб.

Одним зі способів вирішення цих проблем є використання технологій та інновацій у медицині. Великі мовні моделі, зокрема, можуть використовуватись для автоматизації процесів аналізу медичної інформації, підтримки прийняття рішень та надання персоналізованих рекомендацій. Вони можуть допомогти зробити превентивну медицину більш доступною та ефективною, дозволяючи пацієнтам самостійно слідкувати за своїм здоров'ям і отримувати інформовані рекомендації для профілактики хвороб.

Метою цієї дипломної роботи є дослідження складових та принципів роботи великих мовних моделей та розробка архітектури, яка продемонструє як саме можна використовувати великі мовні моделі в персоналізованій превентивній медицині.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 ППППМ як сучасна парадигма у охороні здоров'я

Традиційні підходи до охорони здоров'я в основному зосереджені на діагностуванні та лікуванні хвороб після того, як вони вже проявилися. Сфера охорони здоров'я стрімко розвивається завдяки появі нових технологій, досягненням молекулярної біології та кращому розумінню людського геному. І такий розвиток медицини відкрив шлях до трансформаційних змін у сфері, переходу до більш проактивного, превентивного та персоналізованого підходу. Ця нова парадигма, відома як прогностична, превентивна та персоналізована медицина (ППППМ), має потенціал для значного покращення результатів лікування пацієнтів, зменшення витрат на охорону здоров'я та оптимізацію надання медичної допомоги.

Прогностична медицина використовує передові технології, такі як секвенування(процес визначення послідовності нуклеотидів у ДНК або РНК молекулі) всього геному, для визначення генетичних маркерів, які можуть вказувати на ризик розвитку певних захворювань. Ця інформація може бути використана для прогнозування початку та прогресування хвороби, що дозволить вчасно втрутитись та вжити профілактичних заходів. [1, 2]

Розуміючи генетичну схильність людини, медичні працівники можуть розробити персоналізовану оцінку ризиків і відповідно адаптувати профілактичні стратегії. Такий підхід може допомогти виявити осіб з високим ризиком розвитку хвороб, полегшити ранню діагностику, а також запобігти або відстрочити початок захворювань.

Профілактична медицина підкреслює важливість раннього втручання для запобігання або пом'якшення наслідків захворювань. Цей підхід ґрунтується на виявленні факторів ризику, таких як генетична схильність, фактори

навколишнього середовища та вибір способу життя, для розробки цілеспрямованих стратегій профілактики [1, 2].

Профілактичні заходи можуть включати модифікацію способу життя, наприклад, зміну раціону харчування та фізичних навантажень, а також фармакологічні втручання для зниження ризику виникнення або прогресування захворювання. Ці стратегії спрямовані на зменшення загального тягаря хвороб для окремих людей і систем охорони здоров'я.

Персоналізована медицина передбачає адаптацію медичних втручань на основі унікального генетичного складу людини, клінічної історії та факторів навколишнього середовища. Розуміючи специфічні біологічні особливості пацієнтів, медичні працівники можуть розробляти більш ефективні методи лікування з меншою кількістю побічних ефектів. [1, 2]

Такий підхід може призвести до покращення результатів лікування пацієнтів, оскільки він враховує унікальні біологічні та екологічні фактори, які впливають на прогресування захворювання та реакцію на лікування. Персоналізована медицина також може сприяти розробці спеціальних методів лікування, таких як, наприклад, прецизійні методи лікування онкології, які спрямовані на конкретні мутації раку та розробку індивідуальних підходів до лікування, заснованих на особливостях пухлини та пацієнта.

Партисипативна медицина, у свою чергу, залучає пацієнтів як активних учасників процесу охорони їхнього здоров'я. Цей підхід заохочує пацієнтів брати на себе відповідальність за своє здоров'я і співпрацювати з медичними працівниками для розробки персоналізованих планів охорони здоров'я [1, 2].

Залучення пацієнтів до процесів прийняття рішень у сфері охорони здоров'я може підвищити задоволеність пацієнтів, покращити дотримання планів лікування та сприяти формуванню у них почуття відповідальності за результати свого здоров'я. Прикладом партисипативної медицини може стати

співпраця пацієнта з діабетом зі своєю медичною командою для управління своїм станом. Вони можуть працювати разом, щоб створити персоналізований план лікування, відстежувати прогрес і коригувати лікування за потреби. Пацієнт використовуватиме, наприклад, мобільний додаток для відстеження рівня цукру в крові, дієти та фізичних навантажень, ділячись цими даними зі своїми лікарями. Такий спільний підхід дасть можливість пацієнту брати активну участь у лікуванні, що призведе до кращих результатів та покращення якості життя.

Використовуючи досягнення геноміки, системної біології та інших високопродуктивних технологій, ПППМ має потенціал для революції у сфері охорони здоров'я та покращення результатів лікування пацієнтів [1].

Інтеграція прогностичної, профілактичної, персоналізованої та партисипативної медицини може призвести до створення більш ефективних та дієвих систем охорони здоров'я, зменшення навантаження на медичних працівників та покращення загальної якості життя пацієнтів. Оскільки дослідження в цій галузі продовжують розвиватися, важливо включити модель ПППМ в політику, освіту і практику охорони здоров'я, що в кінцевому підсумку прокладе шлях до більш стійкої і орієнтованої на пацієнта системи охорони здоров'я.

1.2 Штучний інтелект в медицині

Штучний інтелект (ШІ) — це термін, який використовується для опису використання комп’ютерів і технологій для моделювання інтелектуальної поведінки та критичного мислення, порівнянного з людським. У 1956 році Джон Маккарті вперше описав термін штучний інтелект як науку та техніку створення інтелектуальних машин. [3]

Штучний інтелект (ШІ) і пов’язані з ним технології все більше поширюються в суспільстві та бізнесі, і починають застосовуватися в охороні здоров’я. З моменту появи машинного навчання та глибокого навчання, застосування штучного інтелекту в медицині розширилися, що створило можливості для персоналізованої медицини, а не лише на основі алгоритмів. Ці технології мають потенціал для трансформації багатьох аспектів: від діагностики та лікування, до адміністративних та операційних процесів. Вони можуть заощадити час, гроші та підвищити якість догляду: від роботизованої хірургії до віртуальних помічників медсестер. [4]

Зараз у медичній сфері вже використовується штучний інтелект, починаючи від онлайн-запису на прийом, онлайн-реєстрації в медичних центрах, оцифрування медичних записів, нагадувань про повторний огляд і вакцинацію до ліків, алгоритмів дозування та попередження про побічні ефекти при призначенні кількох лікарських засобів. На рис. 1.1 наведено широкі можливості застосування ШІ в медицині.

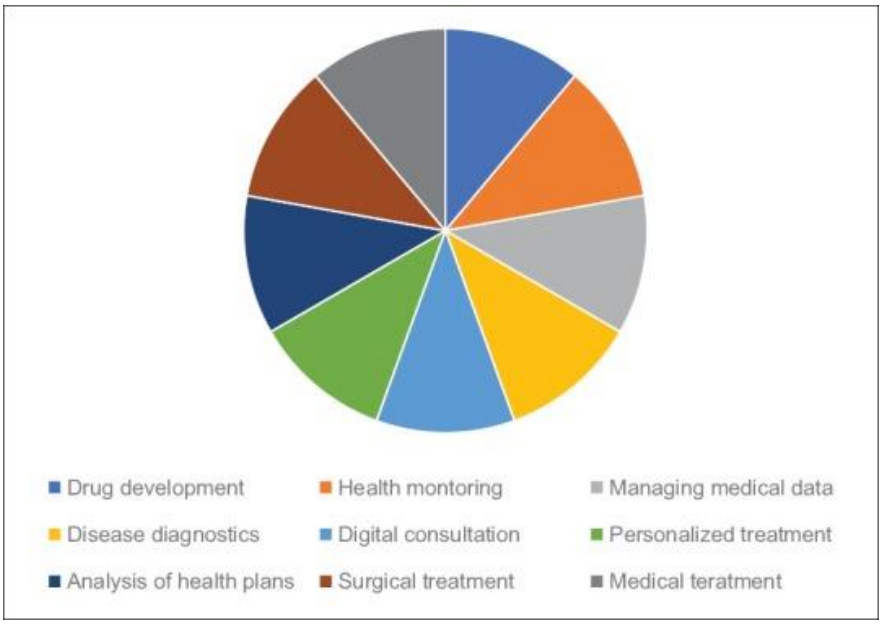

Рисунок 1.1 – Застосування штучного інтелекту в охороні здоров'я [3]

Штучний інтелект використовується для аналізу медичних зображень, таких як рентгенограми, комп'ютерна томографія та магнітно-резонансна

томографія, щоб допомогти виявляти аномалії та діагностувати захворювання. Arterys та Zebra Medical Vision використовують алгоритми штучного інтелекту для такого аналізу, покращуючи точність та швидкість діагностики.

Компанії, такі як Intuitive Surgical та Stryker, використовують штучний інтелект для розробки хірургічних роботів, які використовуються для допомоги хірургам під час операцій і можуть покращити результати лікування пацієнтів та скоротити час відновлення.

Штучний інтелект використовується для створення персоналізованих планів лікування пацієнтів на основі їх генетичного профілю та медичної історії. Компанії, такі як Tempus та 2bPrecise, користуються цим для аналізу даних пацієнтів та планів лікування.

Алгоритми штучного інтелекту використовуються для аналізу електронних медичних записів (EHR) з метою поліпшення клінічного прийняття рішень. Cerner та IBM Watson Health використовують штучний інтелект для аналізу даних EHR та виявлення закономірностей та інсайтів, які можуть допомогти лікарям більш ефективно діагностувати та лікувати пацієнтів.

Штучний інтелект використовується для прискорення відкриття та розробки ліків, зменшуючи час та вартість випуску нових ліків на ринок. Insilico Medicine та Atomwise використовують алгоритми штучного інтелекту для аналізу величезної кількості даних та прогнозування ефективності нових ліків.

HealthTap та Your.MD використовують штучний інтелект для створення чат-ботів та віртуальних асистентів, які можуть взаємодіяти з пацієнтами на природній мові, надаючи персоналізовані поради щодо здоров'я, відповіді на медичні запитання та планування записів на прийом.

Алгоритми штучного інтелекту використовуються для аналізу великих наборів даних пацієнтів та прогнозування результатів здоров'я, таких як ймовірність розвитку певних захворювань або ускладнень. Ця інформація може бути використана для розробки більш ефективних планів профілактики та лікування. Hindsait використовує штучний інтелект для надання прогнозувальної аналітики в охороні здоров'я.

Датчики та носима електроніка, що працюють на штучному інтелекті, використовуються для віддаленого моніторингу пацієнтів та надання реального часу даних медичним працівникам. Це може допомогти у запобіганні повторних госпіталізацій та наданні більш персоналізованої допомоги. Компанії, такі як VitalConnect, використовують штучний інтелект для розробки рішень з моніторингу пацієнтів.

Крім того, збирання даних з різних медичних пристроїв та смарт-годинників також може допомогти пацієнтам отримувати корисну інформацію про своє здоров'я та отримувати рекомендації щодо підтримки його. Наприклад, Fitbit пропонує своїм користувачам можливість вимірювання серцевого ритму та рівня кисню в крові, а також ведення щоденника їжі та води. Ці дані можуть бути оброблені ШІ, щоб побудувати модель пацієнта та рекомендувати конкретні дії для підтримки його здоров'я.

І це лише кілька прикладів того, як штучний інтелект використовується в охороні здоров'я. З розвитком технологій штучного інтелекту ми можемо очікувати ще більше інноваційних застосувань у майбутньому.

1.3 Великі мовні моделі: визначення, класифікація, основні характеристики

Великі мовні моделі – це моделі машинного навчання, які використовують глибоке навчання та натреновані на величезній кількості текстових даних для створення тексту, подібного до того, який написала б людина. Запровадження архітектури Transformer стало важливим проривом, який дозволив моделям виявляти складні шаблони та зв'язки у тексті, що дозволяє їм виконувати завдання, такі як генерація тексту, переклад, аналіз тональності, та відповіді на питання [5].

Величезні обсяги тексту вводяться в алгоритм штучного інтелекту за допомогою неконтрольованого навчання – коли моделі надається набір даних без чітких вказівок, що з ним робити. Завдяки цьому методу модель вивчає слова, а також зв’язки між ними та поняття, що стоять за ними. І так само, як людина, яка володіє мовою, може здогадатися, що може бути наступним у реченні чи абзаці – або навіть сама придумати нові слова чи поняття – велика мовна модель може застосувати свої знання для прогнозування та генерування вмісту.

Великі мовні моделі також можна налаштувати для конкретних випадків використання, у тому числі за допомогою таких методів, як точне налаштування(fine-tuning) або оперативне налаштування(prompt tuning), що є процесом передачі моделі невеликих фрагментів даних, на яких потрібно зосередитися, щоб навчити її для конкретної програми [6].

Існує кілька способів класифікації великих мовних моделей, але загальний підхід базується на їх архітектурі та методах попереднього навчання. Деякі відомі приклади включають:

- Моделі на основі Transformer: ці моделі використовують механізми самоуваги для паралельної обробки вхідних послідовностей, на відміну

від традиційних рекурентних або згорткових нейронних мереж [5]. Трансформери стали основою для багатьох передових мовних моделей, таких як BERT та GPT-3.

- BERT (Bidirectional Encoder Representations from Transformers): BERT - це модель на основі Transformer, яка під час попереднього навчання використовує завдання з маскуванням мовних моделей та прогнозуванням наступного речення [7]. Це дозволяє BERT вчитися з обох сторін контексту, що покращує його здатність виконувати завдання NLP.
- GPT-3 (Generative Pre-trained Transformer 3): GPT-3 - це потужна модель на основі Transformer, яка використовує попередньо навчені знання та точне налаштування для адаптації до конкретних завдань NLP з допомогою невеликої кількості прикладів[8]. GPT-3 відомий своєю великою кількістю параметрів та високою точністю в багатьох завданнях NLP.

- Моделі з генеративним попереднім навчанням: ці моделі базуються на попередньому навчанні з використанням наборів даних, які містять структуровані та неструктуровані дані, для підвищення продуктивності моделей у майбутніх завданнях NLP [9].
- Моделі з переносом навчання: ці моделі використовують знання, набуті під час навчання на одних задачах, для поліпшення продуктивності на інших завданнях NLP [10]. Цей підхід допомагає ефективно використовувати навчальні ресурси та досягати кращих результатів з меншою кількістю даних.
- ELMo (Embeddings from Language Models): ELMo є одним з ранніх прикладів використання контекстно-залежних слівних векторів для поліпшення різноманітних завдань NLP [11]. ELMo створює вбудовування слів на основі різних рівнів репрезентації мовної моделі, надаючи більш глибоке розуміння контексту для кожного слова.

Ці великі мовні моделі можуть бути поділені на різні класи на основі архітектури, підходів до навчання та способів використання. Загалом, вони продемонстрували значні успіхи в багатьох завданнях NLP, таких як машинний переклад, класифікація текстів, генерація тексту, відповіді на питання та багато інших.

Великі мовні моделі мають декілька ключових характеристик, які роблять їх потужними інструментами для обробки природної мови та інших завдань, пов'язаних з мовою:

- Масштаб та архітектура

Великі мовні моделі мають величезну кількість параметрів, що дозволяє їм вчитися та зберігати великі обсяги інформації. У цих моделях використовуються архітектури глибокого навчання, такі як Transformer. Великий масштаб цих моделей призводить до покращення продуктивності та можливостей узагальнення, які є важливими для вирішення складних завдань обробки природної мови[5].

- Навчальні дані та переднавчання

Мовні моделі навчаються на великих наборах даних, які містять текст з різних джерел, таких як книги, статті та веб-сайти. Процес переднавчання дозволяє моделям вивчити граматику, синтаксис, факти та навіть деякі здібності мислення. Великий обсяг навчальних даних дозволяє великим мовним моделям вловлювати тонкі взаємозв'язки між словами та словосполученнями, підвищуючи їхню здатність генерувати зрозумілий та контекстуально відповідний текст [7].

- Тонке налаштування (fine tuning)

Процес адаптації переднавченої мовної моделі до конкретних завдань або доменів. Цей етап дозволяє моделі стати точнішою та кориснішою для

спеціалізованих застосунків. Навчаючись на меншому наборі даних, який адаптовано до цільового завдання, доопрацювання допомагає моделі оптимізувати свою продуктивність без потреби в широкому навчанні. Цей процес забезпечує економію часу та обчислювальних ресурсів[12].

- Передавальне навчання(Transfer Learning)

Передавальне навчання дозволяє знанням, отриманим на одному завданні, застосовуватися до іншого, схожого завдання. Великі мовні моделі використовують цю можливість, що призводить до поліпшення продуктивності в різних завданнях обробки природної мови. Переднавчені моделі слугують основою, а процес переносу навчання дозволяє їм швидко адаптуватися до нових завдань з мінімальним додатковим навчанням. Ця особливість робить великі мовні моделі високоадаптивними та ефективними [13].

- Контрольованість та безпека

Оскільки мовні моделі стають потужнішими, контролювати їх і забезпечити безпечну генерацію контенту стає все важливішим. Дослідники розробляють методи покращення контрольованості та безпеки цих моделей. Ці підходи включають навчання з підкріпленням на основі відгуків людей, правило-орієнтовані винагороди та змагальне машинне навчання, серед іншого. Забезпечення того, щоб мовні моделі виробляли бажані та безпечні результати, є постійним викликом у галузі, і для удосконалення цих методів необхідні подальші дослідження [14].

Розуміння основних характеристик мовних моделей є важливим кроком для використання їх потенціалу та зменшення можливих ризиків.

1.4 Висновки першого розділу

Поєднання великих мовних моделей (ВММ) і ПППМ має потенціал для трансформації охорони здоров'я. Завдяки своїй здатності обробляти та аналізувати величезні обсяги даних, ВММ можуть допомогти у виявленні закономірностей, прогнозуванні результатів і наданні персоналізованих рекомендацій для поліпшення лікування пацієнтів. У поєднанні з принципами ППМД ці моделі можуть сприяти ранньому виявленню, профілактиці та індивідуалізованим втручанням, що в кінцевому підсумку покращити надання медичної допомоги та результати лікування пацієнтів.

Отже, у даній роботі метою є проведення аналізу таких великих мовних моделей як MedPalm і GPT. В рамках дослідження розгляд їх архітектури, процесу навчання, можливості та обмеження в застосуванні. Також аналіз того, як вони можуть бути використані в персоналізованій превентивній медицині.

РОЗДІЛ 2. ТЕОРЕТИЧНІ АСПЕКТИ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Великі мовні моделі є одним з найбільш досліджуваних напрямків у галузі штучного інтелекту, і за останні кілька років стали надзвичайно популярними. Вони дозволяють створювати програми, які можуть розуміти та генерувати людську мову, що відкриває безліч можливостей для застосування в різних галузях, включаючи медицину.

В даному розділі представлено теоретичне дослідження великих мовних моделей, зосереджуючись на їх архітектурі та принципах роботи. Розгляд великих мовних моделей дозволить зрозуміти їх потенціал використання в медицині.

2.1 Архітектура та принципи роботи великих мовних моделей

2.1.1 Еволюція великих мовних моделей

В області обробки природної мови відбулася значна кількість змін, і нині використовуються потужні мовні моделі для вирішення складних задач. Ці моделі еволюціонували від ранніх рекурентних нейронних мереж до сучасних трансформерів, забезпечуючи все більш високу точність та ефективність в отриманих результатах.

Першим значним кроком у розвитку мовних моделей стало виникнення рекурентних нейронних мереж. RNN (Recurrent Neural Network)[15] - це тип нейронної мережі, яка призначена для роботи з послідовністю даних. Відрізняючись від звичайних нейронних мереж, RNN призначені для обробки послідовностей будь-якого розміру. “Послідовність” означає, що кожен елемент вхідних даних має певний зв'язок зі своїми “сусідами” або вплив на них. RNN були розроблені з метою збереження інформації про попередні стани, щоб забезпечити моделювання залежностей в межах послідовності. Це дозволило

RNN використовувати контекст попередніх елементів для вирішення задач обробки послідовності, таких як мовний переклад, генерація тексту, аналіз тексту тощо.

Базові нейронні мережі з прямим поширенням також “пам'ятають” контекст, але вони пам'ятають те, що вони вивчили під час тренування. В цей же час, RNN навчаються аналогічно під час тренування, але також пам'ятають дані, які вони вивчили з попередніх вхідних даних під час генерації вихідних даних.

Типи рекурентних нейронних мереж можуть бути класифіковані за архітектурою та за призначенням. Ось деякі з основних типів RNN з огляду на їх застосування:

- Модель "Вектор-Послідовність" (Vector-Sequence Models) приймає послідовність фіксованого розміру на вхід та виводить послідовність довільної довжини. Наприклад, при створенні опису зображення в якості вхідних даних використовується зображення, а на виході отримується опис зображення.
- Модель "Послідовність-Вектор" (Sequence-Vector Model) приймає на вхід послідовність будь-якого розміру та виводить послідовність фіксованого розміру. Наприклад, аналіз емоційного забарвлення оцінки фільму класифікує відгук на фільм як позитивний або негативний у вигляді вектора фіксованого розміру.
- Модель "Послідовність-Послідовність"(Sequence-Sequence Model) – найбільш популярний та найчастіше використовуваний варіант, де на вхід приходить послідовність та на виході генерується інша послідовність вже іншої довжини. Такі моделі використовуються в задачах машинного перекладу, розшифровки тексту, відповіді на запитання та багатьох інших задачах обробки природних мов.

Рекурентні нейронні мережі мають деякі недоліки, які обмежують їх ефективність та застосування. Під час навчання RNN, градієнти можуть зменшуватися або збільшуватися експоненційно з часом, що призводить до затухання або вибухового градієнта. Це ускладнює навчання моделі та оновлення ваг. Крім того, RNN важко навчатися на даних з великою кількістю залежностей, оскільки мережа втрачає інформацію про попередні частини послідовності зі збільшенням часового кроку. Це обмежує здатність моделі вчитися зі складними текстами або послідовностями даних. RNN не мають вбудованого механізму уваги, який дозволив би моделі зосереджуватися на важливих аспектах вхідних даних під час вирішення задачі.

Ці недоліки спонукали дослідників шукати альтернативні архітектури, такі як LSTM та Transformer, які відомі своєю здатністю ефективно вирішувати проблеми з великими залежностями та обробляти великі послідовності даних.

Довга короткочасна пам'ять (LSTM) [16] - особливий вид рекурентної нейронної мережі (RNN), створений спеціально для вирішення проблеми затухання градієнту. Вони здатні навчатися на даних з великою кількістю залежностей. Нейрони LSTM, на відміну від звичайних нейронів, мають спеціальний механізм, який дозволяє передавати інформацію через окрему гілку, уникаючи необхідності у довготривалій обробці поточної комірки. Це дозволяє мережі зберігати пам'ять на довший період часу. Хоча це допомагає вирішувати проблему зникнення градієнту, але не є ідеальним рішенням - ефективність LSTM починає знижуватися після певної кількості слів, зазвичай близько 100. Для більш довгих послідовностей, таких як 1000 слів, LSTM може мати проблеми з ефективним зберіганням інформації.

Аналогічно звичайним RNN, навчання LSTM також дуже повільне, і навіть повільніше. Вони отримують вхідні дані послідовно, один за одним, що не дозволяє ефективно використовувати можливості обчислювальної потужності GPU, які розроблені для паралельних обчислень.

Після LSTM науковці продовжували розвивати нові та ефективніші архітектури. У 2017 році було представлено архітектуру Transformer, яка принесла значні зміни в галузі глибокого навчання.

Transformer [5] відрізняється від RNN та LSTM своєю архітектурою, яка повністю відмовляється від послідовності обчислень та рекурентності. Замість цього він використовує механізм уваги (attention mechanism), який дозволяє моделі зосереджуватися на різних аспектах вхідних даних під час вирішення задачі. Трансформер складаються з кодера та декодера, які містять декілька однакових шарів з механізмами уваги та повнозв'язними нейронними мережами.

Ключовими особливостями трансформаторів є:

- Механізм уваги дозволяє моделям зосереджуватися на важливих аспектах вхідних даних та ігнорувати менш важливі деталі.
- Трансформатори мають можливість паралельного обчислення, що сприяє швидкому навчанню та ефективній обробці великих наборів даних.
- Завдяки механізму уваги, трансформери можуть краще враховувати велику кількість залежностей між вхідними даними, ніж RNN та LSTM.

Трансформери стали основою для ряду потужних моделей мови, таких як BERT, GPT-2, GPT-3 та інші. Вони широко застосовуються в задачах машинного перекладу, класифікації тексту, аналізу емоційного забарвлення, відповіді на запитання, генерації тексту та багатьох інших.

2.1.2 Механізм уваги

Механізм уваги є одним з ключових компонентів Transformer, який дозволяє моделям ефективно працювати з послідовностями даних, аналізуючи важливі залежності між окремими елементами. Цей механізм значно покращує

здатність моделі вирішувати задачі, що вимагають використання контексту та великої кількості залежностей.

Основна ідея механізму уваги полягає в тому, що замість фіксованої ваги кожного слова в послідовності, модель навчається виділяти важливість різних слів, залежно від контексту.

Механізм уваги працює на основі трьох матриць, які є результатом лінійного перетворення вхідних даних. Ці матриці називаються ключами (K), значеннями (V) та запитами (Q). Вони використовуються для обчислення ваги уваги між різними словами в послідовності.

Алгоритм роботи механізму уваги складається з наступних кроків:

1. Обчислення матриць K, Q, V для кожного слова в послідовності:

$$Q = XW^Q, K = KW^K, V = XW^V, \tag{2.1}$$

$$W^Q \epsilon\, R^{d_{model} \times d_k},\ W^K \epsilon\, R^{d_{model} \times d_k},\ W^V \epsilon\, R^{d_{model} \times d_v}, \tag{2.2}$$

де X є вектором вхідних даних, а W^Q, W^K та W^V – це ваги, що навчаються, d_{model} – це розмір векторного представлення, d_k – розмір ключів, а d_v – розмір значень.

2. Обчислення скалярного добутку між матрицями Q та K, а також масштабування його за допомогою квадратного кореня розміру ключа (d_k):

$$atention\_score = \frac{QK^T}{\sqrt{d_k}} \tag{2.3}$$

Масштабування скалярного добутку має важливе значення в механізмі уваги. Це виконується для забезпечення належної обробки великих значень скалярного добутку. Без масштабування великі значення скалярного добутку можуть призвести до нестабільного навчання та зниження продуктивності механізму уваги.

Коли масштабування застосовується до скалярного добутку, воно допомагає контролювати величину ваг уваги. Це, в свою чергу, дозволяє моделі коректно розподілити увагу між різними словами в послідовності. В результаті, модель краще зосереджується на важливих словах та враховує відповідний контекст при обробці мовних даних.

3. Застосування функції активації softmax для нормалізації ваг уваги:

$$attention\_weights = softmax(attention\_score), \tag{2.4}$$

де $softmax$ – це функція активації, яка перетворює вхідні значення у ймовірності, які в сумі дають одиницю. Це забезпечує, що кожне слово отримує відповідний рівень уваги відносно інших слів у послідовності. Функція $softmax$ визначається наступним чином:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1} e^{x_j}} \tag{2.5}$$

де x – вхідний вектор.

4. Обчислення вихідного вектора за допомогою ваг уваги та матриці значень (V):

$$Attention = attention\_weights * V, \quad (2.6)$$

Результат обчислення є вектором, який відображає важливість кожного слова в послідовності відносно запиту.

2.1.3 Архітектура Transformer

Transformer – це архітектура, яка значно поліпшила якість машинного перекладу та інших завдань обробки мови. Головною особливістю трансформера є використання механізму уваги, який дозволяє моделям динамічно зосереджуватись на різних частинах вхідної послідовності під час обробки даних.

Архітектура трансформера складається з двох основних компонентів: енкодера та декодера. Кожен з цих компонентів містить кілька шарів, що складаються з різних блоків. Архітектура трансформера представлена на рис.2.1.

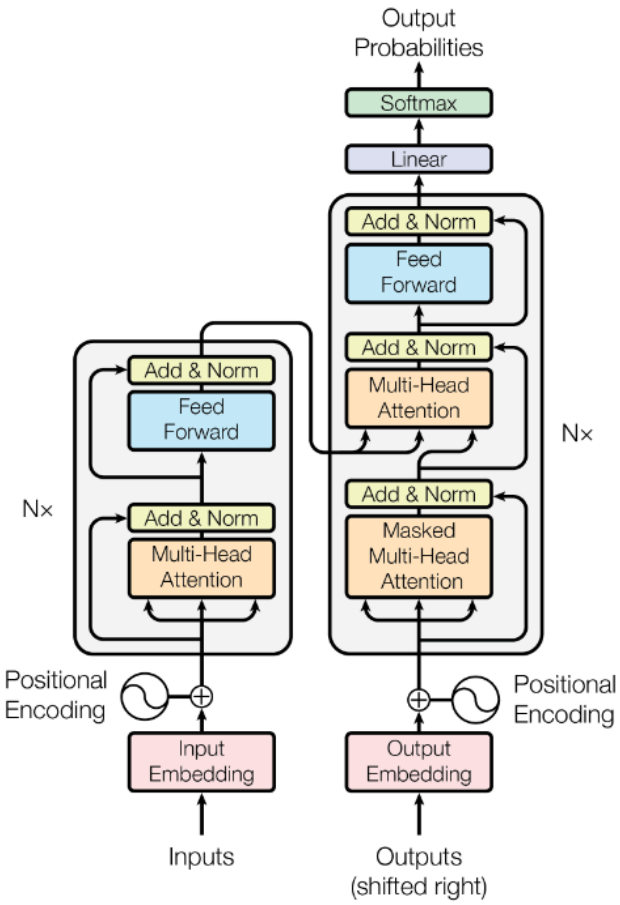

Рис. 2.1 – Архітектура моделі Transformer [5]

Енкодер перетворює вхідні дані (текст) в набір векторів, які відображають семантичне значення та контекст. Енкодер складається з N однакових шарів, кожен з яких містить два підшари: шар уваги на основі механізму самоуваги та позиційно-залежний шар повнозв'язної мережі.

Так як комп'ютери не розуміють слів, вони працюють з числами, векторами або матрицями, застосовується Input Embedding. Embedding Space це як відкритий простір або словник, де слова зі схожими значеннями згруповані разом або присутні близько одне до одного в цьому просторі. Цей простір називається простором вбудовування, і тут кожне слово, відповідно до його значення, зіставляється з певним значенням. Кожне слово представляється як вектор фіксованого розміру, що відображає його семантичне значення у багатовимірному просторі. Ці вектори вбудовування слів можуть навчатися

разом з моделлю або використовувати переднавчані вектори вбудовування, такі як word2vec, GloVe або BERT.

Позиційне кодування (Positional Encoding) відповідає за додавання інформації про порядок слів у послідовності до вбудовувань слів. Трансформер не має вбудованого поняття порядку слів у послідовності, тому позиційне кодування вводиться для збереження інформації про порядок слів. Це допомагає моделі враховувати порядок слів під час аналізу тексту та забезпечує зв'язок між словами у послідовності.

Позиційне кодування розраховується за допомогою синусоїдальних та косинусоїдальних функцій з різними частотами. Формули для позиційного кодування наступні:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}), \quad (2.7)$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}), \quad (2.8)$$

де pos – позиція слова у послідовності, i – індекс вектора вбудовування, d_{model} – розмірність вбудовування.

Після обчислення позиційних кодувань, вони додаються до вихідних векторів вбудовування слів. Таким чином, кожен вектор вбудовування містить інформацію як про семантичне значення слова, так і про його позицію у послідовності.

Multi-Head Attention – це механізм уваги, який використовується в моделі для взаємодії між різними елементами послідовності. Основна ідея полягає в тому, що вхідні дані розбиваються на декілька груп, кожна з яких обробляється окремо. Кількість груп, або "голів", може бути вибрана як параметр моделі.

Механізм допомагає виявляти смислові зв'язки між елементами послідовності, що можуть бути складними для виявлення однією групою.

Для кожної групи Multi-Head Attention обчислюється ваговий коефіцієнт уваги, для кожної пари вхідних векторів – запиту та ключа, який відображає ступінь важливості цих елементів. Вагові коефіцієнти використовуються для обчислення зваженої суми значень векторів значень, які пов'язані з кожним вхідним вектором, що дає вихідний вектор уваги. Результати обробки кожної групи об'єднуються в один вектор, що представляє увагу.

Feed Forward в моделі Transformer є важливим прошарком, який забезпечує здійснення нелінійних перетворень вихідних значень Multi-Head Attention. Це допомагає покращити репрезентацію вхідного рядка, зміцнюючи різні аспекти контексту та знижуючи вплив шуму або некорисних сигналів. Крім того, застосування Feed Forward допомагає уникнути проблеми згасання градієнту, яка може виникнути при використанні багатошарових нейронних мереж.

Декодер, в свою чергу, використовує вектори, отримані від енкодера, для генерації вихідного тексту (переклад, відповідь тощо). Декодер також складається з N шарів, але має три підшари: механізм уваги з маскуванням, механізм уваги з використанням вихідних даних енкодера, і позиційно-залежний шар повнозв'язної мережі.

Декодер починає роботу з ембедінгування вхідної послідовності. Кожен токен на вході перетворюється у вектор фіксованої довжини за допомогою ембедінгової матриці. Цей ембедінг зображує слово як вектор у відповідному просторі, що дозволяє йому взаємодіяти з іншими векторами в моделі.

Після ембедінгування декодер використовує маскування, щоб забезпечити, що на кожному кроці генерації майбутній контекст не буде використовуватися. Це досягається шляхом застосування маски, яка не дає

декодеру побачити частину вихідної послідовності, яка йшла після поточної позиції.

Далі декодер використовує Multi-Head Attention на вихідну послідовність. Для кожного шару декодера використовується окремий механізм уваги, який дозволяє декодеру зосередитися на різних аспектах вихідної послідовності. Multi-Head Attention генерує вектор контексту для кожного токена вхідної послідовності.

Після мульти-головної уваги декодер використовує механізм уваги на енкодер, щоб отримати вектор контексту для кожного токена вхідної послідовності. Це дозволяє декодеру збільшити контекст, що враховується при генерації кожного токена вихідної послідовності. Механізм уваги на енкодер використовує вектори контексту, згенеровані мульти-головною увагою на вихідну послідовність, для знаходження ваги для кожного токена вхідної послідовності.

Після отримання векторів контексту з мульти-головної уваги на вихідну послідовність та уваги на енкодер, декодер об'єднує ці вектори за допомогою конкатенації. Отриманий вектор контексту проходить через два шари Feed Forward для обробки та подальшої згортки. Це допомагає моделі збагатити вихідний вектор і здійснювати додаткову обробку для кращого відображення структури даних.

Останнім кроком декодера є генерація вихідного токена. Декодер використовує отриманий вектор контексту, щоб передбачити вихідний токен, який додається до вихідної послідовності. Процес генерації повторюється до тих пір, поки не буде згенерована вся вихідна послідовність.

Трансформер став переломним моментом у розробці великих мовних моделей, оскільки він дозволив досягти нового рівня точності та якості генерації природної мови. Завдяки трансформеру, стали можливими нові

застосування в області машинного навчання та обробки природної мови, що забезпечує збільшення продуктивності та ефективності у багатьох сферах, таких як технічна підтримка, автоматичне створення контенту та машинний переклад.

2.2 Огляд сучасних великих мовних моделей

2.2.1 GPT

GPT (Generative Pre-trained Transformer) – це мовна модель, яка використовує архітектуру Transformer для генерування тексту, схожого на створений людиною. Модель була представлена компанією OpenAI в червні 2018 року. Вона навчалась на великому масиві текстових даних у неконтрольований спосіб, що означає, що модель не потребує спеціальних анотацій чи міток під час процесу навчання. GPT використовує двоетапний процес навчання: попереднє навчання(pre-training) та точне налаштування(fine-tuning).

Під час попереднього навчання модель GPT обробляє вхідний текст фрагментами – токенами. Токени можуть представляти окремі слова або підслова, залежно від використовуваної схеми токенізації. Автори моделі використовували підхід на основі кодування байтових пар (BPE), який дозволяє моделі обробляти слова, що не входять до словника, розбиваючи їх на підсловосполучення(subword units).

Завданням навчання GPT є максимізація ймовірності передбачення наступної лексеми в послідовності, враховуючи попередні. Це досягається за рахунок процесу, який називається "masked language modeling". Під час попереднього навчання, випадковим чином, обирається певний відсоток вхідних токенів і замінюється на спеціальний токен, який називається

"[MASK]". Після цього, модель навчається передбачати вихідну лексему з контексту, на основі відомих токенів.

Після етапу попереднього навчання, GPT переходить до етапу точного налаштування. На цьому етапі, модель навчається виконувати конкретні подальші завдання, які вимагають розуміння мови, наприклад, класифікація тексту, відповіді на запитання або переклад. Процес точного налаштування передбачає надання маркованих даних для цих завдань і адаптацію попередньо навченої моделі для їхнього виконання.

Після попереднього навчання, GPT переходить до точного налаштування. Процес точного налаштування передбачає використання маркованих даних і адаптацію попередньо навченої моделі для їх виконання. Цей етап дозволяє моделі узагальнити своє розуміння мови і використовувати попередньо набуті знання для ефективного виконання різних подальших завдань. Завдяки попередньому навчанню на великому корпусі немаркованого тексту GPT здатна вивчати багаті репрезентації мови, фіксуючи синтаксичну, семантичну та контекстну інформацію. [9]

Експерименти, проведені авторами, продемонстрували, що GPT досягає високі результати на широкому спектрі еталонних наборів даних у різних завданнях NLP. Модель продемонструвала потужні можливості розуміння мови, включаючи закінчення речень, визначення схожості речень та генерацію логічно зв'язаного тексту.

2.2.1.1 GPT-3

GPT-3 – це модель 3-го покоління серії GPT-n, яка була представлена у травні 2020 року. Вона навчалась з 175 мільярдами параметрів. Модель може тренуватись за допомогою few-shot, one-shot, zero-shot навчання. При zero-shot модель прогнозує відповідь, маючи лише опис завдання природною мовою, при

one-shot, крім опису завдання, модель бачить єдиний приклад завдання, а few-shot - декілька. GPT-3 досягає 81.5 F1 на датасеті CoQA в режимі zero-shot, 84.0 F1 на CoQA в режимі one-shot, 85.0 F1 в режимі few-shot. Аналогічно, GPT-3 досягає 64.3% точності на датасеті TriviaQA в режимі zero-shot, 68.0% в режимі one-shot і 71.2% в режимі few-shot. Оцінка F1 є мірою точності (precision) та повноти (recall) моделі в задачах класифікації інформації. Вона об'єднує ці дві міри в одне число, яке відображає загальну ефективність моделі.

GPT-3, навчена за допомогою one-shot і few-shot, демонструє вміння виконувати завдання на швидку адаптацію та міркування "на льоту", які включають розшифровку слів, виконання арифметичних дій та використання нових слів у реченні після того, як вони були визначені лише один раз.

Для навчання GPT-3 було використано набір даних CommonCrawl, що охоплює період з 2016 по 2019 рік. До фільтрації ці дані займали приблизно 45 ТБ стисненого відкритого тексту, а після, набір даних зменшився до приблизно 570 ГБ, що приблизно еквівалентно 400 мільярдам токенів, закодованих парами байт.

Важливо зазначити, що процес навчання не передбачав вибірки наборів даних пропорційно до їхнього розміру. Замість цього, набори даних, які вважалися більш якісними, відбиралися частіше. В результаті, набори даних CommonCrawl та Books2 були відібрані менше одного разу під час навчання, в той час як інші набори даних були відібрані 2-3 рази. Такий підхід дозволив надати пріоритет більш якісним навчальним даним, а також погодитися на невелику кількість перенавчання, щоб отримали більш якісно натреновану модель. [8] В таблиці 2.1 відображено набори даних, які використовувались при навчанні моделі GPT-3.

Таблиця 2.1 – Набори даних, які використовувались для навчання GPT-3. [8]

Датасет	Кількість токенів	Відсоток в тренувальних даних
Common Crawl (відфільтрований)	410 мільярдів	60%
WebText2	19 мільярдів	22%
Books1	12 мільярдів	8%
Books2	55 мільярдів	8%
Wikipedia	3 мільярди	3%

Навчаючись на тексті, написаному людьми, GPT-3 також вчиться писати як людина, як з найкращими, так і найгіршими людськими характеристиками. Дослідження GPT-3 визначає кілька проблемних областей упередженості, як расизм і сексизм, як і можна було очікувати від моделі, навченої на даних з Інтернету.

Відповідно до GPT-3 продуктивність мовної моделі масштабується як степенева функція розміру моделі, розміру набору даних і обчислювальних ресурсів. Крім того, мовна модель з достатніми навчальними даними може вирішувати завдання НЛП, з якими вона ніколи не стикалася.

2.2.1.2 GPT-4

GPT-4 – це модель, представлена у березні 2023 року, яка може приймати на вхід як текст, так і зображення, генеруючи у відповідь текст. Модель демонструє результативність людського рівня, за різними професійними та академічними критеріями, такими як, наприклад, складання симульованого адвокатського іспиту з результатом у межах 10% найкращих учасників тестування.

У тесті MMLU, англомовному наборі запитань з множинним вибором, що охоплює 57 предметів, GPT-4 не лише значно випереджає існуючі моделі англійською мовою, але й демонструє високі показники іншими мовами. Модель GPT-4 була доопрацьована за допомогою методу навчання з підкріпленням на основі зворотного зв'язку (Reinforcement Learning from Human Feedback, RLHF).

На рис. 2.2. зображено результати оцінювання GPT-4 за дев'ятьма категоріями даних. Точність показана на осі Y, чим вище, тим краще. Точність 1.0 означає, що відповіді моделі відповідають правильним відповідям(як у людей) на всі запитання в оцінюванні. Порівняння відбувалось з трьома попередніми версіями ChatGPT, заснованими на GPT-3.5. GPT-4 є точніше моделі GPT-3.5 на 19%, причому значний приріст спостерігається в усіх темах. [17]

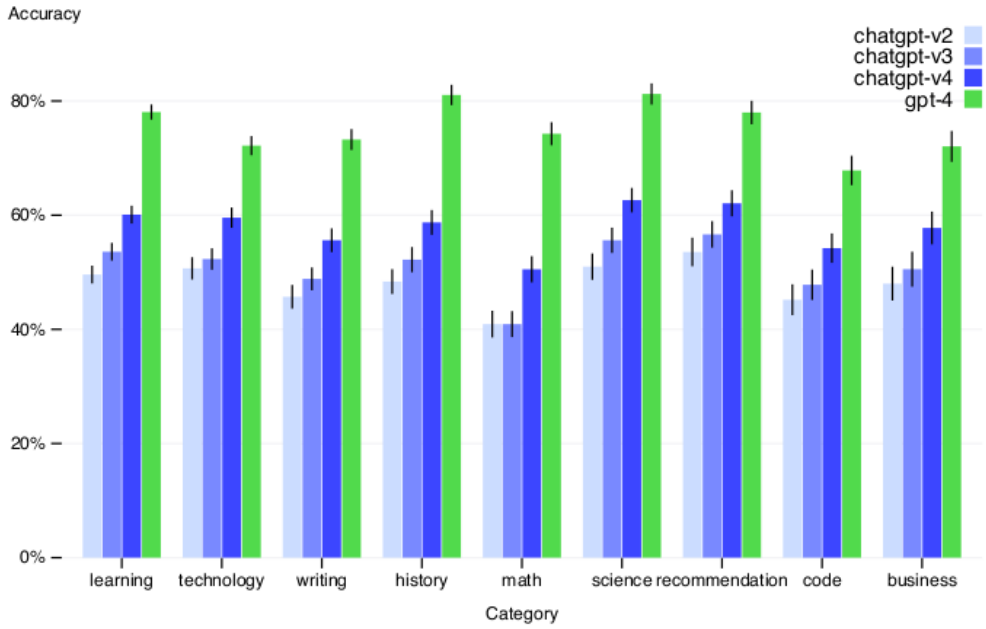

Рисунок 2.2 – Результати оцінювання GPT-4

2.2.1.2 ChatGPT

У листопаді 2022 року компанія OpenAI представила ChatGPT - чат-бот, який використовує штучний інтелект, розроблений на основі архітектури GPT-3.5(вдосконалена версія GPT-3). Модель була натренована надавати детальну відповідь за інструкцією в запиті.

Під час процесу налаштування моделі використовувалось як навчання з вчителем, так і навчання з підсиленням на основі зворотного зв'язку від людини (RLHF). Обидва підходи потребували людей-тренерів для покращення продуктивності моделі. У випадку навчання з вчителем моделі надавалися бесіди, у яких тренери грали роль як користувача, так і помічника ШІ. Під час навчання з підсиленням, люди спочатку оцінювали відповіді, створені моделлю у попередній бесіді, а потім ці оцінки використовувалися для створення "моделей нагороди", які далі використовувалися для додаткової налаштування моделі за допомогою кількох ітерацій методу наближеного покращення політики (PPO).

Основною функцією чат-бота є імітація розмови з людиною, проте він може генерувати код, допомагати в розробці програмного забезпечення, імітувати стиль виступів відомих людей та створювати бізнес-презентації, писати казки та студентські есе, відповідати на тестові питання (іноді, залежно від тесту, на рівні, що перевищує середній рівень людини-тестувальника), писати поезію та пісні, перекладати та узагальнювати тексти та багато іншого. Крім того, OpenAI збирає відгуки від користувачів ChatGPT, щоб навчати і додатково налаштовувати сервіс. Користувачі можуть голосувати за відповіді, які вони отримують від ChatGPT, і надсилати додаткові коментарі в текстовому полі.

На відміну від більшості чат-ботів, ChatGPT запам'ятовує деяку кількість попередніх запитів, наданих йому у тій самій розмові, і генерує наступні відповіді в контексті минулих. Щоб уникнути відображення та генерації образливих відповідей з боку ChatGPT, запити фільтруються через API "Moderation endpoint" OpenAI (окремий ШІ, заснований на GPT), і потенційно расистські або сексистські запити відкидаються.

На рис. 2.3 та рис. 2.4 зображено приклади запитів, які може обробляти ChatGPT.

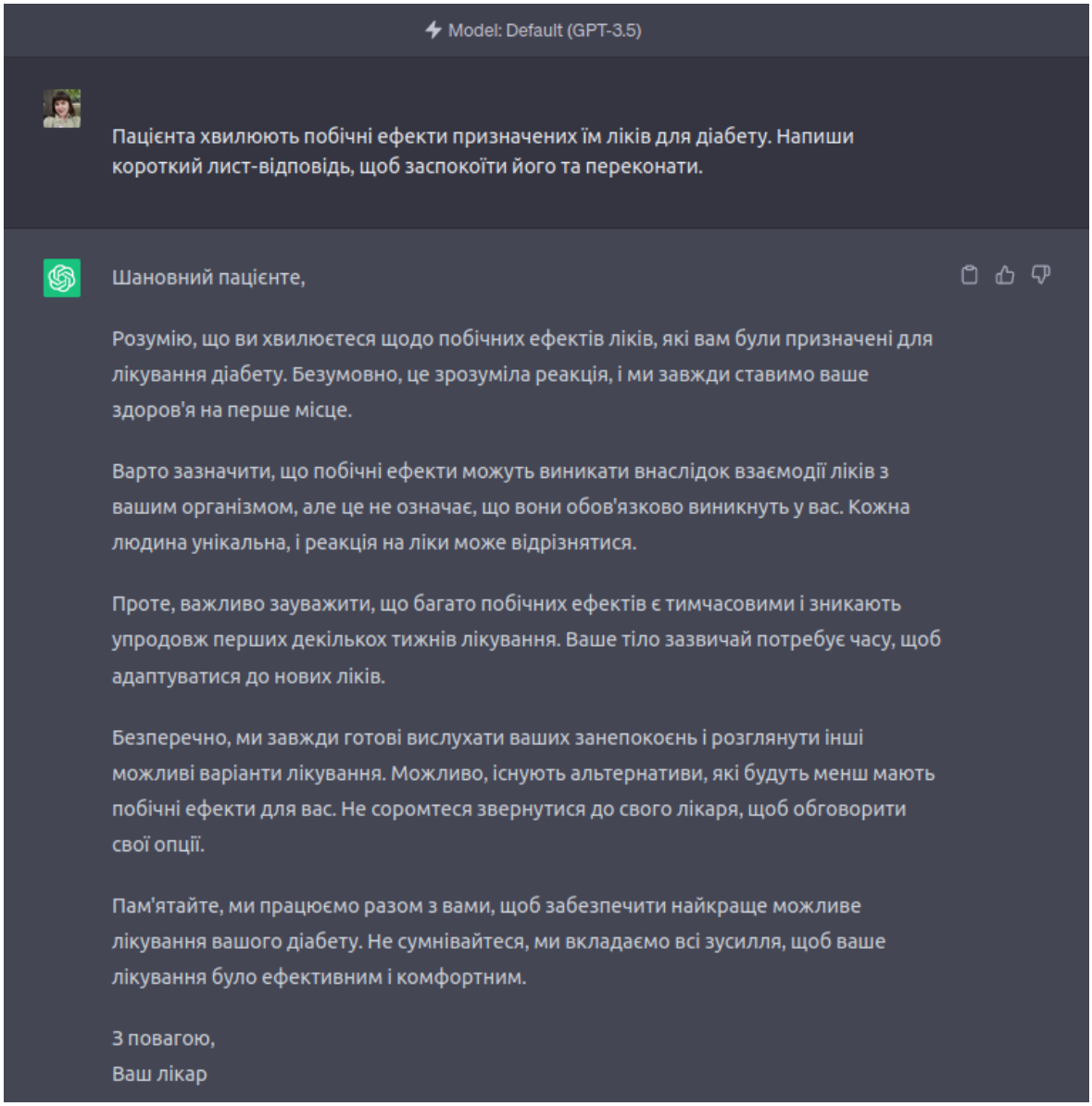

Рисунок 2.3 – Запит до ChatGPT написати листа до пацієнта, який приймає ліки.

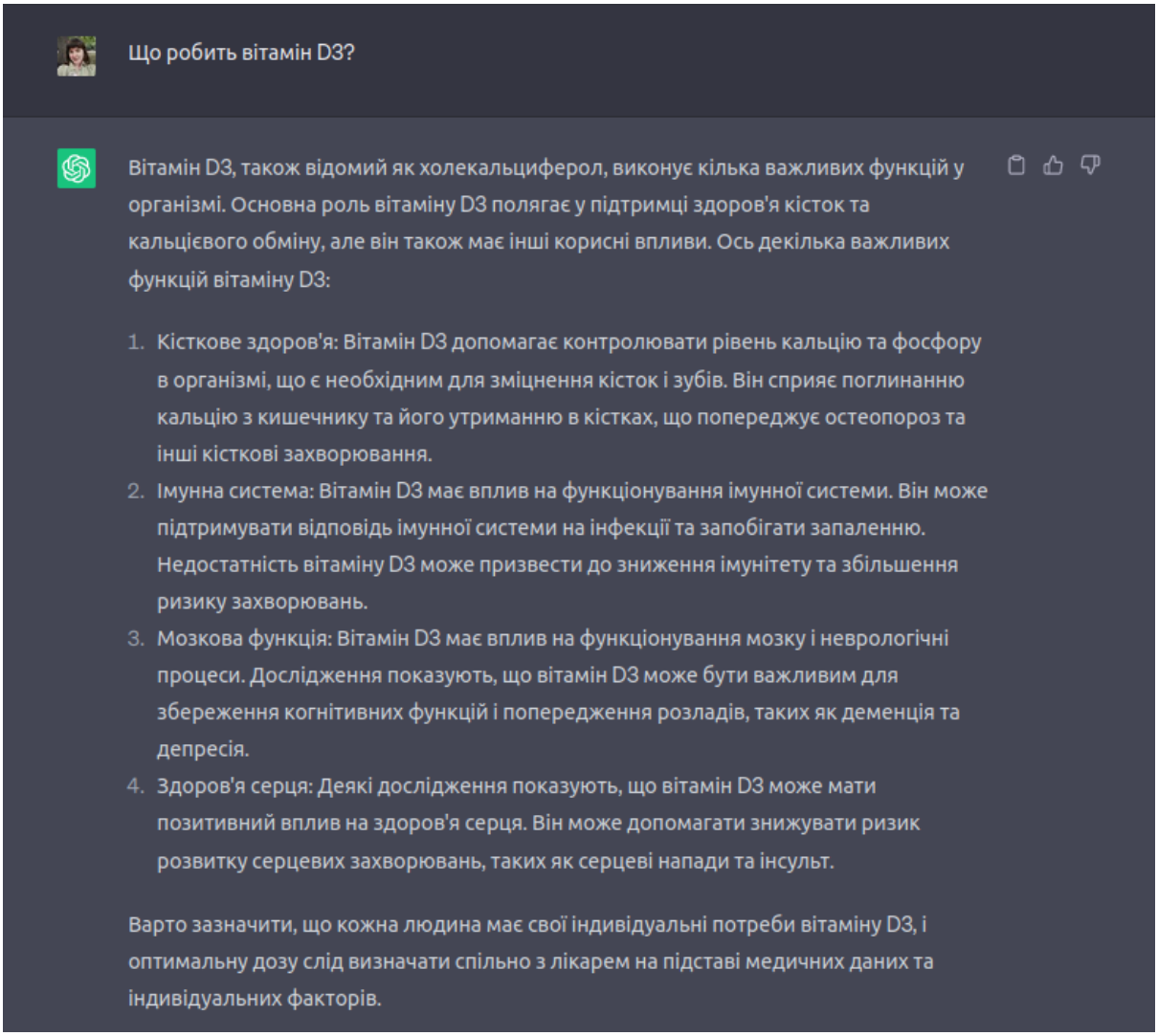

Рисунок 2.4 – Запит ChatGPT пояснити, що робить вітамін D3.

2.2.3 Med-PaLM

Med-PaLM є чат-ботом на основі штучного інтелекту, розробленим Google та DeepMind для генерації "безпечних та корисних відповідей" на запитання в галузі медицини, які ставлять медичні працівники та пацієнти. Ця модель була побудована на базі моделі PaLM з 540-ма мільярдами параметрів та шести доступних датасетів з питаннями та відповідями в галузі медицини (NedQA, MedMCQA, PubMedQA, LiveQA, MedicationQA та MMLU). Крім того, розробники створили власний HealthSearchQA, використовуючи питання, взяті

з професійних медичних іспитів, дослідженнями та запитами споживачів щодо медицини. [18]

Технологія була протестована на MultiMedQA, відкритому бенчмарку для медичних запитань та відповідей. В рамках перевірки оцінювалася фактологічність, точність, можлива шкодливість та упередженість відповідей.

2.2.3.1 Архітектура мовної моделі PaLM

Мовна модель PaLM (Pathways Language Model) використовує стандартну архітектуру моделі Transformer з декодером, що дозволяє моделі звертатися лише до себе та минулих кроків, з такими модифікаціями [19]:

- Активації SwiGLU:

Застосування активацій SwiGLU для проміжних активацій у багатошаровому перцептроні показало значні позитивні результати у порівнянні зі стандартними активаціями ReLU, GeLU та Swish [20].

SwiGLU [3] - це функція активації, що поєднує в собі переваги функцій активації Swish та Gated Linear Unit (GLU).

Swish [3] – це немонотонна функція активації, запропонована дослідниками Google у 2017 році, і визначається як:

$$Swish(x) = x * \sigma(\beta x), \tag{2.9}$$

$$\text{де } \sigma(x) = \frac{1}{1+e^{-\beta x}}, \tag{2.10}$$

β – параметр, який можна навчити.

Gated Linear Units (GLU) [21] – це функція активації, запропонована дослідниками з Microsoft у 2016 році, яка поєднує лінійну функцію з сигмоїдальною активаційною функцією:

$$GLU(x) = x * \sigma(Wx + b), \tag{2.11}$$

де W і b – параметри, які можна навчити.

GLU схожий на Swish тим, що він поєднує лінійну функцію з нелінійною функцією. Однак у GLU лінійна функція регулюється сигмоїдальною активаційною функцією.

SwiGLU — це комбінація функцій активації Swish і GLU. SwiGLU визначається наступним чином:

$$SwiGLU(x) = x * \sigma(\beta x) + (1 - \sigma(\beta x))(Wx + b), \tag{2.12}$$

де W, b і β є параметрами, які можна навчити.

Застосування активації SwiGLU для проміжних активацій у багатошаровому перцептроні показало значні позитивні результати у порівнянні з іншими стандартними функціями активації [24]. Використання SwiGLU у моделі мови PaLM дозволяє досягнути кращих результатів у генерації тексту.

- Parallel Layers

На відміну від традиційних рекурентних нейронних мереж (RNN), які обробляють вхідні дані послідовно, для опрацювання вхідних даних

використовувались паралельні операції, що дозволяє обробляти вхідні дані одночасно.

Зокрема, стандартне формулювання можна записати так:

$$y = x + MLP(LayerNorm(x + Attention(LayerNorm(x))), \quad (2.13)$$

Тоді паралельна обробка може бути записана як:

$$y = x + MLP(LayerNorm(x)) + Attention(LayerNorm(x)), \quad (2.14)$$

де y – вихідний тензор блоку Transformer, x – вхідний тензор блоку Transformer, LayerNorm(x) – Нормалізація за шарами, MLP – багатошаровий персептрон який є повнозв'язною нейронною мережею, Attention – механізм уваги. Механізм уваги дозволяє моделі навчитися встановлювати зв'язки між різними частинами вхідної послідовності, зосереджуючись на важливих елементах та ігноруючи менш важливі.

Паралельна обробка забезпечує більшу швидкість навчання на великих масштабах, оскільки множення матриць вводу MLP та Attention можуть бути об'єднані, і призводить до приблизно 15% прискорення. Механізм уваги дозволяє моделі встановлювати зв'язки між різними частинами вхідної послідовності, зосереджуючись на важливих елементах та ігноруючи менш важливі, що забезпечує досягнення кращих результатів у роботі моделі.

- Multi-Query Attention

  Це варіація механізму уваги, який дозволяє одночасно обробляти кілька запитів (queries) для кожного слова або токена. Вихідні результати уваги для кожного запиту агрегуються, а потім передаються на наступні шари

трансформера. Основна ідея multi-query attention полягає в тому, що, замість обробки одного запиту на кожному слові, модель одночасно генерує кілька запитів для кожного слова, що дозволяє краще враховувати різні аспекти контексту та відносин між токенами. Дослідники виявили, що це не впливає на якість моделі та швидкість навчання [21], але дає значне зменшення витрат часу при авторегресивному декодуванні.

- RoPE Embeddings (Rotary Position Embedding)

  Це спосіб кодування позицій слів у вхідній послідовності в нейромережі, який дозволяє моделі змінювати увагу на слова в залежності від їх відносної позиції, а не тільки на абсолютну позицію у послідовності. Для досягнення цього мети до вектору кожного слова додається вектор, який представляє синуси та косинуси різних частот, які кодують відносні позиції слів у послідовності. Цей підхід дозволяє моделі легше знаходити залежності між словами на великих вхідних послідовностях, де абсолютна позиція може бути менш важливою.

- Shared Input-Output Embeddings – для вхідних та вихідних слів використовуються однакові векторні представлення, тобто вектор, який відповідає певному слову в вхідному рядку, також використовується для представлення цього ж слова в вихідному рядку. Це дає можливість моделі краще розуміти зв'язок між вхідними та вихідними даними та поліпшує якість перекладу.
- Модель не має жодних упереджень або перекручень у своїх передбаченнях або результатах.
- Vocabulary. було використано словник SentencePiece [23] з 256 тисячами токенів, який був вибраний для підтримки великої кількості мов в навчальних корпусах без надмірної токенізації. Словник було

згенеровано з тренувальних данних, що у свою чергу покращує тренувальну ефективність.

Словник є абсолютно відновлюваним та зберігає всі пробіли (особливо важливо для кодового вводу), а символи Юнікод, які відсутні у словнику, розділяються на байти UTF-8, для кожного байта передбачено токен словника.

Набір даних для попереднього навчання PaLM складається з корпусу на 780 мільярдів токенів, які представляють широкий спектр прикладів використання природної мови. Набір даних містить фільтровані веб-сторінки, книги, сторінки Вікіпедії, новини, програмний код та розмови в соціальних мережах. Цей набір даних базується на наборах даних, використаних для навчання LaMDA [25] та GLaM [25]. Навчання моделі проводилось на рівно одній епохі даних. Таблиця 2.2 ілюструє пропорції даних для навчання моделі.

Таблиця 2.2 – Пропорції даних з кожного джерела в тренувальному наборі даних.

Джерело даних	Пропорція
Чати в соціальних мережах (різними мовами)	50%
Відфільтровані веб сторінки (різними мовами)	27%
Книги (англійською)	13%
Код з github	5%
Сторінки Вікіпедії (різними мовами)	4%
Новини (англійською)	1%

2.2.3.2 Адаптація мовної моделі до медичної галузі.

Враховуючи високу важливість безпеки в медичній сфері, модель потрібно адаптовувати та належним чином налаштовувати з доменно-орієнтованими даними. Стандартні методи передавального навчання та адаптації до домену ґрунтуються на комплексному тонкошаровому навчанні моделі, використовуючи велику кількість даних з відповідної галузі, проте такий підхід, у даному випадку, ускладнений через нестачу медичних даних.

Крім того, було виявлено, що великі мовні моделі володіють високою здатністю до швидкого навчання з малою кількістю прикладів, досягнутою через стратегії підказок (prompting). За допомогою кількох прикладів-демонстрацій, закодованих у вигляді тексту підказок у вхідному контексті, моделі здатні створювати узагальнення та нові приклади без будь-яких оновлень градієнта чи точного налаштування (fine tuning).

Для навчання моделі MedPaLM було створено новий метод оцінки клінічних знань моделей – MultiMedQA. Цей фреймворк включає сім наборів запитань з медицини, включаючи шість існуючих: MedQA, MedMCQA, PubMedQA, LiveQA, MedicationQA та MMLU clinical topics, а також новостворений HealthSearchQA, який містить загальні запитання про здоров'я. HealthSearchQA – новий набір запитань з відкритими відповідями на медичні питання, який складається з 3375 питань про загально поширені медичні проблеми, що шукаються користувачами в Інтернеті [22].

Ці набори даних відрізняються за наступними параметрами:

- Формат: питання з багатьма варіантами відповідей або довгі відповіді;
- Тестування можливостей: наприклад, оцінка запам'ятовування медичних фактів в окремості порівняно з оцінкою медичного мислення нарізі із запам'ятовуванням фактів;

- Домен: відкриті або закриті питання;
- Джерело запитань: професійні медичні іспити, медичні дослідження або запити від людей, що шукають медичну інформацію;
- Мітки та метадані: наявність міток або пояснень та їх джерела.

Під час для навчання MedPaLM, дослідники зосередилися на стратегіях ефективного налаштування, заснованих на prompting та prompt tuning. Було використано стандартні методи – few-shot, chain-of-thought and self-consistency.

Few-shot prompting передбачає надання моделі невеликого набору прикладів, зазвичай від 2 до 5, для певної задачі або проблеми. Ці приклади кодуються у вигляді input-output пар, де input – це постановка проблеми або запитання, а output – відповідь або рішення.

Chain-of-thought prompting полягає в тому, що кожний приклад з вибірки в запиті доповнюється крок за кроком розбиттям та набором проміжних кроків міркування до кінцевої відповіді. Цей підхід імітує людський процес мислення при вирішенні задач, які вимагають багатокрокових обчислень та міркувань. Успіх стратегії послідовності думок був продемонстрований на кількох наукових тестах, та ефективно вирішує складні багатокрокові задачі в медичних запитаннях.

Self-consistency prompting – полягає у створенні запитів та отриманні кількох декодованих результатів від моделі. Остаточна відповідь обирається за більшістю голосів. Основна міркування за цим підходом полягає у тому, що в такій галузі, як медицина, зі складними шляхами міркувань, може існувати кілька потенційних шляхів до правильної відповіді. Врахування всіх шляхів міркувань може привести до найбільш узгодженої відповіді. Стратегія промптінгу було використано для наборів даних з питаннями з кількома варіантами відповідей: MedQA, MedMCQA, PubMedQA та MMLU.

Prompt tuning – це ще один підхід до адаптації великих мовних моделей до конкретних задач з обмеженими даними. Підхід полягає в навчанні м'яких prompt векторів за допомогою зворотного розповсюдження, забезпечуючи незмінність решти моделі LLM. Це дозволяє легко використовувати одну модель для різних задач і, як правило, вимагає лише кількох прикладів, щоб досягти хорошої продуктивності.

Було проведено випадковий відбір прикладів з наборів даних MultiMedQA, що містять відкриті відповіді (HealthSearchQA, MedicationQA, LiveQA), за допомогою п'яти лікарів, які мають спеціалізацію в первинній допомозі, хірургії, внутрішніх хворобах та педіатрії. Лікарі відфільтрували питання/відповіді, які вони вважали не найкращими для навчання моделі, насамперед тому, що не можна надати "ідеальну" відповідь на певне запитання, якщо не відома необхідна інформація. На жаль, інформація для відповіді на запитання не завжди доступна. Таким чином, залишилися 40 прикладів з HealthSearchQA, MedicationQA та LiveQA для навчання методом instruction prompt tuning.

2.2.3.3 Ефективність Med-PaLM

Під час розробки моделі Med-PaLM було досягнуто вражаючих результатів. Зокрема, відповіді, згенеровані цією моделлю, відповідали науковому консенсусу у 92,9% випадків. Крім того, лише 6,0% відповідей Med-PaLM були визначені як потенційно шкідливі, що порівняно сприятливо порівняно з відповідями, створеними лікарями (6,5%).

У той же час, відповіді Med-PaLM проявляли здатність до розуміння в 97,5% випадків, та виявляли правильне відтворення та міркування медичних знань на рівні 95,4% та 93,5%, що знову досить близько до результатів, які можна очікувати від лікарів(97,8% та 97,7%).

Проте, Med-PaLM все ще трохи відстає від лікарських відповідей. Зокрема, 18,7% відповідей Med-PaLM містили непридатний або неправильний вміст, порівняно з 1,4% у відповідях лікарів. Також відповіді Med-PaLM містили відсутність важливих даних у 15,1% випадків, що порівняно з 11,1% у відповідях лікарів.

Незважаючи на ці невеликі недоліки, Med-PaLM все ж була корисною у більшості випадків, приносячи користь у 80,1% ситуацій. Однак, результати лікарських відповідей, що були корисними у 90,8% випадків, залишаються золотим стандартом у цій області.

Хоча новий інструмент від Google та його підрозділу штучного інтелекту DeepMind не досягнув того ж рівня, що і людина-лікар, результати його роботи були помітно кращими у порівнянні з іншими подібними моделями, які досліджувалися. Згідно зі статтею, опублікованою дослідниками цього інструменту штучного інтелекту, Med-PaLM має можливість зіграти ключову роль у клінічних застосуваннях після певного вдосконалення. Крім того, зазначено, що хоча інструмент Google працює обнадійливо, він залишається менш ефективним у порівнянні з медичними працівниками. [22]

РОЗДІЛ 3. АНАЛІЗ МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ В ПЕРСОНАЛІЗОВАНІЙ ПРЕВЕНТИВНІЙ МЕДИЦИНІ

У даному розділі представлено опис приклад високорівневої архітектури IT з використанням ChatGPT у персональній превентивній медицині. В описаній системі використовуватиметься версія модель GPT, яка буде навчена на великій кількості медичних даних і перевірена на точність та етичність, для того, щоб мінімізувати ризики нанесення шкоди користувачам.

Також при використанні даного додатку/програми/архітектури можливе залучення приладів вимірювання життєвих показників користувачів, таких як розумний годинник, датчики сну, синхронізація з датчиками в мобільному телефоні.

3.1. Загальна структура сервісу

Суть архітектури полягає в тому, щоб здійснювати індивідуальну оцінку ризиків розвитку захворювань у конкретної особи і приймати відповідні заходи з їх попередження або виявлення на ранній стадії. Враховуючи унікальні характеристики кожної людини, такі як генетичний склад, фактори ризику та спосіб життя, лікарі можуть розробляти персоналізовані рекомендації щодо діагностики, профілактики та лікування. При цьому, велика мовна модель, навчена на великій кількості медичних даних, може надавати допомогу в даному контексті, шляхом надання інформації, рекомендацій та роз'яснень на запити від людей, які цікавляться власним здоров'ям. ChatGPT може пояснити складні медичні терміни або поняття, надавати загальні поради щодо здорового способу життя, профілактики захворювань, правильного харчування, фізичної активності та інших аспектів, пов'язаних з підтримкою загального здоров'я на основі індивідуальних характеристик. Крім того, завдяки навченій моделі

можна буде дізнатись про загальні ризикові фактори розвитку захворювань і як їх уникнути за своїми, конкретними, потребами та обставинами.

В залежності від кінцевої сфери застосування, архітектура сервісу може приймати різний вигляд. Можливі рішення можна поділити на дві категорії – для комерційного застосування (наприклад, лікарні та медичні центри) та некомерційного (наприклад персональне використання у вигляді мобільного додатку).

Спосіб використання кінцевого сервісу може дещо накладати обмеження на архітектуру. Для комерційних рішень дійовими особами є лікар та його пацієнт. При цьому, у пацієнта буде можливість придбати додаткове устаткування, яке вимірюватиме його показники.

В основному, взаємодія медичних працівників відбуватиметься через браузерний клієнт, куди вони вноситимуть дані своїх клієнтів після оглядів або отримання аналізів, а також переглядати інформацію і статус. Статус стану людини буде ідентифікований за допомогою ChatGPT та окремого сервісу, на основі запитів користувача про симптоми, та показників, які були отримані від клієнта(або вручну введені ним, або зчитані з датчиків), щоб лікар бачив, чи не з’являються у його підопічного якісь ризики захворювань чи погіршення загального стану. У разі виявлення чогось подібного буде можливість назначити прийом / проконсультувати пацієнта, щоб надати клієнту рекомендації щодо покращення стану його самопочуття, уникнення ускладнень, а також перевіряти та корегувати рівень ризику погіршення стану здоров’я кожного з пацієнтів.

Клієнт(пацієнт) напряму взаємодіятиме з мобільним додатком та, при можливості, зі спеціальними вимірювальним приладом(який буде універсальним збірником показників життєдіяльності). Мобільний застосунок здійснюватиме обробку даних, отриманих з пристроїв користувачів, і відправлятиме їх на сервера та зберігатиме у базі даних. Після виміру

показників, або просто заходячи у додаток, клієнт матиме змогу поспілкуватись з чат ботом - ChatGPT(на новій розробленій медичній моделі), який оцінить його стан здоров’я, дасть рекомендації щодо профілактики, і надасть усю інформацію про ризики. Якщо ж буде виявлено якісь ризики, надасть пацієнту рекомендацію звернутись до лікаря за додатковою консультацією.

Основні незмінні дані користувача, такі як історія хвороби, група крові, які вже знаходяться в базі також будуть відправлятися в повідомленні, після якого буде оцінено стан користувача.

Крім того, клієнт зможе вручну вносити свої показники, і якщо вони не викликатимуть занепокоєнь при обробленні(класифікацією) просто вийти з додатку, якщо ж будуть, чат автоматично надішле повідомлення про це.

На рис. 3.1 зображено приблизну архітектуру такого сервісу.

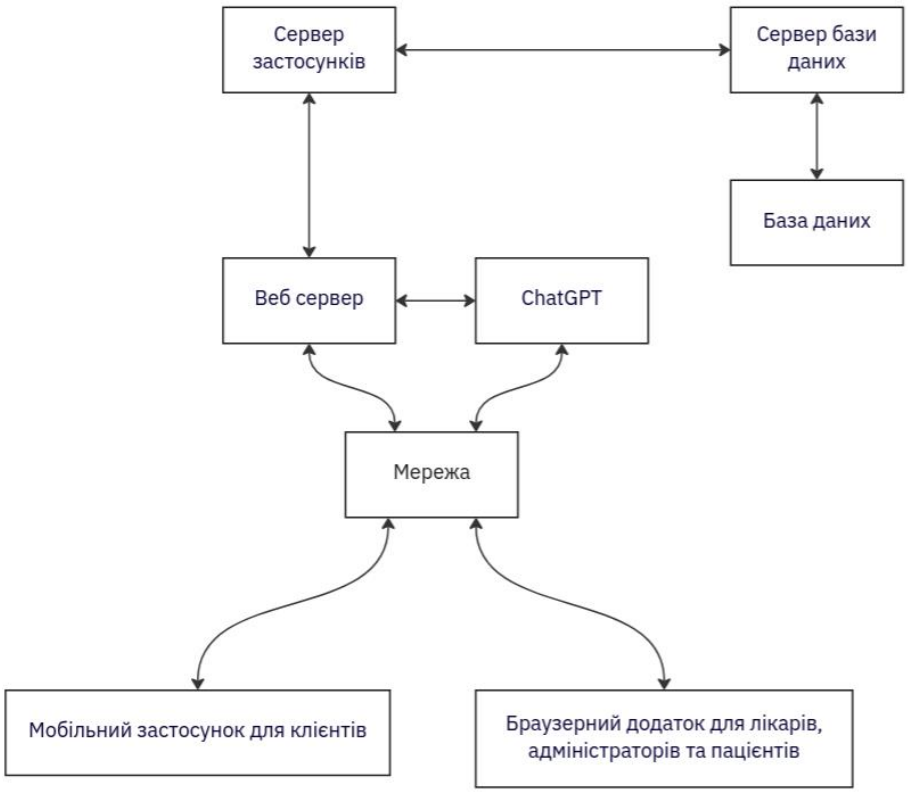

Рис. 3.1 – Високорівнева архітектура сервісу

3.2 Типи користувачів та сценарії використання

У системі матимуть місце 3 дійові особи: клієнт (або пацієнт), лікар та адміністратор.

Клієнт. Може бути як користувачем, закріпленим за якимсь медичним закладом, так і тим, хто користується додатком без професіонального нагляду. Він має встановити мобільний додаток на свій смартфон та зареєструватися в системі, налаштувати додаток на збір показників зі своїх вимірювальних пристроїв. Крім того, заповнити форму про стан свого здоров’я та певні константні показники. Протягом дня додаток автоматично збиратиме дані про наприклад серцевий ритм, кров'яний тиск та інші показники з підключених пристроїв. Періодично, за запитом, клієнт зможе оновлювати рекомендацій та оцінки свого стану здоров'я, наданих ChatGPT. У разі виявлення змін у стані здоров'я або підвищеного ризику, ChatGPT надішле користувачу повідомлення про рекоменацію звернутися до лікаря. Крім того, користувач має змогу “проконсультуватись” з ботом щодо стану свого здоров’я. Є дві опції - відкритий і закритий чат. Різниця тільки в тому, що відкритий чат матиме змогу також переглядати лікар.

Лікар. Лікар може бути як приватний, так і представником державного медичного закладу. Він отримує доступ до системи через браузерний клієнт та авторизується під своїм обліковим записом(якщо від закладу), і реєструється як лікар, якщо приватний. Обов’язковим кроком для приватних медичних працівників є підтвердження кваліфікації. Лікар може переглядати список своїх пацієнтів та їх медичні дані, які були зібрані мобільним додатком. Лікар може аналізувати дані про пульс, кров'яний тиск та інші показники, що надійшли від пацієнтів. Також він може аналізувати відкритий чат свого пацієнта, отримувати нотифікації від серверу, якщо ChatGPT виявить потенційні ризики. У випадку виявлення небажаних змін або підозри про ризики погіршення стану здоров'я, лікар може звернутися до пацієнта для уточнення і надання

рекомендацій. Лікар також може оновлювати статуси пацієнтів у системі та додавати інформацію до історії їх станів та хвороб.

Адміністратор. Адміністратор матиме доступ до системи через адміністративний інтерфейс. Він керуватиме правами доступу лікарів та користувачів, створюватиме облікові записи для нових лікарів та деактивовуватиме застарілі облікові записи. Адміністратор також забезпечує передачу медичних даних сервісам з перевірки кваліфікації лікарів.

3.3 Сервіси системи

Прогнозується залучення таких сервісів:

- Сервіс збору показників – відповідатиме за збір та обробку показників життєдіяльності користувача, може отримувати дані, як з телефону (синхронізація з датчиками), так і підключатися до зовнішніх приладів, які використовує користувач для виміру. Приведені дані, відправлятимуться на головний сервер та зберігатимуться у базі даних, ці дані в подальшому використовуватимуться ChatGPT, та будуть передані лікарю через API запит до сервера.
- Чат-бот – отримує дані пацієнта з сервера за допомогою API. На основі цих даних, спілкується з користувачем, і крім того, при виявлені ризиків, надсилає повідомлення лікарю.
- Сервіс повідомлень – відповідатиме за обробку повідомлень ChatGPT, логістику надсилання до лікаря та користувача. Дані відкритих чатів надсилатиме на сервер, і вони також зберігатимуться в базі даних.
- Сервер баз даних – відповідає за збереження всіх даних отриманих від клієнтів, а також статистику та оброблені результати діагностики від лікарів.
- Сервіс обробки показників – сервіс, який відповідатиме за обробку та аналіз даних користувачів, які отримуватимуться з бази даних. Дані

будуть оброблені та приведені до вигляду, в якому лікар зможе їх проглянути і зробити висновки щодо стану пацієнта.

- Сервіс перевірки кваліфікації - відповідатиме за обробку документів, які були надіслані лікарем.

3.4 Діаграма послідовностей та взаємодія основних сутностей сервісу

Розглянемо детальніше, як саме працює сервіс і які взаємодії відбуваються між його основними елементами. На діаграмі послідовностей нижче зображено взаємодію користувачів і лікарів з сервісом.

Користувач має можливість самостійно або за рекомендацією лікаря встановити додаток на свій пристрій. Після успішної реєстрації в сервісі, якщо користувач не був зареєстрований раніше, система створює йому ідентифікатор користувача. Цей ідентифікатор буде використовуватися для зберігання всіх зібраних даних, показників, історії хвороб, загальної інформації про користувача, а також для зберігання чатів з ChatGPT та лікарем.

При вході в систему користувач вводить свій логін та пароль, інформація надсилається на сервер, який перевіряє правильність цих даних для облікового запису користувача. Якщо дані є вірними, створюється нова сесія, в рамках якої будуть зберігатися моніторингові дані.

Після цього на фоні запускається синхронізація з датчиками пристроїв користувача. Отримані дані оброблятимуться і надсилатимуться на сервер, де відбуватиметься подальший аналіз. У випадку відсутності зв'язку з сервером, дані можуть зберігатися в локальній базі даних і в момент відновлення зв'язку синхронізуватися з сервером.

Користувач може переглядати статистику зміни своїх показників, доповнювати їх, а також запросити детальний звіт у системи.

У випадку, якщо ChatGPT, аналізуючи дані, виявить значення поза допустимими межами, то автоматично, залежно від прописаних правил, системою буде згенерована подія для оповіщення користувача, та/або його лікаря.

Лікар матиме змогу переглянути дані пацієнта та прийняти рішення про те, настільки критичний стан був знайдений системою. Крім того, йому будуть надсилатися повідомлення у випадку виявлення ChatGPT ризиків. Також лікар може надати рекомендації, які будуть показані користувачу.

Користувач може обмінюватись повідомленнями з ChatGPT, відкриті чати також надсилатимуться до серверу, а також будуть видимі для лікаря.

Коли користувач завершить роботу з пристроєм, він може вийти(logout) з системи. При цьому, зв’язок з датчиками перерветься та зупиниться аналіз стану користувача та чат в ChatGPT.

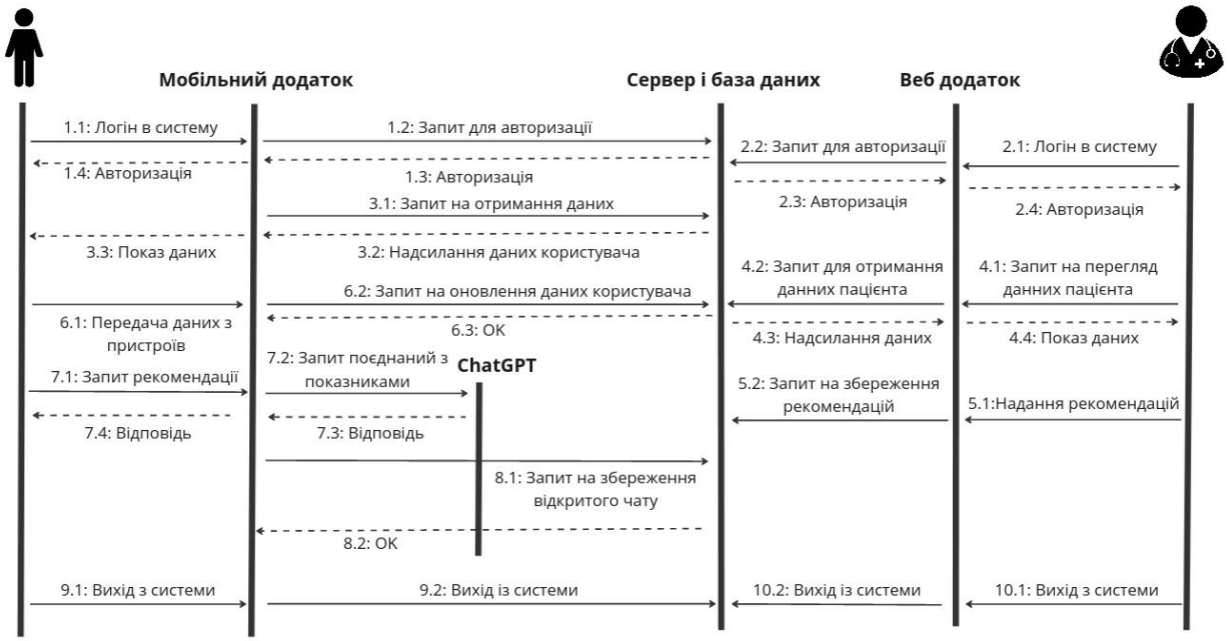

Рисунок 3.2 – Діаграма послідовності взаємодії користувачів та лікарів з сервісом

РОЗДІЛ 4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В даному розділі буде проведено оцінку основних характеристик для майбутнього програмного продукту, що спеціалізується на аналізі показників здоров’я людей і надання рекомендацій щодо персоналізованої превентивної медицини.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи рекомендацій для персональної превентивної медицини. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних користувачів.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- функціонування на смартфонах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який надає рекомендації та аналізує показники здоров'я користувачів. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – аналіз медичних даних.

F_3 – вибір середовища програмуванн.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) JavaScript

б) Python

Функція F_2:

а) Застосування вбудованих функцій.

б) Створення своїх обчислювальних методів.

Функція F_3:

а) Visual Studio Code

б) PyCharm.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

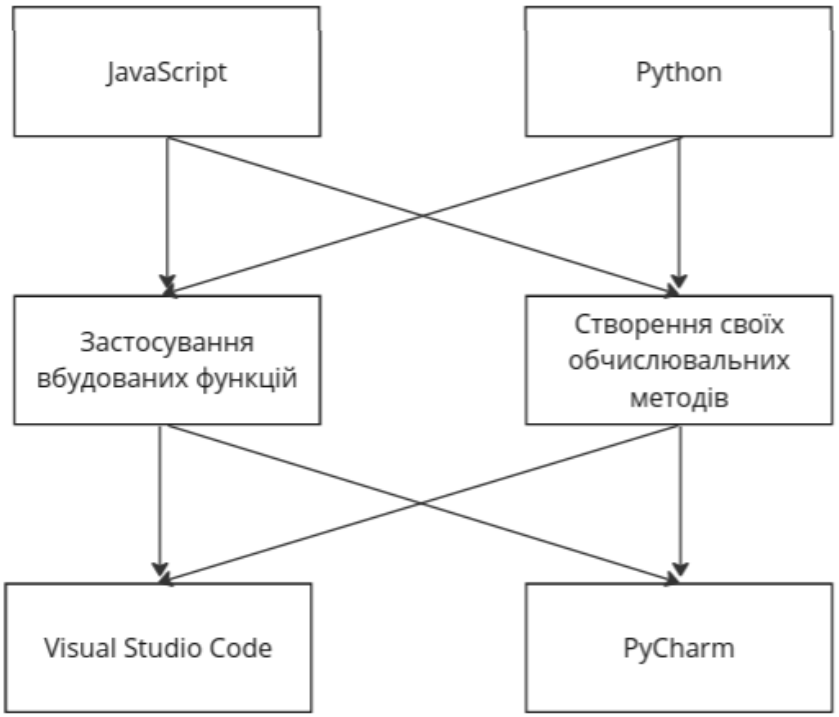

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Широко підтримується в браузерах та веб-розробці, велика кількість доступних бібліотек та фреймворків; асинхронна модель програмування	Відсутність стандартної бібліотеки для роботи з мережею; менша швидкодія порівняно з мовами, що компілюються; динамічна типізація
	Б	Велика спільнота розробників та багата документація; висока швидкодія та ефективність; підтримка об'єктно-орієнтованого та функціонального підходів	Недостатнє підтримка у веб-розробці, відсутність вбудованої підтримки для браузера та фронтенду
F_2	А	Швидше впровадження засобів аналізу даних; можливість використання перевірених рішень; менше витрат часу та зусиль на розробку та реалізацію	Обмеження у використанні функціональності; відсутність гнучкості у внесенні змін та модифікації; залежність від виробника

Продовження таблиці 4.1

Функції	Варіанти реалізації	Переваги	Недоліки
F_2	Б	Гнучкість у створенні власних методів, відповідно до потреб; більший контроль над процесом обробки та аналізу даних; можливість налаштування методів під конкретні медичні потреби.	Високий рівень експертизи та знань для створення власних методів; потреба у додаткових ресурсах для розробки та підтримки; можливі проблеми з оптимізацією та продуктивністю.
F_3	А	Легкий та швидкий редактор коду; велика кількість розширень та плагінів; широка підтримка різних мов програмування; інтеграція з інструментами контролю версій (наприклад, Git).	Обмежені можливості порівняно з повноцінною IDE; відсутність деяких вбудованих функцій IDE, таких як рефакторинг коду; менша кількість автоматичних інструментів для виявлення помилок.
	Б	Повноцінне інтегроване середовище розробки (IDE); потужний вбудований дебагер та інструменти профілювання; зручне управління віртуальними середовищами та залежностями.	Вимагає більш високих системних ресурсів; ліцензійна модель може бути витратною для комерційного використання.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевага надається полегшенню веб-розробки і широкому вибору бібліотек. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм пам’яті для обчислень та збереження даних;

- X3 – час навчання даних;
- X4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	5000	9000	18000
Об’єм пам’яті	X2	Мб	564	256	128
Час попередньої обробки даних	X3	мс	2000	900	500
Потенційний об’єм програмного коду	X4	кількість рядків коду	10000	7000	4000

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

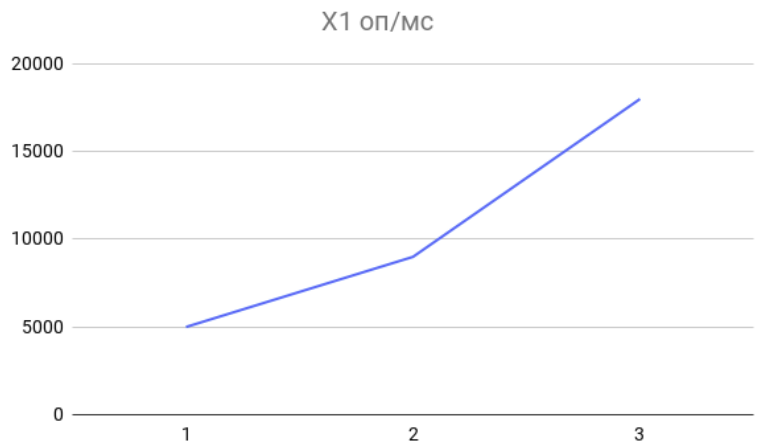

Рисунок 4.2 – X1, швидкодія мови програмування

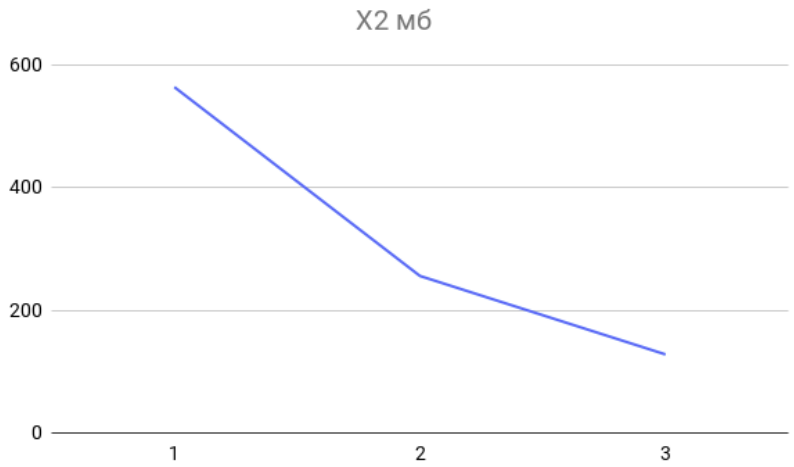

Рисунок 4.3 – X2, об’єм пам’яті

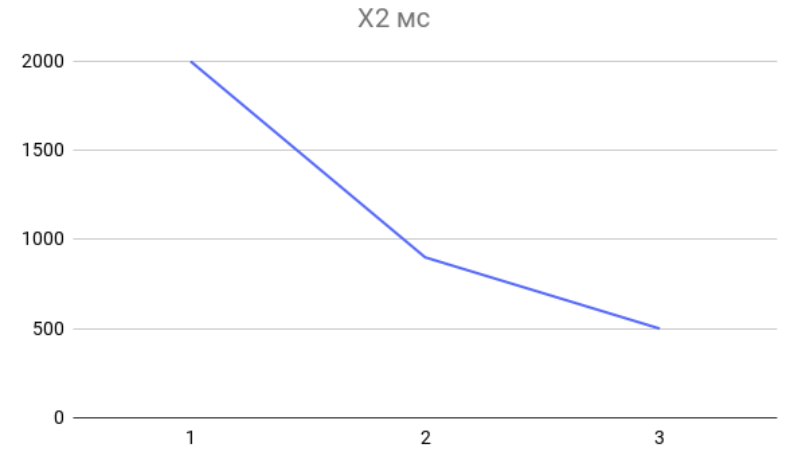

Рисунок 4.4 – X3, час попередньої обробки даних

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порів-няння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значущості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	5	6	7	2	6	7	4	37	12.5	156.25
X2	Об’єм пам’яті	Mb	3	4	3	2	2	2	7	23	-1.5	2.25

Продовження таблиці 4.3

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X3	Час попередньої обробки даних	мс	3	2	3	2	1	4	2	17	-7.5	56.25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	3	2	1	8	5	1	1	21	-3.5	12.25
	Разом		14	14	14	14	14	14	14	98	0	227

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 98. \quad (4.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 24.5. \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 227. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 227}{7^2(4^3-4)} = 0{,}93 > W_k = 0{,}67, \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	=	>	>	<	>	1.5
X1 і X3	>	>	>	=	>	>	>	>	1.5
X1 і X4	>	>	>	<	>	>	>	>	1.5
X2 і X3	=	>	=	=	>	<	>	=	1.0
X2 і X4	=	>	>	<	<	>	>	>	1.5
X3 і X4	=	=	>	<	<	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*-го параметра над *j*-тим, a_{ij} визначається по формулі:

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \; 1.0 \text{ при } X_i = X_j \; 0.5 \text{ при } X_i < X_j \;, \tag{4.6}$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \tag{4.7}$$

$$b_i = \sum_{i=1}^{N} a_{ij}, \tag{4.8}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j, \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1	1.5	1.5	1.5	5.5	0.34	21.25	0.36	78.625	0.36
X2	0.5	1	1	1.5	4	0.25	14.5	0.24	53.5	0.24
X3	0.5	1.0	1	1.5	4	0.25	14.5	0.24	53.5	0.24
X4	0.5	0.5	0.5	1	2.5	0.16	9.25	0.16	34.375	0.16
Всього:					16	1	59.5	1	220	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2 (Об’єм пам’яті), X3 (час попередньої обробки даних) та X4 (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \qquad (4.11)$$

де n – кількість параметрів;

$K_{\text{в}i}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Парамет ри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	9000	20	0.36	7.2
F3	A	X2	128	17	0.24	4.08

Продовження таблиці 4.6

Основні функції	Варіант реалізації функції	Парамет ри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F3	Б	X3	256	16	0.24	3.84
F4	А	X4	4000	13	0.16	2.08

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 7{,}2 + 4{,}08 + 2{,}08 = 13{,}36;$$

$$K_{K2} = 7.2 + 3{,}84 + 2{,}08 = 13{,}12.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (4.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ = 22 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.5. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 22 \cdot 1.5 \cdot 0.9 = 29,7 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р = 95$ людино-днів, $К_П = 1.3$, $К_{СК} = 1$, $К_{СТ} = 0.9$:

$$T_2 = 95 \cdot 1.3 \cdot 0.9 = 111.15 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (29,7 + 111.15 + 4.7 + 111.15) \cdot 8 = 2053.6 \text{ людино-годин.}$$
$$T_{II} = (29,7 + 111.15 + 7.11 + 111.15) \cdot 8 = 2072.88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь п’ять програмістів з окладом 35510 грн., один аналітик в області даних з окладом 41000. Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{грн.} \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{(35510 * 5) + 41000}{6 \cdot 21 \cdot 8} = 216{,}82 \text{ грн.} \quad (4.15)$$

Тоді, розраховуємо заробітну плату за формулою:

$$СЗП = С_ч \cdot Т_i \cdot КД, \quad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 216.82 \cdot 2053.6 \cdot 1.2 = 534\ 313.86$ грн.

II. $С_{ЗП} = 216.82 \cdot 2072.88 \cdot 1.2 = 539\ 330.21$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 534\ 313.86 \cdot 0.22 = 117\ 549.05$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 539\ 330.21 \cdot 0.22 = 118\ 652.65$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 35510 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 35510 \cdot 0{,}2 = 85\ 224 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 85\ 224 \cdot (1 + 0.2) = 102\ 268.8 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД}= С_{ЗП} \cdot 0.22 = 102\ 268.8 \cdot 0{,}22 = 22\ 499.14 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 27000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.2 \cdot 0.25 \cdot 27000 = 8100 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.2 \cdot 27000 \cdot 0.05 = 1620 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 11) \cdot 8 \cdot 0.8 =$$

$$= 1491.2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1491.2 \cdot 0,5 \cdot 0,2 \cdot 4,86 = 724.72 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 27000 \cdot 0,67 = 18090 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$$С_{ЕКС} = 102\ 268.8 + 22\ 499.14 + 8100 + 1620 + 724.72 + 18090 =$$

$$= 153\ 302.66 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 153\ 302.66 \ / \ 1491.2 = 102.8 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М-Г} \cdot T, \quad (4.18)$$

I. $C_М = 102.8 \cdot 2053.6 = 211\ 110.08$ грн.

II. $C_М = 102.8 \cdot 2072.88 = 213\ 092.06$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0.67, \quad (4.19)$$

I. $C_Н = 534\ 313.86 \cdot 0{,}67 = 357\ 990.29$ грн.

II. $C_Н = 539\ 330.21 \cdot 0{,}67 = 361\ 351.24$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \quad (4.20)$$

I. $C_{ПП} = 534\ 313.86 + 117\ 549.05 + 211\ 110.08 + 357\ 990.29 =$

$= 1\ 220\ 963.28$ грн.

II. $C_{ПП} = 539\ 330.21 + 118\ 652.65 + 213\ 092.06 + 361\ 351.24 =$

$= 1\ 232\ 426.16$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{К j} \ / \ С_{Ф j}, \quad (4.21)$$

$$К_{ТЕР1} = 13{,}36 / 1\ 220\ 963.28 = 1.0942 \cdot 10^{-5},$$
$$К_{ТЕР2} = 13{,}12 / 1\ 232\ 426.16 = 1.0645 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1.0942 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу, що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1.0942 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- Мова програмування – JavaScript;
- Застосування вбудованих функцій;
- Visual Studio Code

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку та зручну реалізацію програми, доступний функціонал для роботи.

4.8 Висновки до четвертого розділу

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу, що розробляється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

Превентивна медицина відіграє критичну роль у забезпеченні здорового суспільства. Застосування стратегій профілактики та раннього виявлення захворювань може значно знизити ризики розвитку серйозних хвороб та покращити загальний стан здоров'я населення.

У той же час, використання великих мовних моделей у медичних дослідженнях та практиці виявляє великий потенціал. Ці моделі здатні аналізувати великі обсяги медичної інформації, розпізнавати патерни та виконувати складні завдання діагностики та рекомендацій з профілактики захворювань, забезпечуючи індивідуальний підхід до кожного пацієнта, враховуючи його особливості, історію захворювань та інші фактори.

У рамках дослідження було визначено поняття персональної превентивної медицини, проаналізовано основні поняття, пов’язані з великими мовними моделями, розглянуто їх структуру та принципи роботи. Детально розглянуто BMM – GPT та Med-PaLM. Крім того, було досліджено архітектуру Transformer, яка зараз все більше набирає популярності у світі штучного інтелекту, описано механізм самоуваги.

У третьому розділі було представлено приклад архітектури додатку для медичних рекомендацій та аналізу показників, який має на меті допомогти людям слідкувати за своїм здоров'ям і надавати персоналізовані рекомендації на основі їх даних. Основна ідея полягає в тому, щоб забезпечити користувачам зручний і ефективний інструмент для моніторингу їх стану здоров'я та підтримки їх загального благополуччя.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hood L. A personal view on systems medicine and the emergence of proactive P4 medicine: predictive, preventive, personalized and participatory [Electronic resource] / Leroy Hood, Mauricio Flores // New Biotechnology. – 2012. – Vol. 29, no. 6. – P. 613–624. – Mode of access: https://doi.org/10.1016/j.nbt.2012.03.004. – Title from screen.
2. Sergey Suchkov. Predictive, Preventive and Personalized Medicine & Molecular Diagnostics [Electronic resource] / Sergey Suchkov // Las Vegas, 3 November 2014. – [S. l.]. – Mode of access: https://www.walshmedicalmedia.com/conference-abstracts-files/2153-0645.S1.001-004.pdf. – Title from screen.
3. Overview of artificial intelligence in medicine [Electronic resource] / Amisha [et al.] // Journal of Family Medicine and Primary Care. – 2019. – Vol. 8, no. 7. – P. 2328. – Mode of access: https://doi.org/10.4103/jfmpc.jfmpc_440_19. – Title from screen.
4. Davenport T. The potential for artificial intelligence in healthcare [Electronic resource] / Thomas Davenport, Ravi Kalakota // Future Healthcare Journal. – 2019. – Vol. 6, no. 2. – P. 94–98. – Mode of access: https://doi.org/10.7861/futurehosp.6-2-94. – Title from screen.
5. Attention Is All You Need [Electronic resource] / Ashish Vaswani [et al.]. – 2017. – P. 1–15. – Mode of access: https://doi.org/10.48550/arXiv.1706.03762. – Title from screen.
6. Angie Lee. What Are Large Language Models Used For? [Electronic resource] / Angie Lee. – Mode of access: https://blogs.nvidia.com/blog/2023/01/26/what-are-large-language-models-used-for/. – Title from screen.
7. Jacob Devlin [et al.] // Proceedings of the 2019 Conference of the North, Minneapolis, Minnesota. – Stroudsburg, PA, USA, 2019. – Mode of access: https://doi.org/10.18653/v1/n19-1423 – Title from screen.

8. Language Models are Few-Shot Learners [Electronic resource] / Tom B. Brown [et al.]. – 2022. – Mode of access: https://doi.org/10.48550/arXiv.2005.14165. – Title from screen.
9. Improving Language Understanding by Generative Pre-Training [Electronic resource] / Alec Radford [et al.]. – 2018. – Mode of access: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf. – Title from screen.
10. Transfer Learning in Natural Language Processing [Electronic resource] / Sebastian Ruder [et al.] // Proceedings of the 2019 Conference of the North, Minneapolis, Minnesota. – Stroudsburg, PA, USA, 2019. – Mode of access: https://doi.org/10.18653/v1/n19-5004. – Title from screen.
11. Deep Contextualized Word Representations [Electronic resource] / Matthew Peters [et al.] // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, Louisiana. – Stroudsburg, PA, USA, 2018. – Mode of access: https://doi.org/10.18653/v1/n18-1202. – Title from screen.
12. Andrej Karpathy. A Recipe for Training Neural Networks [Electronic resource] / Andrej Karpathy. – Mode of access: https://karpathy.github.io/2019/04/25/recipe/. – Title from screen.
13. Howard J. Universal Language Model Fine-tuning for Text Classification [Electronic resource] / Jeremy Howard, Sebastian Ruder // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia. – Stroudsburg, PA, USA, 2018. – Mode of access: https://doi.org/10.18653/v1/p18-1031. – Title from screen.
14. Dario Amodei. AI and compute [Electronic resource] / Dario Amodei, Danny Hernandez. – Mode of access: https://openai.com/research/ai-and-compute. – Title from screen.

15. Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview [Electronic resource] / Robin M. Schmidt. – 2019. – Mode of access: https://doi.org/10.48550/arXiv.1912.05911. – Title from screen.
16. Ralf C. Staudemeyer. – Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Network [Electronic resource] / Ralf C. Staudemeyer, Eric Rothstein Morris. – 2019. – Mode of access: https://doi.org/10.48550/arXiv.1909.09586. – Title from screen.
17. OpenAI. GPT-4 Technical Report [Electronic resource] / OpenAI. – 2023. – Mode of access: https://doi.org/10.48550/arXiv.2303.08774. – Title from screen.
18. Large Language Models Encode Clinical Knowledge [Electronic resource] / Karan Singhal [et al.]. – 2022. – Mode of access: https://doi.org/10.48550/arXiv.2212.13138. – Title from screen.
19. PaLM: Scaling Language Modeling with Pathways [Electronic resource] / Aakanksha Chowdhery [et al.]. – 2022. – Mode of access: https://doi.org/10.48550/arXiv.2204.02311. – Title from screen.
20. Noam Shazeer. GLU Variants Improve Transformer [Electronic resource] / Noam Shazeer. – 2020. – Mode of access: https://doi.org/10.48550/arXiv.2002.05202. – Title from screen.
21. Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need [Electronic resource] / Noam Shazeer. – 2019. – Mode of access: https://doi.org/10.48550/arXiv.1911.02150. – Title from screen.
22. RoFormer: Enhanced Transformer with Rotary Position Embedding [Electronic resource] / Jianlin Su [et al.]. – 2021. – Mode of access: https://doi.org/10.48550/arXiv.2104.09864. – Title from screen.
23. Kudo T. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing [Electronic resource] / Taku Kudo, John Richardson // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations,

Brussels, Belgium. – Stroudsburg, PA, USA, 2018. – Mode of access: https://doi.org/10.18653/v1/d18-2012. – Title from screen.

24. LaMDA: Language Models for Dialog Applications [Electronic resource] / Romal Thoppilan [et al.]. – 2022. – Mode of access: https://doi.org/10.48550/arXiv.2201.08239. – Title from screen.
25. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts [Electronic resource] / Nan Du [et al.]. – 2021. – Mode of access: https://doi.org/10.48550/arXiv.2112.06905. – Title from screen.