Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістра
(освітній ступінь))

на тему: **Методи та засоби створення і розгортання інфраструктури комп'ютерних систем за допомогою генеративних алгоритмів машинного навчання**

Виконав: студент (ка) 6 курсу, групи СІм-61
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

	(підпис)	Островський А.Я. (прізвище та ініціали)
Керівник	(підпис)	Луцків А.М. (прізвище та ініціали)
Нормоконтроль	(підпис)	Луцик Н.С. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Осухівська Г.М. (прізвище та ініціали)
Рецензент	(підпис)	Мудрик І.Я. (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра комп'ютерних систем та мереж

ЗАТВЕРДЖУЮ
Завідувач кафедри *Осухівська Г.М.*

«_____» 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Островському Андрію Ярославовичу*
(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) *Методи та засоби створення і розгортання інфраструктури комп'ютерних систем за допомогою генеративних алгоритмів машинного навчання*

Керівник проекту (роботи) *Луцків Андрій Мирославович, к.т.н., доц.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*01*» *грудня* 2023 року №*4/7-1132*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *Основні поняття великих моделей даних, типи апаратного і програмного забезпечення в генеративних моделях, критерії ефективності LLM*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз сучасного стану розвитку та організації великих даних
2. Критерії та методи забезпечення оптимальності функціонування великих мовних моделей
3. Організація експериментів застосування великих мовних моделей з генеративними
4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета дослідження. 2. Задачі, об'єкт і предмет, наукова новизна і практична цінність дослідження. 3. Властивості великих даних. 4. Життєвий цикл великих даних. 5. Архітектура трансформерів. 6. Метод оцінювання інфраструктури LLM моделей 7. Результати оцінювання продуктивності інфраструктури LLM моделей. 8. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	*Осухівська Г.М., зав. каф. КС*		
безпека в надзвичайних			
ситуаціях	*Стадник І.Я., проф. каф ОХ*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз сучасного стану розвитку та організації*	*01.12.2023-*	*виконано*
	великих даних	*05.12.2023*	
2.	*Критерії та методи забезпечення оптимальності*	*05.12.2023-*	*виконано*
	функціонування великих мовних моделей	*12.12.2023*	
3.	*Організація експериментів застосування великих*	*12.12.2023-*	*виконано*
	мовних моделей з генеративними алгоритмами	*17.12.2023*	
4.	*Охорона праці та безпека в надзвичайних ситуаціях*	*18.12.2023*	*виконано*
5.	*Оформлення пояснювальної записки*	*20.12.2023*	*виконано*
6.	*Оформлення графічного матеріалу*	*21.12.2023*	*виконано*
7.	*Попередній захист кваліфікаційної роботи магістра*	*22.12.2023*	*виконано*
8.	*Захист кваліфікаційної роботи магістра*		

Студент _____ *Островський А.Я.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Луцків А.М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи та засоби створення і розгортання інфраструктури комп'ютерних систем за допомогою генеративних алгоритмів машинного навчання // Кваліфікаційна робота магістра// Островський Андрій Ярославович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем та програмної інженерії, група СІм-61 // Тернопіль, 2023 // с. – 85 , рис. – 32 , табл. – 13 , аркушів А1 –8 , додат. – 1, бібліогр. – 25.

Ключові слова: метод, засіб, інфраструктура, алгоритм, розгортання, машинне навчання.

У кваліфікаційній роботі магістра проаналізовано життєвий цикл великих даних та визначено особливості етапів і процесів, характерних при опрацюванні великих даних. Визначено базові концепції великих мовних моделей з реалізацією генеративних алгоритмів і встановлено, що вони які дають змогу досліджувати статистичні шаблони, граматику та семантику людської мови та використовують потужні набори даних і відповідно значні апаратно-програмні ресурси.

Запропоновано критерії продуктивності у процесі попереднього опрацювання запитів до моделей з генеративними алгоритмами: час формування першого токену, час генерації вихідного токену для кожного користувача, прихована затримка формування відповіді користувачам та пропускної здатності моделі. Розроблено метод та показники оптимізації використання ресурсів пам’яті та графічних процесорних ядер при формуванні висновків (відповідей) великими мовними моделями. Програмно імплементовано механізми прямого доступу до моделей з генеративними алгоритмами машинного навчання та експериментально доведено їх ефективність.

ABSTRACT

Methods and tools for creating and deploying computer system infrastructures using generative machine learning algorithms /Master’s graduation thesis / Ostrivskyi Andrii/ Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and software engineering, group CIm -61 // Ternopil, 2023// p. - 85, fig. – 32, table. – 13, Sheets A1 – 8, Add – 1, Ref. – 25.

Keywords: method, tool, infrastructure, algorithm, deployment, machine learning.

The Master's thesis analyzed the life cycle of big data and identified the features of the stages and processes characteristic of big data processing. The basic concepts of large language models with the implementation of generative algorithms are defined and it is established that they enable the study of statistical patterns, grammar and semantics of human language and use powerful data sets and, accordingly, significant hardware and software resources.

Performance criteria in the process of pre-processing requests to models with generative algorithms are proposed: the time of generating the first token, the time of generating the initial token for each user, the hidden delay of generating a response to users and the bandwidth of the model.

The method and indicators for optimizing the use of memory resources and graphic processor cores when forming conclusions (answers) using large language models have been developed. Mechanisms of direct access to models with generative machine learning algorithms were implemented in software and their effectiveness was experimentally proven.

ЗМІСТ

ВСТУП

Актуальність теми. Сьогодні застосування нових методів і засобів проектування, реалізації та підтримки програмно-апаратних платформ дають змогу значно розширити та автоматизувати процеси управління бізнесом у різних сферах. Окрім цього, вони породжують нові та розвивають існуючі системи аналітики даних на основі підходів і методів штучного інтелекту. В останні роки сфера штучного інтелекту (ШІ) стала свідком трансформаційного розвитку з появою великих (Large) мовних (Language) моделей (Models) (LLM). Ці моделі, наприклад GPT різних версій від OpenAI, є нейронними мережами з мільйонами або навіть мільярдами параметрів, призначених для розуміння та створення людської мови.

LLM привернули величезну увагу завдяки своїм надзвичайним можливостям, і здатністю формувати майбутнє програм штучного інтелекту в різних областях. Також вони відіграють важливу роль і глибоко впливають на розвиток технологій, бізнесу і суспільства в цілому.

Однак, при побудові та використанні великих мовних моделей постає проблема щодо раціонального та ефективного використання наявних апаратних ресурсів, можливості паралельного і розподіленого опрацювання великих масивів інформації, налаштуванні мільйонів або навіть мільярдів гіперпараметрів алгоритмів глибоких нейронних мереж та ряду інших.

Оскільки, напрям щодо дослідження великих мовних моделей та генеративних алгоритмів сьогодні інтенсивно розвивається і не всі деталі вивчено у повній мірі, то актуальними задачами, які доцільно було б розв’язати в рамках комп’ютерної інженерії та штучного інтелекту, полягають у розробці методів і засобів організації та оцінювання ефективності програмно-апаратної інфраструктури при розширенні цих моделей, їх донавчанню та оцінюванні результатів роботи LLM моделей.

Оскільки, великі мовні моделі передбачають опрацювання гігабайтів чи навіть терабайтів даних, то важливим в цьому контексті є також дослідження у сфері великих даних (Big Data) та їх одержанні та опрацюванні (Data Engineering).

В основному праці щодо дослідження методів і засобів організації великих даних та їх опрацювання присвячені багатьма закордонними вченими (D. Pfeffermann, P. T. Endo, M. Rodrigues, G. E. Goncalve та ін.), хоча і є невелика кількість українських науковців, які працюють у цьому сегменті ІТ (Луцків А.М., Саріогло В. Г., Кислова О. та ін.).

Однак у цих дослідженнях мало уваги приділено аспектам організації програмно-апаратних інфраструктур та способів інтеграції нових даних для навчання існуючих моделей. Тому актуальними задачами при практичній імплементації LLM та генеративних алгоритмів є вдосконалення або розробка нових методів і засобів щодо оцінювання ефективності використання апаратних ресурсів, донавчання існуючих моделей на основі підходу transfer learning, а також розвитку платформ управління генерацією даних.

Мета кваліфікаційної роботи передбачає розробку, обґрунтування і дослідження методів і засобів організації інфраструктури комп’ютерних систем при імплементації генеративних алгоритмів машинного навчання.

При проведенні дослідження необхідно розв’язати наступні задачі:

– аналіз наукових праць і практичних способів організації інфраструктури великих даних;

– аналітичне дослідження великих мовних моделей (LLM) та генеративних алгоритмів на базі енкодерів та декодерів;

– обґрунтування вибору платформ для навчання моделей на нових даних з існуючою архітектурою глибоких нейронних мереж;

– програмна імплементація чат-бота з використанням LLM моделей;

– розробка методу оцінювання ефективності інфраструктури комп’ютерної системи з генеративними алгоритмами.

Об’єкт дослідження: процес організації та оцінювання ефективності інфраструктури комп’ютерної системи з великими мовними моделями.

Предмет дослідження: великі мовні моделі, алгоритми глибоких нейронних мереж, апаратна інфраструктура LLM, методи оцінювання ефективності програмно-апаратних інфраструктур.

Методи дослідження: Для вирішення поставлених задач використано наступні методи: аналіз та узагальнення – при аналізі основних понять та принципів організації великих даних; формалізації – при розробці критеріїв і процедур оцінювання ефективності інфраструктур великих мовних моделей; проектування, програмування та інтеграція – при розробці способів донавчання великих мовних моделей; експеримент та вимірювання – при визначенні оптимальних інфраструктур для ефективного функціонування моделей з генеративними алгоритмами.

Наукова новизна отриманих результатів. Наукова новизна полягає у вирішенні науково-практичної задачі імплементації технології блокчейн в організацію розподілених систем зберігання даних.

– уперше запропоновано критерії продуктивності у процесі попереднього опрацювання запитів до моделей з генеративними алгоритмами: час формування першого токену, час генерації вихідного токену для кожного користувача, прихована затримка формування відповіді користувачам та пропускної здатності моделі, що дало змогу керувати ресурсами і налаштуваннями інфраструктури апаратно-програмної інфраструктури і оптимально їх розподіляти.

– уперше запропоновано метод та показники оптимізації використання ресурсів пам’яті та графічних процесорних ядер при формуванні висновків (відповідей) великими мовними моделями, що дало змогу здійснювати оптимальне налаштування інфраструктури і забезпечити баланс продуктивності та витрат на навчання та донавчання моделей.

Практичне значення одержаних результатів. Програмно імплементовано механізми прямого доступу до моделей з генеративними

алгоритмами машинного навчання та експериментально доведено їх ефективність. Визначено необхідність та доцільність проведення додаткових досліджень щодо вибору оптимальних платформ для підтримки підходу transfer learning з врахуванням запропонованих критеріїв та методу оптимізації використання ресурсів програмно-апаратної інфраструктури.

Публікації. Результати кваліфікаційної роботи апробовані на XII Міжнародній науково-практичній конференції молодих учених та студентів (6-7 грудня 2023 р.) та XI науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (13-14 грудня 2023 року) як тези конференцій.

1. Луцків А.М., Островський А.Я. Характеристики та сфера застосування великих мовних моделей. Матеріали XII міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (6-7 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 452.

2. Луцків А.М., Островський А.Я. Організація доступу до моделі GPT-3 засобами мови Python. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 168.

Структура роботи. Кваліфікаційна робота містить розрахунково-пояснювальну записку та графічний матеріал. До складу записки входить вступу, 4 розділи, загальні висновки, список використаних джерел і додатки. Обсяг роботи: розрахунково-пояснювальна записка – 85 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ ТА ОРГАНІЗАЦІЇ ВЕЛИКИХ ДАНИХ

1.1. Аналіз базових понять і властивостей великих даних

Перш, ніж перейти до безпосереднього аналізу великих даних, необхідно окреслити саме поняття даних. Під даними варто розуміти будь-які числа, букви та символи, над якими комп'ютерна система може виконувати операції і які можуть зберігатися та передаватись у формі електричних сигналів та записуватися на магнітні, оптичні чи механічні носії запису.

Великі дані (Big Data) представляють собою множину даних величезного розміру, потужність якої зростає за експоненційним законом [1]. Особливою характеристикою великих даних є те, що вони не можуть бути збережені та ефективно опрацьовані за допомогою традиційних засобів управління даними, що безпосередньо пов'язано з їх складністю та об'ємом.

На сьогодні існує багато прикладів і джерел формування великих об'ємів інформації. Як приклад, статистично в середньому у соціальну мережу Facebook потрапляє понад 500 ТБ новозгенерованих даних щоденно. Ця інформація в основному формується внаслідок завантаження користувацьких фото і відео, при групових переписках, а також у коментарях до постів тощо. Інший приклад – один реактивний двигун може генерувати 10+ терабайт даних за 30 хвилин польоту. З багатьма тисячами рейсів на день генерація даних досягає багатьох петабайт.

Фондові біржі також генерують великі об'єми даних, які формують близько одного терабайта нових торгових даних на день (рис. 1.1).

Рис. 1.1. Фондова біржа Нью-Йорка

Розрізняють три види великих даних:

– структуровані;
– не структуровані;
– частково або напівструктуровані.

Будь-які дані, які підлягають збереженню з можливістю одержання доступу до них, а також опрацювання у формі деякого наперед визначеного формату, називають структурованими. Для зберігання та опрацювання такого виду даних розроблено ефективні методи і способи, які дають змогу отримувати нову цінність з них. Однак, зважаючи на значне зростання структурованої інформації, можуть виникати проблеми з їх зберіганням, оскільки розміри при цьому досягають кількох зетабайтів. Дивлячись на ці цифри, можна легко зрозуміти, чому дано назву Big Data, і уявити собі проблеми, пов’язані з їх зберіганням і обробкою.

Як приклад структурованих даних можна представити таблицю у реляційних базах даних (рис. 1.2).

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

Рис. 1.2. Структура даних у вигляді таблиці

Під неструктурованими даними, зазвичай, розуміють набір будь-яких даних, форма і структура яких наперед не відома. Окрім великого об’єму, такий вид даних формує багато проблем, пов’язаних з їх опрацюванням та одержанням цінності для кінцевого користувача. Яскравим прикладом неструктурованої інформації є джерело, яке генерує дані різної природи – текст, зображення, відео і т.п.

Багато компаній сьогодні володіють величезною кількістю даних, проте, на жаль, вони не знають, як отримати з них цінність, оскільки ці дані знаходяться в необробленому або неструктурованому форматі. Як приклад не структурованих даних можна навести результати пошуку за допомогою Google Engine (рис. 1.3).

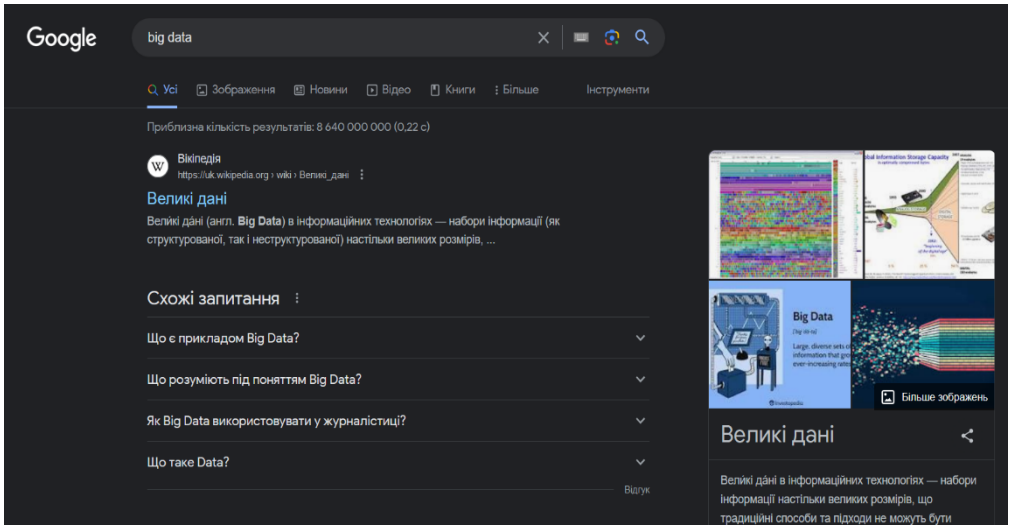

Рис. 1.3. Результат пошуку у вигляді неструктурованих даних

Частково структуровані дані можуть включати обидва, попередньо розглянуті, формати даних. Їх можна вважати структурованими за формою, однак насправді вони не визначені, наприклад, порожні записи у таблиці в реляційній СКБД. У якості прикладу цього виду великих даних можна вважати частково структуровані дані, які представляються у форматах XML або JSON. Приклад вмісту XML-файлу представлено на рис. 1.4.

```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

Рис. 1.4. Приклад частково структурованих даних

Прогнозування щодо росту даних за багатьма дослідженнями [2-5] матиме експоненційний характер, як показано на рис. 1.5.

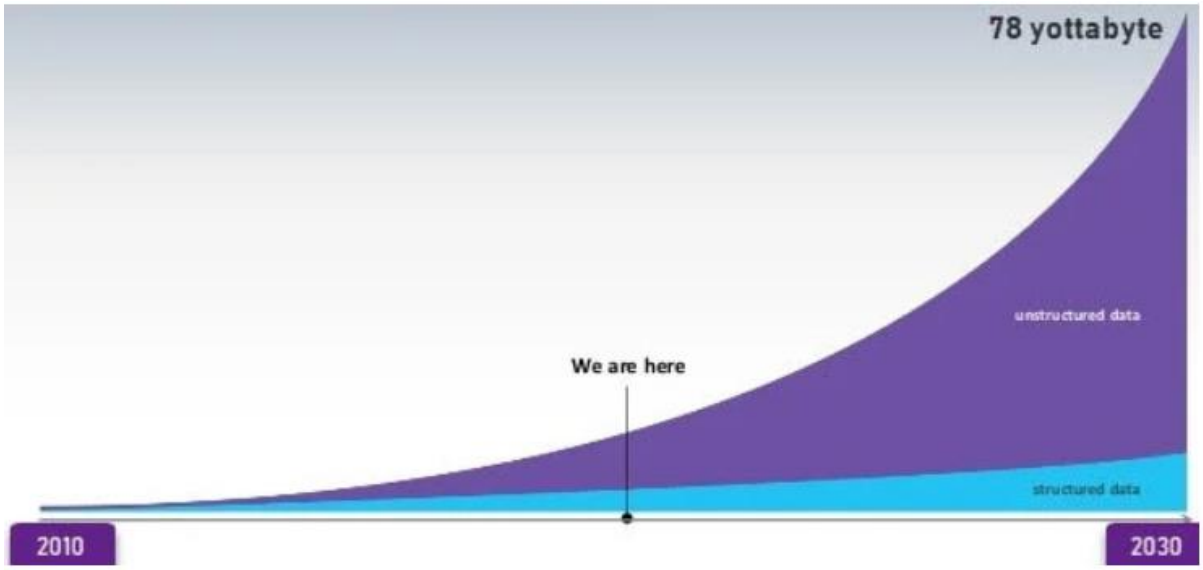

Рис. 1.5. Тренд щодо зростання кількості даних

Потрібно звернути увагу на те, що джерелом неструктурованої інформації у веб-додатках є дані, які формуються у журналах, логах,

історіях транзакцій. OLTP системи орієнтовані в основному на опрацювання структурованих даних, зокрема, при використанні реляційного підходу до їх організації у вигляді таблиць.

Великі дані характеризуються наступними властивостями:

– об’єм;
– різнорідність;
– швидкість;
– варіативність.

Щодо атрибуту «об’єм» (volume), то вже з самої назви «Big Data» стає зрозумілим, що мова іде про дані величезного розміру. Такий об’єм інформації визначає значну частку впливу при виявленні її цінності. На основі розміру даних визначається також і те, чи наявна множина дійсно належить до класу великих даних. Отже, «об’єм» є однією з тих властивостей, на яку обов’язково потрібно звертати увагу при розробці бізнес-рішень.

Наступна властивість великих даних стосується їх різноманітності та різнорідності (variety). Поняття різнорідності даних застосовують до неоднорідних джерел, які можуть генерувати різні види даних (структуровані та неструктуровані). До цього, в більшості випадків усі програмні додатки використовували або електронні таблиці, або реляційні бази даних, що було пов’язано із розвинутістю як математичного так і прикладного інструментарію. На теперішній час дані можуть бути представлені у вигляді листів електронної пошти, фотозображень, відеопотоків, даних IoT-пристроїв, різних форматів текстових даних, аудіо файлів, які також підлягають аналізу і мають певну цінність. Така різнорідність і неструктурованість даних формує ряд проблем щодо їхнього зберігання, добування та аналізу.

Властивість «швидкість» (velocity) пов’язана з продуктивністю при генерації даних. Саме швидкість визначає формування та опрацювання даних у відповідності до вимог і дає змогу визначити їх реальну цінність та

потенціал. Окрім цього, швидкість, в контексті великих даних, відображає міру щодо надходження інформації з різних джерел, зокрема, це стосується метаданих бізнес-процесів, журналів логування, соціальних мереж, IoT-пристроїв, мобільних гаджетів і т.п. При цьому потік даних є чималим і безперервним.

Варіативність або мінливість стосується неузгодженості, яку час від часу можуть демонструвати дані. Це перешкоджає процесу ефективного їх опрацювання та керування ними.

Можливість опрацьовувати великі дані в СКБД забезпечує значні переваги, зокрема:

– компанії можуть використовувати зовнішню інформацію при прийнятті рішень, зокрема, це стосується даних із соціальних та пошукових систем, що дає змогу організаціям точно налаштовувати свої бізнес-стратегії;

– покращення процесу підтримки та обслуговування клієнтів – на зміну традиційним системам зворотного зв’язку приходять нові системи, розроблені з використанням технологій Big Data, у яких великі дані та технології опрацьовування природної мови використовуються для читання та оцінки відгуків споживачів;

– раннє виявлення ризику для продукту/послуг за умови його попередньої експлуатації;

– підвищення ефективності роботи – технології Big Data можна використовувати для створення зони розміщення або посадкової зони для нових даних перед тим, як визначити, які з них потрібно перемістити до сховища даних. Крім того, така інтеграція технологій великих даних і сховища даних допомагає організації розвантажувати дані, до яких рідко звертаються.

1.2. Аналіз особливостей сучасного життєвого циклу інженерії великих даних

Фахівці з великих даних сьогодні відчувають кардинальні зміни у роботі завдяки інструментам опрацювання даних, що пов’язано із зростанням стратегічності завдань та подальшого застосування добутих даних. Оскільки специфіка фреймворків Big Data стала більш абстрагованою, сучасний інженер з великих даних може зосередитися на ширшій картині, дедалі більше дбаючи про завдання, що знаходяться далі у конвеєрі формування вартості, зокрема, моделювання даних, якості, безпеки, управління, архітектури та оркестрації. У той же час фахівці дедалі більше застосовують найкращих практик сфери комп’ютерної і програмної інженерії. Незважаючи на те, що інженерія програмного забезпечення та інженерія даних є різними напрямами, вони також мають деякі суттєві подібності: обидва вимагають вирішення проблеми шляхом написання, розгортання та підтримки коду. Таким чином, сучасний інженер Big Data повинен обов’язково бути знайомим із гнучкою методологією розробки програмних рішень, тестуванням коду та методами контролю версій. Крім найкращих практик, інженерія даних також запозичила концепції програмної інженерії. Чудовим прикладом є функціональна інженерія даних, яка бере свій початок у функціональному програмуванні. Основна ідея цього полягає в тому, що завдання, наприклад переміщення даних із системи A до системи B, має бути ідемпотентним, що означає одержання завжди однакового результату при виконанні завдання.

Коли завдання не виконуються або потрібно змінити логіку, потрібно знати, що його повторний запуск є безпечним і не призведе до дублювання даних або будь-якого іншого типу неправильного стану. Тому ідемпотентність є важливою для працездатності конвеєра даних.

Іншою прийнятою концепцією є декларативне програмування, рівень абстракції на основі імперативного програмування, який фокусується на

тому, що, а не як. Декларативний конвеєр даних скаже: «перемістіть дані із системи А до системи Б», не вказуючи точний потік даних.

Декларативні конвеєри актуальні в розробці даних, оскільки вони є хорошою основою для спостереження, моніторингу якості даних і походження даних. Декларативна концепція тісно пов’язана з тенденцією відходу від конвеєрів даних і охоплення продуктів даних, що стало можливим завдяки абстракціям, які надають сучасні оркестратори даних.

Інженери даних тепер більше думають про продукт, який призначений конвеєр, як-от певну інформаційну панель або представлення, а потім створюють конвеєр на основі цього. Ідея даних як продукту є основною цінністю архітектури мережі даних. Розвиток Python також позначився в інженерії даних, оскільки протягом багатьох років він утверджувався як дуже надійна мова програмування. Спільнота Big Data широко прийняла Python завдяки численним вбудованим бібліотекам для обробки та відображення даних. Одним із яскравих прикладів є pandas, розроблена спеціально для видобування та перетворення даних. З’явилися інші важливі інструменти на основі Python, такі як PySpark, інтерфейс для Apache Spark, який дозволяє інтерактивно аналізувати дані в розподіленому середовищі. Можна з упевненістю сказати, що Python і SQL є мовами, які повинен знати кожен інженер даних сьогодні. Те, як організації структурують команди обробки даних, змінилося з роками. Зараз можна спостерігати перехід до децентралізованих груп обробки даних, самообслуговуваних платформ даних і способів зберігання даних за межами сховища даних, таких як озеро даних, або згадана раніше мережа даних, щоб краще задовольняти потреби кожного споживача.

Незважаючи на те, що організаційні зміни є суперечливими, а експерти мають різні думки щодо того, що найкраще працює, спостерігається тенденція щодо формування експертів домену власниками даних, які вони використовують, замість того, щоб мати одну центральну команду, яка відповідає за «джерело правди».

У результаті багато інженерів даних тепер належать до команди центральної платформи, яка відповідає за оптимізацію різних аспектів стеку даних замість того, щоб володіти даними. Головною проблемою, пов’язаною з вищезазначеними архітектурними та організаційними змінами, є підтримання загального розуміння даних. Ось чому сьогодні спостерігається впровадження таких концепцій, як семантичний рівень, який відображає складні дані у знайомі бізнес-терміни, щоб запропонувати уніфіковане консолідоване уявлення про дані в системах. Отже, сьогоднішню роль інженерії даних можна означити згідно [5]: «Інженерія даних – це розробка, впровадження та підтримка систем і процесів, які приймають необроблені дані та створюють високоякісну послідовну інформацію, яка підтримує подальший їх аналіз і машинне навчання. Інженерія даних поєднує безпеку, керування даними, DataOps, архітектуру даних, оркестровку та розробку програмного забезпечення». У зв’язку з цим, життєвий цикл при опрацюванні великих даних має вигляд як показано на рис. 1.6.

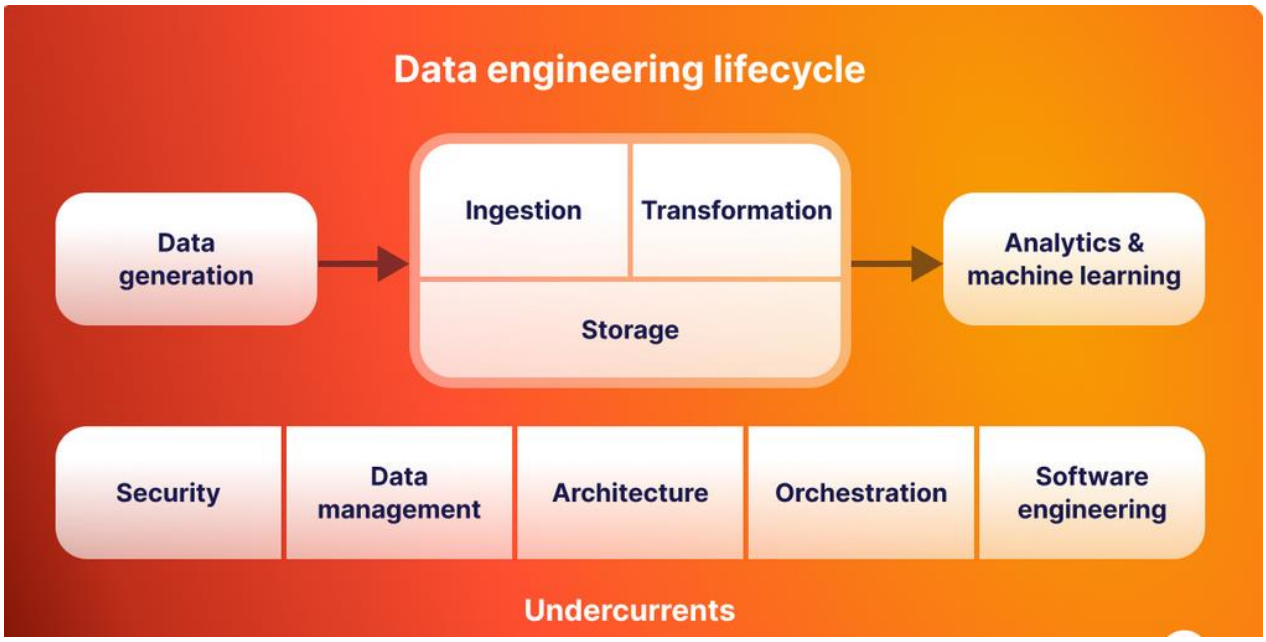

Рис. 1.6. Життєвий цикл інженерії великих даних

Як видно з рис. 2.6, інженер обробки даних сьогодні контролює весь процес їх розробки даних, починаючи від збору із різних джерел і закінчуючи доступом до подальших процесів. Роль фахівця вимагає знайомства з декількома етапами життєвого циклу розробки даних і здатності оцінювати інструменти обробки даних для оптимальної продуктивності в кількох вимірах, включаючи ціну, швидкість, гнучкість, масштабованість, простоту, багаторазове використання та сумісність.

1.3. Аналіз базових концепцій LLM

Великі мовні моделі (LLM), згідно [6], представляють собою підмножину моделей глибокого навчання, відомих як моделі «нейронної мови». Їх особливістю є те, що вони навчаються на потужному наборі текстових даних, а це в свою чергу дає змогу їм вивчати статистичні шаблони, граматику та семантику людської мови.

На відміну від попередніх підходів щодо опрацювання природної мови, LLM можуть забезпечувати генерацію зв’язного і змістовно релевантного тексту, що робить їх надзвичайно універсальними для розуміння природної мови та завдань створення.

До ключових характеристик LLM належить:

– масштаб: LLM великі, часто містять від сотень мільйонів до мільярдів параметрів, що дозволяє їм фіксувати складні мовні шаблони.

– попередня підготовка: вони попередньо навчені масивним текстовим корпусам, що дає їм широке розуміння мови.

– «тонка настройка»: LLMs можна точно налаштувати для конкретних завдань, що робить їх адаптованими до широкого спектру застосування.

– генерація: вони можуть створювати текст, схожий на той, який фоомує людина, включаючи статті, код, вірші тощо.

– відповіді на запитання: LLM чудово відповідають на запитання, надаючи стислі відповіді на основі контексту.

Великі мовні моделі мають велике значення у різних сферах діяльності, зокрема:

1. Розуміння природної мови. LLMs досягли надзвичайного успіху в задачах розуміння природної мови. Це стосується аналізу емоцій та настроїв людини, мовного перекладу і чат-ботів. Здатність до розуміння контексту і генерації зв'язного тексту на основі LLM відкрила нові можливості для автоматизації та покращення спілкування.

2. Генерація контенту: Великі мовні моделі відіграють вагому роль у створенні завдань, починаючи від творчого написання та підсумовування вмісту до створення коду та написання звітів. Це має застосування в творчих галузях, створенні контенту та автоматизації повторюваних завдань.

3. Пошук інформації: LLM забезпечують більш ефективний і точний пошук інформації. Вони забезпечують роботу пошукових систем, систем рекомендацій і персональних помічників, допомагаючи користувачам точно і швидко знаходити адекватний контент.

4. Доступність. У даному випадку під доступністю розуміють підвищення рівня якості життя для людей з обмеженими можливостями. LLM можуть перетворювати текст у звук, допомагати у мовному перекладі та покращувати загальний досвід користувача для тих, хто має особливі потреби.

5. Наукові дослідження. LLMs підтримують дослідження в різних дисциплінах, аналізуючи та узагальнюючи наукову літературу, допомагаючи дослідникам в аналізі даних і надаючи ідеї в таких галузях, як охорона здоров'я та матеріалознавство.

6. Бізнес-додатки. LLM революціонізують бізнес-ландшафт завдяки додаткам у чат-ботах підтримки клієнтів, маркетингових дослідженнях, контент-маркетингу та аналітиці даних. Вони покращують залучення клієнтів і сприяють ухваленню рішень на основі даних.

7. Етичні та суспільні наслідки. Поширення LLM викликає етичні питання, пов’язані з дезінформацією, конфіденційністю та упередженням у ШІ. Їхня важливість підкреслюється необхідністю вирішення цих проблем, одночасно використовуючи їхній потенціал.

Підсумовуючи, великі мовні моделі представляють собою монументальний стрибок у можливостях ШІ, пропонуючи широкий спектр застосувань у різних галузях. Їх здатність розуміти, створювати та маніпулювати людською мовою має глибокі наслідки для технологій, бізнесу та суспільства. Розкриваючи нові можливості, відповідальна розробка та використання LLMs має першорядне значення для вирішення етичних і суспільних проблем.

Варто зазначити, що великі мовні моделі не є офіційним терміном, це відноситься до сфери опрацювання природної мови, що використовує моделі на основі глибокого навчання, зокрема трансформери. До складу архітектури трансформерів входять тисячі параметрів, що допомагає досягти кращих результатів у завданнях NLP. З часом дослідники генерували нові ідеї, а розмір моделі почав зростати з точки зору кількості параметрів, що використовуються в архітектурі, вимог до корпусу даних, обчислювальних пристроїв, часової ефективності та багатьох інших. Через це ці моделі отримали назву великих мовних моделей.

Із розвитком генеративних алгоритмів та прогресом у вдосконаленні моделі GPT, то: перша версія GPT-1, випущена у 2018 році, включає в себе 117 млн параметрів із 985 млн слів. Друга версія GPT-2 вийшла у 2019 році та вже містила 1,5 млрд параметрів. GPT-3 реалізована у 2020 році та включає в себе 175 млрд параметрів. Chat GPT також базується на цій моделі. Очікується, що фінальна версія моделі GPT-4 буде випущена у 2023 році і, ймовірно, буде містити трильйони параметрів.

1.4. Висновки до розділу

У даному розділі одержано наступні результати:

1 Проведено аналіз базових понять та властивостей великих даних, встановлено особливості їх формування та методи опрацювання, що дало змогу обґрунтувати доцільність визначення критеріїв продуктивності програмно-апаратної інфраструктури для моделей з генеративними алгоритмами.

2 Проаналізовано життєвий цикл великих даних та визначено особливості етапів і процесів, характерних при опрацюванні великих даних, а також функції Big Data інженерів, що дало змогу чітко визначити процеси конвеєра роботи з такими даними, зокрема щодо моделювання даних, якості, безпеки, управління, архітектури та оркестрації.

3 Визначено базові концепції великих мовних моделей з реалізацією генеративних алгоритмів і встановлено, що вони належать до класу інтелектуальних моделей, які дають змогу досліджувати статистичні шаблони, граматику та семантику людської мови та використовують потужні набори даних і відповідно значні апаратно-програмні ресурси.

РОЗДІЛ 2
КРИТЕРІЇ ТА МЕТОДИ ЗАБЕЗПЕЧЕННЯ ОПТИМАЛЬНОСТІ ФУНКЦІОНУВАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

2.1. Процес генерації тексту у великих мовних моделях

Генерація тексту великими мовними моделями відбувається у два етапи:

1. Попереднє заповнення – токени вхідного запиту опрацьовуються паралельно;
2. Декодування – текст генерується по одному «токену» за раз у авторегресійний спосіб.

Кожен згенерований токен додається до вхідних даних і повертається у модель для формування наступного токена. Генерація припиняється, коли LLM виводить спеціальний стоп-токен або коли виконується умова, визначена користувачем (наприклад, згенеровано певну максимальну кількість токенів).

Лексеми можуть бути словами або підсловами; точні правила поділу тексту на лексеми відрізняються у різних моделях. Наприклад, можна порівняти те, як моделі Llama та OpenAI токенізують текст.

Хоча постачальники результатів роботи великих мовних моделей часто говорять про продуктивність у показниках на основі токенів (наприклад, токени/секунда), ці цифри не завжди можна порівняти між типами моделей, враховуючи ці відмінності.

Для конкретного прикладу команда Anyscale виявила, що токенізація Llama 2 на 19% довша, ніж токенізація ChatGPT

Дослідники з HuggingFace також виявили, що Llama 2 потребує приблизно на 20% більше токенів для навчання такої ж кількості тексту, як GPT-4.

2.2. Визначення критеріїв обслуговування великих мовних моделей

У кваліфікаційній роботі для визначення ефективності обслуговування LLM пропонується використати чотири ключові показники:

1. Час формування першого токена (TTFT). Даний критерій інтерпретує швидкість з якою користувачі починають бачити вихідні дані моделі після введення свого запиту. Короткий час очікування відповіді важливий для взаємодії в режимі реального часу, але менш важливий для роботи в режимі офлайн. Ця метрика визначається часом, необхідним для опрацювання запиту та генерації першого вихідного токена.

2. Час формування вихідного токену для кожного користувача (TPOT) – критерій, який використовується для визначення часу створення вихідного токена для кожного користувача, який сформував запит до системи. Цей показник відповідає тому, як кожен користувач сприйме «швидкість» моделі. Наприклад, TPOT у 100 мілісекунд/токен складатиме 10 токенів на секунду на користувача або ~450 слів на хвилину, що швидше, ніж може прочитати звичайна людина.

3. Затримка. Показник затримки інтерпретує загальний час, який потрібний моделі для створення повної відповіді для користувача. Загальну затримку відповіді можна розрахувати за допомогою попередніх двох показників:

$$Delay = TTFT + TPOT * (NAT) \qquad (2.1)$$

де $Delay$ – затримка на генерацію відповіді LLM моделлю на запит користувача;

$TTFT$ – час формування першого токену;

$TPOT$ – час формування токену для кожного користувача;

NAT – кількість згенерованих токенів у відповіді на запит користувача.

4. Пропускна здатність. Показник пропускної здатності інтерпретує кількість вихідних токенів за одиницю часу, яку сервер може згенерувати для всіх користувачів і запитів.

Використання таких критеріїв обслуговування великих мовних моделей, або по-іншому моделй з генеративними алгоритмами формує оптимізаційну задачу, яка полягає у забезпеченні найкоротшого часу отримання першого токена, найвищої пропускної здатності і найшвидшого час формування вихідного токену. Іншими словами необхідно зробити так, щоб моделі генерували текст якомога швидше для якомога більшої кількості користувачів, яку може підтримувати апаратна інфраструктура.

Варто відмітити, що існує компроміс між пропускною здатністю та часом формування вихідного токена: якщо модель опрацьовує 16 запитів користувачів одночасно, то буде вища пропускна здатність порівняно з послідовним виконанням запитів, але при цьому знадобиться більше часу для створення вихідних токенів для кожного користувача.

Для оцінювання затримки формування відповідей моделями з генеративними алгоритмами можуть також використовуватися латентні критерії, які відрізняються за своєю важливістю для користувача:

Довжина виведення домінує над загальною затримкою відповіді: для обчислення середньої затримки, зазвичай, можна брати очікувану/максимальну довжину вихідного токена та помножити її на загальний середній час на вихідний токен для моделі.

Довжина вхідних даних не є важливою для продуктивності, але важлива для вимог до обладнання: додавання 512 вхідних токенів збільшує затримку менше, ніж генерація 8 додаткових вихідних токенів у моделях MPT (модель трансформера з відкритим вихідним кодом, розроблена MosaicML для вирішення проблем навчання та розгортання великих мовних моделей). Однак необхідність підтримувати довгі вхідні дані може

ускладнити обслуговування моделей. Наприклад, рекомендовано використовувати A100-80GB (або новіший) для обслуговування MPT-7B з максимальною довжиною контексту 2048 токенів.

Загальна затримка сублінійно змінюється залежно від розміру моделі: на тому самому апаратному забезпеченні більші моделі працюють повільніше, але коефіцієнт швидкості не обов'язково відповідатиме коефіцієнту кількості параметрів. Затримка MPT-30B приблизно в 2,5 рази перевищує затримку MPT-7B. Затримка Llama2-70B приблизно в 2 рази перевищує затримку Llama2-13B. Потенційні клієнти часто просять надати середню затримку формування результатів.

Однак, перш ніж прив'язуватися до конкретних цільових показників затримки («потрібно менше ніж 20 мс на маркер»), слід витратити деякий час на визначення очікуваної вхідної та бажаної довжини згенерованої відповіді.

2.3. Генерації висновків у LLM моделях

Оптимізація висновків LLM отримує переваги над такими загальними методами як:

– злиття операторів – поєднання різних суміжних операторів разом часто призводить до більшої затримки.

– квантування – активації та ваги стискаються для використання меншої кількості бітів;

– стиснення – формують розрідженість або дистиляцію.

– паралелізм – тензорний паралелізм на кількох пристроях або конвеєрний паралелізм для більших моделей.

Окрім цих методів, існує багато важливих оптимізацій, специфічних для Transformer. Яскравим прикладом цього є кешування Key-Value (ключ-значення). Механізм звернення уваги в моделях на основі Transformer, що містять лише декодер, неефективний з точки зору обчислень.

Кожен токен звертається до всіх раніше відомих токенів і, таким чином, повторно обчислює багато однакових значень, коли створюється кожен новий токен.

Для прикладу, генерація токена N передбачає звернення токена $(N-1)$ до попередньо згенерованих токенів, включаючи самого першого. Подібним чином, під час генерації $(N+1)$-го токена увагу для N-го токена знову потрібно звернути на попередні. Кешування ключ-значення (KV), тобто збереження проміжних ключів-значень для рівнів уваги, використовується для збереження цих результатів з метою подальшого повторного використання, уникаючи повторних обчислень.

2.4. Пропускна здатність пам'яті

Обчислення у моделях з генеративними алгоритмами переважно складаються з операцій множення матриць. Такі операції з матрицями малого розміру, як правило, обмежені пропускною здатністю пам’яті на більшості апаратного забезпечення.

Під час генерації токенів авторегресійним способом розмірb матриць активації (визначений розміром партії (batch) та кількістю токенів у послідовності) є малим при малих розмірах партії. Таким чином, швидкість залежить від того, наскільки швидко можна завантажити параметри моделі з пам’яті графічного процесора в локальні кеш/реєстри, а не від того, як швидко можна обчислювати завантажені дані.

Доступна і досягнута пропускна здатність пам’яті в апаратному забезпеченні є кращим показником швидкості генерації токенів, ніж їхня пікова продуктивність при виконанні обчислень.

Ефективне використання апаратного забезпечення при формуванні висновків є дуже важливим з точки зору економії витрат. Графічні процесори дорогі, і вони необхідні для того, щоб виконувати якомога більше операцій.

Спільне використання сервісів формування висновків передбчає збереження коштів, тобто витрати будуть на низькому рівні. Це досягається шляхом поєднання робочих навантаження від багатьох користувачів із заповненням окремих прогалин та об’єднанням подібних запитів.

Для таких великих моделей, як Llama2-70B, можна досягти хорошого співвідношення ціна/продуктивність лише при великих розмірах партій. Наявність системи обслуговування висновків, яка може працювати з великими розмірами пакетів чи партій, має вирішальне значення для ефективності витрат. Однак велика партія означає більший розмір кешу KV, що, у свою чергу, збільшує кількість графічних процесорів, необхідних для обслуговування моделі. Тут операторам спільних послуг потрібно піти на певні компроміси щодо витрат і оптимізувати системи.

2.5. Використання пропускної здатності моделі

Як було зазначено раніше, висновок для LLM з меншими розмірами пакетів, особливо під час декодування, є вузьким місцем через те що існує проблема забезпечення швидкості завантаження параметрів моделі з пам’яті пристрою в обчислювальні блоки.

Пропускна здатність пам’яті визначає швидкість переміщення даних. Щоб виміряти продуктивність використання основного апаратного забезпечення, пропонується застосувати новий показник під назвою «Використання пропускної здатності моделі» (англ. MBU).

MBU визначається як:

$$MBU = \frac{AMB}{PMB} \tag{2.2}$$

де MBU – показник використання пропускної здатності моделі;

AMB – досягнута пропускна здатність пам’яті;

PMB – пікова пропускна здатність пам’яті.

Досягнута пропускна здатність пам’яті обчислюється за формулою:

$$AMB = \frac{TMPS + Cashe(KV)}{TPOT} \quad 2.3)$$

де $TMPS$ – загальний розмір параметрів моделі;

$Cashe(KV)$ – розмір кешу для зберігання даних ключ-значення;

$TPOT$ – час формування токену для кожного користувача.

Наприклад, якщо параметр 7B, що працює з 16-бітною точністю, має $TPOT$, що дорівнює 14 мс, тоді він переміщує 14 ГБ параметрів за 14 мс, що означає використання пропускної здатності на рівні 1 ТБ/с. Якщо пікова пропускна здатність машини становить 2 ТБ/с, то відповідно використовується 50% MBU. Для простоти в цьому прикладі ігнорується розмір кешу KV, який є малим для менших розмірів пакетів і меншої довжини послідовності. Значення MBU, близькі до 100%, означають, що система формування висновків ефективно використовує доступну пропускну здатність пам’яті.

MBU також корисний показник для порівняння різних систем формування логічного висновку з точки зору використання інфраструктури на рівні апаратного і програмного забезпечення, оскільки забезпечує його нормалізацію. MBU є доповненням до метрики Model Flops Utilization, яка є важливою в налаштуваннях, пов’язаних з обчисленнями.

На рис. 2.1 показано візуальне представлення MBU на графіку, схожому на графік лінії даху. Суцільна похила лінія заштрихованої помаранчевим кольором області показує максимально можливу пропускну здатність, якщо пропускна здатність пам’яті повністю насичена на 100%.

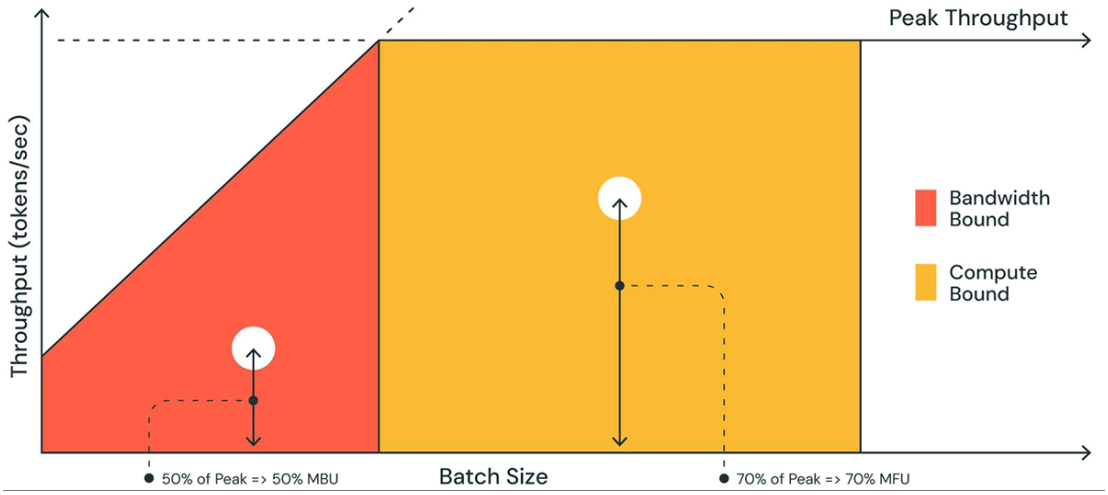

Рис. 2.1. Приклад візуалізації показника MBU

Рис. 2.1 ілюструє показник використання пропускної здатності моделі і показник використання провалів моделі. MBU та MFU – це частки піків, досягнутих відповідно в областях, пов’язаних з пам’яттю та обчисленнями. Однак насправді для малих розмірів партії (біла крапка), спостережувана продуктивність нижча за максимальну настільки, наскільки нижча є міра MBU.

Для великих розмірів пакетів (жовта область) система обмежена обчисленнями, а досягнута пропускна здатність як частка від максимально можливої пропускної здатності вимірюється як використання провалів моделі (MFU).

MBU та MFU визначають, скільки ще доступного простору для збільшення швидкості при формуванні логічного висновку на заданому апаратному забезпеченні. На рис. 2.2 показано виміряний MBU для різних ступенів паралелізму тензорів за допомогою сервера формування висновків на основі TensorRT-LLM.

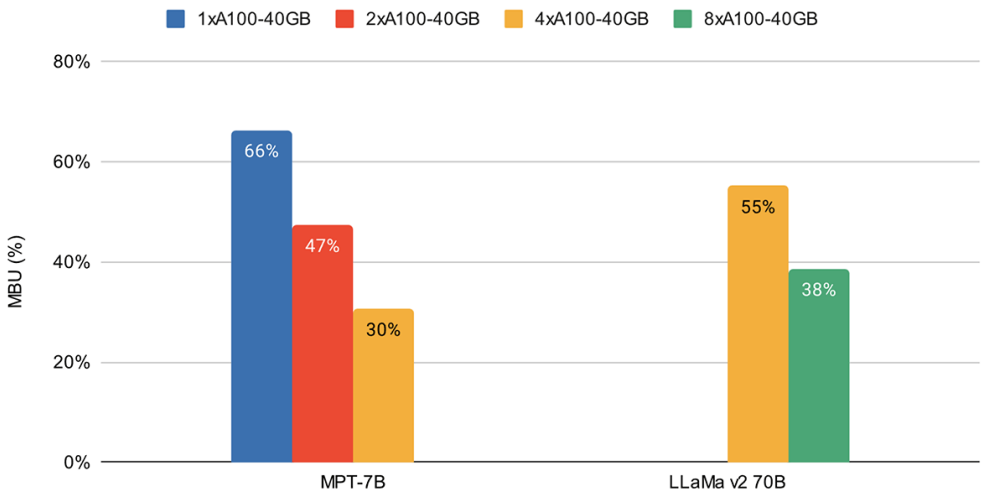

Рис. 2.2. Емпірично спостережуваний MBU для різних ступенів паралелізму тензорів із TensorRT-LLM на графічних процесорах A100-40G.

Експериментальні значення показника використання пропускної здатності моделі, який представлений на рис. 2.2, використовує запити з послідовністю 512 вхідних токенів із розміром пакету 1.

Максимальне використання пропускної здатності пам’яті досягається при передачі великих безперервних фрагментів пам’яті. Коли менші моделі, такі як MPT-7B, розподіляються між кількома графічним процесором, спостерігається менший MBU, оскільки виконується переміщення менших фрагментів пам’яті у кожному графічному процесорі. Рис. 2.3 демонструє емпірично спостережуваний MBU для різних ступенів паралелізму тензорів і розмірів пакетів на графічних процесорах NVIDIA H100.

На рис. 2.3 використовувались запити, що містили послідовності з 512 вхідних токенів. MBU зменшується зі збільшенням розміру партії. Однак, коли відбувається масштабування графічних процесорів, відносне зниження MBU менш значне. Варто також зазначити, що вибір апаратного забезпечення з більшою пропускною здатністю пам’яті може підвищити продуктивність з меншою кількістю GPU.

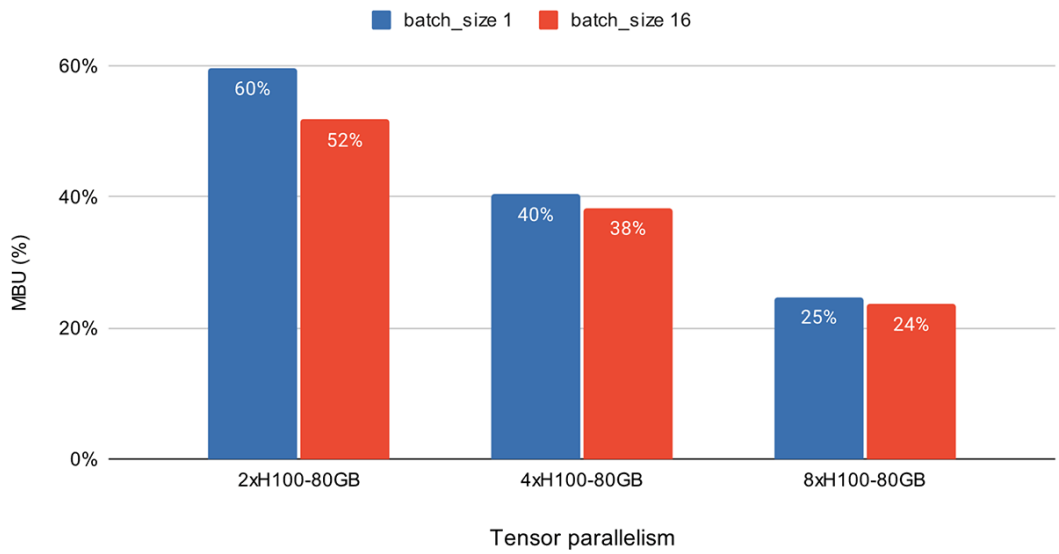

Рис. 2.3. Емпірично спостережуваний MBU для різних розмірів пакетів і режимів паралелізму тензорів на графічних процесорах H100-80G.

При розмірі партії 1 можна досягти вищого MBU 60% на графічних процесорах 2xH100-80 ГБ порівняно з 55% на графічних процесорах 4xA100-40 ГБ (рис. 2.2).

2.6. Порівняльний аналіз показників ефективності LLM моделей та ресурсів програмно-апаратної інфраструктури

Під час проведення порівняльного аналізу першою використовувалась метрика латентної затримки. Спочатку експериментально вимірювався час формування першого токена (TTFT) і час одержання вихідного токену (TPOT) для різних ступенів тензорного паралелізму моделей MPT-7B і Llama2-70B.

У міру того, як вхідні запити масштабуються, час для створення першого маркера починає використовувати значну частину загальної прихованої затримки. Розпаралелювання тензорів на кількох графічних процесорах допомагає зменшити її. На відміну від навчання моделі, масштабування за рахунок збільшення кількості графічних процесорів забезпечує суттєве зменшення затримки формування висновку. Для

прикладу, для Llama2-70B перехід від графічних процесорів 4x до 8x зменшує затримку лише на 0,7x за малих розмірів пакетів.

Однією з причин цього є те, що вищий паралелізм має нижчий MBU. Інша причина полягає в тому, що паралелізм тензорів створює накладні витрати на зв’язок між вузлом GPU.

Наступна метрика – час до формування першого токена. Результати порівняння для різних моделей при генерації першого токену представлені у табл. 2.1.

Таблиця 2.1

Результати часу одержання першого токена

	Час до одержання першого токену (мс)			
Модель	1xA100-40GB	2xA100-40GB	4xA100-40GB	8xA100-40GB
MPT-7B	46 (1x)	34 (0.73x)	26 (0.56x)	-
Llama2-70B	Не визначається		154 (1x)	114 (0.74x)

Час до першого токена, наданого вхідного запиту, становить 512 токенів із розміром пакета 1. Більшим моделям, таким як Llama2 70B, потрібно щонайменше 4xA100-40B графічних процесорів, щоб поміститися в пам’ять При більших розмірах пакетів вищий тензорний паралелізм призводить до більш значного відносного зменшення затримки токена.

На рис. 2.4 показано, як змінюється час формування вихідного токена для MPT-7B.

Затримка не змінюється лінійно зі збільшенням кількості GPU. Запити: послідовності з 128 вхідних і 64 вихідних токенів. При розмірі партії 1 перехід від 2x до 4x зменшує затримку токена лише на ~12%.

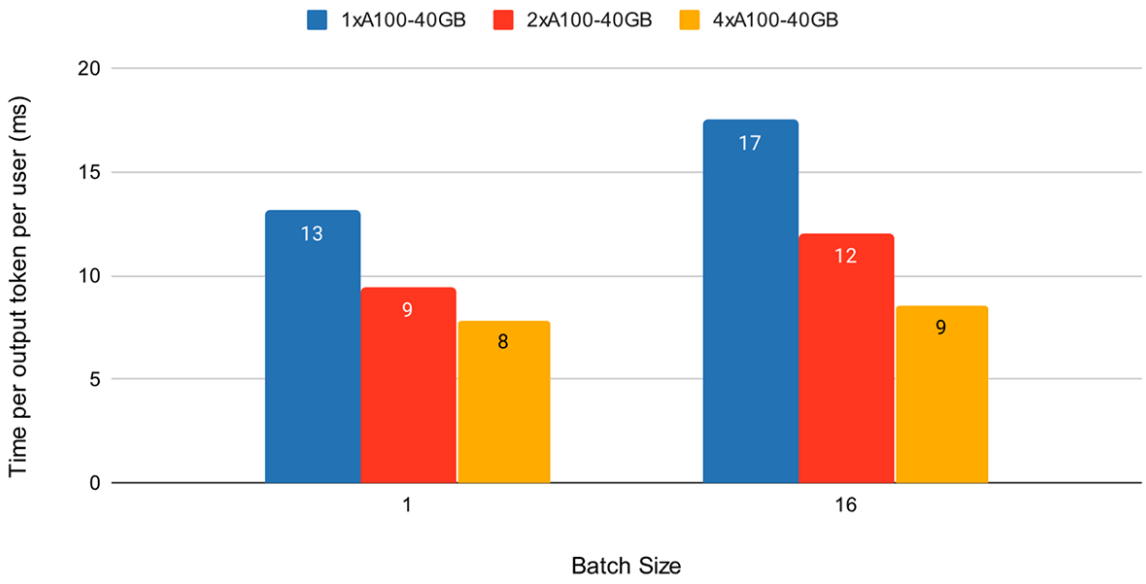

Рис. 2.4. Час на формування вихідного маркера для кожного користувача з масштабованою моделлю MPT-7B на графічних процесорах A100-40GB

При розмірі пакета 16 затримка з 4x на 33% менша, ніж з 2x. Це узгоджується з попереднім спостереженням, що відносне зменшення MBU є меншим при вищих ступенях тензорного паралелізму для розміру партії 16 порівняно з розміром партії 1.

На рис. 2.5 показано подібні результати для моделі Llama2-70B, за винятком того, що відносне покращення між 4x і 8x є менш вираженим. Також проводиться порівняння масштабованості GPU на двох різних апаратних платформах. Оскільки H100-80 ГБ має 2,15-кратну пропускну здатність графічного процесора порівняно з A100-40 ГБ, то можна побачити, що затримка на 36% нижча при розмірі пакета 1 і на 52% нижча при розмірі пакета 16 для систем 4x.

Потрібно звернути увагу, що номери GPU 1x40 ГБ, 2x40 ГБ і 1x80 ГБ тут відсутні, оскільки Llama-v2-70B (у float16) не підходить для цих систем.

Третя метрика, яка використовується при проведенні експериментів – пропускна здатність. Тут варто зазначити, що існує можливість змінювати пропускну спроможність та час формування токену, групуючи запити разом.

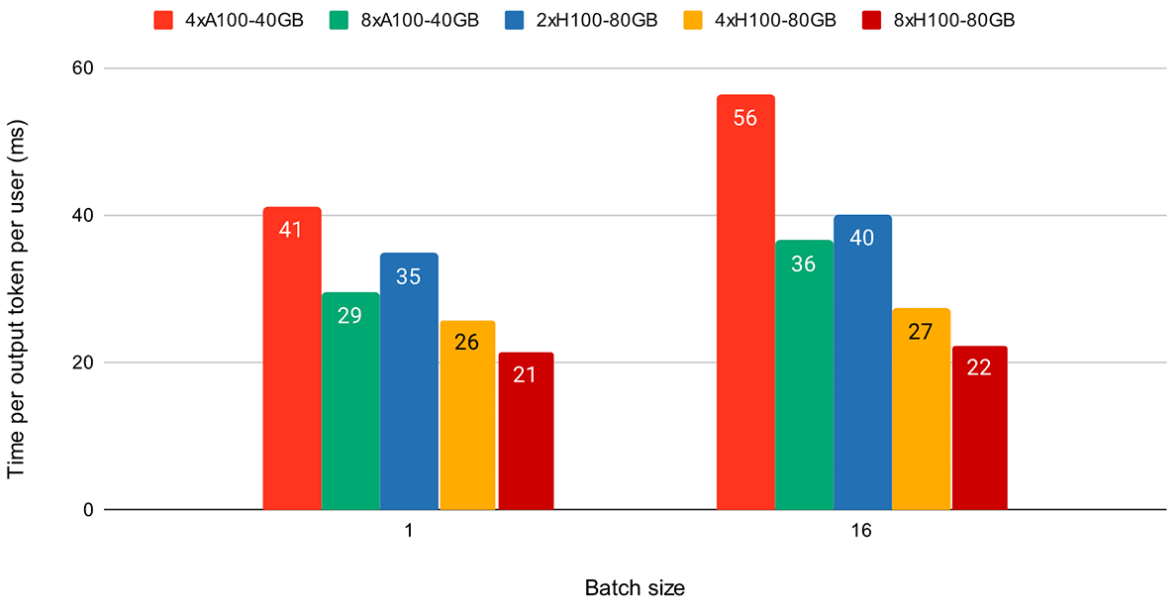

Рис. 2.5. Час формування вихідного маркера для користувача при масштабуванні Llama-v2-70B на кількох графічних процесорах.

Групування запитів під час оцінки графічного процесора збільшує пропускну здатність порівняно з послідовною обробкою запитів, але для виконання кожного запиту знадобиться більше часу (без урахування ефектів черги).

Існує кілька поширених методів групування запитів на формування висновків (рис. 2.6):

– статичне пакетування: клієнт пакує кілька підзапитів у запити, і відповідь повертається після завершення всіх послідовностей у пакеті.

– динамічне пакетування: запити групуються разом «на льоту» всередині сервера. Як правило, цей метод працює гірше, ніж статичне пакетування, але може бути близьким до оптимального, якщо відповіді короткі або мають однакову довжину. Погано працює, коли запити мають різні параметри.

– Безперервне пакетування: ідея групування запитів разом у міру їх надходження реалізована у методі SOTA. Замість того, щоб чекати завершення всіх послідовностей у пакеті, він групує послідовності разом на

рівні ітерації. Він може досягти в 10-20 разів кращої пропускної здатності, ніж динамічне дозування.

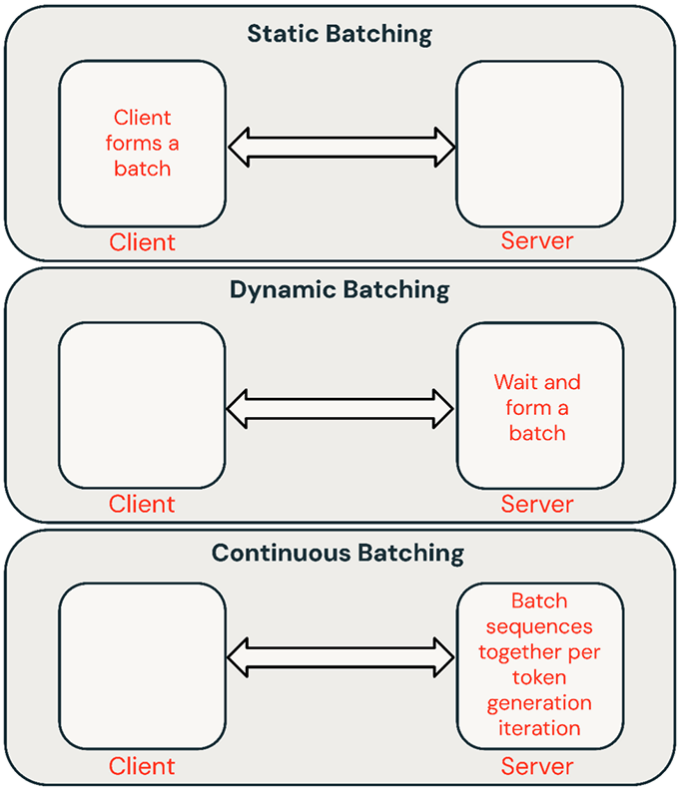

Рис. 2.6. Різні типи пакетування для моделей з генеративним алгоритмами

Пакетування є ефективним способом підвищення ефективності формування логічних відповідей. Безперервне пакетування, зазвичай, є найкращим підходом для спільних служб, але є ситуації, коли два інших можуть бути кращими.

У середовищах із низьким рівнем QPS динамічного пакетування може виникати ситуація перевершення по продуктивності безперервного пакетування. Іноді простіше реалізувати низькорівневу оптимізацію графічного процесора в простішій структурі пакетування. Для робочих навантажень офлайн-пакетного формування відповідей статичне пакетування може уникнути значних витрат і досягти кращої пропускної здатності.

Четвертий показник за яким виконувалось порівняння – розмір пакетів. Наскільки добре працює пакетування, сильно залежить від потоку запитів. Але можна отримати верхню межу його продуктивності, порівнявши статичне пакетування з однорідними запитами.

Таблиця 2.2

Пікова пропускна здатність MPT-7B із статичним пакетуванням і серверною частиною на основі FasterTransformers

	Розмір пакетів						
Апаратне забезпечення	1	4	8	16	32	64	128
1 x A10	0.4 (1x)	1.4 (3.5x)	2.3 (6x)	3.5 (9x)	OOM (Out of Memory) помилка		
2 x A10	0.8	2.5	4.0	7.0	8.0		
1 x A100	0.9 (1x)	3.2 (3.5x)	5.3 (6x)	8.0 (9x)	10.5 (12x)	12.5 (14x)	
2 x A100	1.3	3.0	5.5	9.5	14.5	17.0	22.0
4 x A100	1.7	6.2	11.5	18.0	25.0	33.0	36.5

У табл. 2.2 наведено експериментальні дані для запитів із 512 вхідних і 64 вихідних токенів. Для більших вхідних даних межа OOM буде при менших розмірах партії.

Затримка запиту збільшується із збільшенням розміру пакета. Наприклад, з одним графічним процесором NVIDIA A100, якщо максимізувати пропускну здатність із розміром пакета 64, затримка збільшується в 4 рази, а пропускна здатність – у 14 разів.

Спільні служби формування висновків, зазвичай, вибирають збалансований розмір пакету. Користувачі, які розміщують власні моделі, повинні збалансувати відповідний компроміс затримки/пропускної здатності для своїх програм.

У деяких програмах, зокрема чат-ботах, низька затримка для швидких відповідей є головним пріоритетом, в інших, наприклад, пакетна обробка неструктурованих PDF-файлів, можна пожертвувати затримкою для

опрацювання окремого документу і це забезпечить одночасне опрацювання всіх даних. На рис. 2.7 показана крива пропускної здатності та затримки для моделі 7B. Кожна лінія на цій кривій отримана шляхом збільшення розміру пакета від 1 до 256. Це корисно для визначення того, наскільки великим можна зробити розмір пакета з урахуванням різних обмежень затримки.

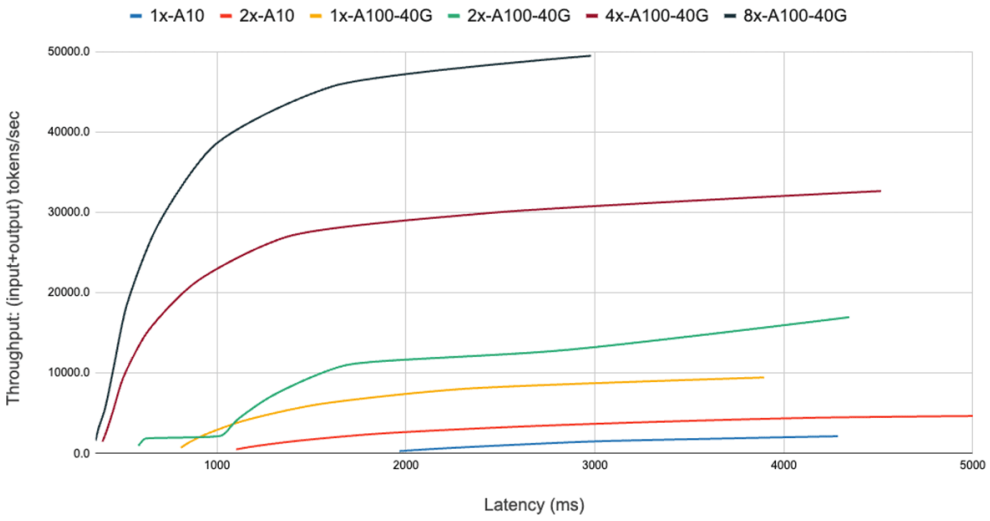

Рис. 2.7. Криві затримки пропускної здатності для моделі MPT-7B

Після певного розміру пакета, тобто коли відбувається перехід до режиму накладання обмежень на обчислення, кожне подвоєння розміру пакета лише збільшує затримку без збільшення пропускної здатності. Це дозволяє користувачам вибрати апаратну конфігурацію, яка відповідає їхнім вимогам щодо пропускної здатності в умовах обмеження затримки. Використовуючи паралелізм, важливо розуміти апаратні деталі низького рівня.

Наприклад, не всі екземпляри 8xA100 однакові в різних хмарах. Деякі сервери мають з’єднання з високою пропускною здатністю між усіма графічними процесорами, інші об’єднують графічні процесори в пари та мають з’єднання з меншою пропускною здатністю між парами. Це може створити вузькі місця, спричиняючи значне відхилення реальної продуктивності від наведених вище кривих.

Практичний приклад оптимізації – квантування. Квантування є загальноприйнятою технікою, яка використовується для зменшення вимог до апаратного забезпечення для формування відповідей від LLM.

Зменшення точності ваг моделі та активації під час логічного виведення може значно зменшити вимоги до апаратного забезпечення. Наприклад, перехід від 16-бітної ваги до 8-бітної ваги може удвічі зменшити кількість необхідних графічних процесорів у середовищах з обмеженим обсягом пам’яті (наприклад, Llama2-70B на A100s).

Зменшення до 4-бітних ваг дає змогу виконувати формування відповідей на користувацькому обладнанні (наприклад, Llama2-70B на Macbook). Як показує досвід, квантування слід застосовувати з обережністю. Наївні методи квантування можуть призвести до суттєвого погіршення якості моделі. Вплив квантування також залежить від архітектури моделі (наприклад, MPT проти Llama) і розміру.

Експериментуючи з такими методами, як квантування запропоновано використовувати тест якості LLM, зокрема Mosaic Eval Gauntlet, щоб оцінити якість системи логічного формування відповдіей, а не лише якість моделі окремо.

Крім того, важливо вивчити глибшу оптимізацію системи. Зокрема, квантування може зробити кеші KV набагато ефективнішими. Як згадувалося раніше, під час авторегресійної генерації токенів минулі ключі/значення (KV) з рівнів уваги кешуються замість того, щоб повторно обчислювати їх на кожному кроці.

Розмір кешу KV змінюється залежно від кількості послідовностей, що опрацьовуються за один раз, і довжини цих послідовностей. Крім того, під час кожної ітерації наступної генерації токенів нові елементи KV додаються до наявного кешу, збільшуючи його в міру генерації нових токенів. Таким чином, ефективне керування кеш-пам’яттю KV під час додавання цих нових значень має вирішальне значення для хорошої продуктивності генерації відповідей моделі.

Моделі Llama2 використовують варіант звернення уваги під назвою Grouped Query Attention (GQA). Варто відзначити, що коли кількість заголовків KV дорівнює 1, GQA є таким самим, як Multi-Query-Attention (MQA). GQA допомагає зменшити розмір кешу KV шляхом спільного використання ключів/значень. Формула для розрахунку розміру кешу KV така:

$$KV\_Size = batch\_size * seqlen * (d\_model/n\_heads) * n\_layers * 2\ (K\ and\ V) * 2\ (byte\ per\ Float16) * n\_kv\_heads \quad (2.5)$$

У табл. 2.3 показано розмір кешу GQA KV, розрахований для різних розмірів пакетів із довжиною послідовності 1024 токени. Для порівняння розмір параметра для моделей Llama2 становить 140 ГБ (Float16) для моделі 70B. Квантування кешу KV є ще одним методом (крім GQA/MQA) для зменшення його розміру, і необхідно активно оцінювати його вплив на якість генерації. Як згадувалося раніше, генерація токенів за допомогою LLM при малих розмірах пакетів є проблемою, пов'язаною з пропускною здатністю пам'яті графічного процесора, тобто швидкість генерації залежить від того, наскільки швидко параметри моделі можна перемістити з пам'яті графічного процесора в кеші на кристалі.

Таблиця 2.3

Розмір кешу KV для Llama-2-70B при довжині послідовності 1024

Розмір пакету	GQA ключ-значення кеш-пам'яті (FP16)	GQA ключ-значення кеш-пам'яті (Int8)
1	0.312 GiB	0.156 GiB
16	5 GiB	2.5 GiB
32	10 GiB	5 GiB
64	20 GiB	10 GiB

Перетворення ваг моделі з FP16 (2 байти) на INT8 (1 байт) або INT4 (0,5 байт) вимагає переміщення меншої кількості даних і, таким чином, прискорює генерацію маркерів. Однак квантування може негативно вплинути на якість генерації моделі.

Кожен із факторів, наведених вище, впливає на спосіб створення та розгортання моделей. Однак одержані результати важливі для того, щоб приймати рішення на основі даних, які враховують тип апаратного забезпечення, стек програмного забезпечення, архітектуру моделі та типові схеми використання. Тому необхідно визначити ціль оптимізації при застосуванні LLM моделей, зокрема, це стосується інтерактивної ефективності, максимізації пропускної здатності, мінімізація витрат. Тут є передбачувані компроміси. Також варто звернути увагу на компоненти затримки: для інтерактивних програм час до першого токена визначає, наскільки швидко реагуватиме сервіс, а час на вихідний токен визначає швидкість.

Пропускна здатність пам’яті є ключовою: генерація першого токена, зазвичай, пов’язана з обчисленнями, а подальше декодування – це операція, пов’язана з пам’яттю.

Оскільки LLM часто працює в налаштуваннях, обмежених пам’яттю, MBU є корисним показником для оптимізації та може використовуватися для порівняння ефективності систем логічного виведення.

Пакетування має вирішальне значення: одночасна обробка кількох запитів має вирішальне значення для досягнення високої пропускної здатності та ефективного використання дорогих графічних процесорів.

Для спільних онлайн-сервісів безперервне пакетування є необхідним, тоді як офлайн-пакетне навантаження може досягти високої пропускної здатності за допомогою простіших методів пакетування.

Стандартні методи оптимізації логічного виведення є важливими (наприклад, злиття операторів, вагове квантування) для LLM, але важливо вивчати більш глибоку оптимізацію систем, особливо ті, які покращують

використання пам’яті. Одним із прикладів є квантування кешу ключ-значення.

Конфігурації апаратного забезпечення: тип моделі та очікуване робоче навантаження слід використовувати для визначення апаратного забезпечення для розгортання. Наприклад, при масштабуванні до кількох графічних процесорів MBU падає набагато швидше для менших моделей, таких як MPT-7B, ніж для більших моделей, таких як Llama2-70B. Продуктивність також має тенденцію до сублінійного масштабування з вищими ступенями паралелізму тензорів. Тим не менш, високий ступінь тензорного паралелізму все ще може мати сенс для менших моделей, якщо трафік високий або якщо користувачі готові платити більше за надзвичайно низьку затримку.

Рішення на основі даних: розуміння теорії є важливим, але рекомендується завжди вимірювати наскрізну продуктивність сервера. Є багато причин, через які розгортання відповідей може працювати гірше, ніж очікувалося. MBU може бути неочікувано низьким через неефективність програмного забезпечення, або відмінності в апаратному забезпеченні між хмарними провайдерами можуть призвести до сюрпризів (різниця в затримці в 2 рази між серверами 8xA100 від двох хмарних провайдерів).

2.7. Висновки до розділу

Основні результати даного розділу полягають в наступному:

1. Проведено аналіз процесів генерування відповідей великими мовними моделями і встановлено, що основними з них є попереднє паралельне заповнення токенами і декодування. Це дало змогу визначити найбільш вузькі місця та критерії управління ефективністю використання програмно-апаратних ресурсів.

2. Запропоновано критерії продуктивності у процесі попереднього опрацювання запитів до моделей з генеративними алгоритмами: час

формування першого токену, час генерації вихідного токену для кожного користувача, прихована затримка формування відповіді користувачам та пропускної здатності моделі, що дало змогу керувати ресурсами і налаштуваннями інфраструктури апаратно-програмної інфраструктури і оптимально їх розподіляти.

3. Запропоновано метод та показники оптимізації використання ресурсів пам’яті та графічних процесорних ядер при формуванні висновків (відповідей) великими мовними моделями, що дало змогу здійснювати оптимальне налаштування інфраструктури і забезпечити баланс продуктивності та витрат на навчання та донавчання моделей.

4. Проведено порівняльний аналіз різних великих мовних моделей та використовуваної ними інфраструктури і запропоновано використання кешування ключ-значення, що дає можливість оптимізувати використання функції уваги при навчанні та функціонуванні моделей з генеративними алгоритмами.

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ЕКСПЕРИМЕНТІВ ЗАСТОСУВАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ З ГЕНЕРАТИВНИМИ АЛГОРИТМАМИ

3.1. Архітектура трансформерів

Архітектура LLM визначається низкою факторів. У першу чергу це стосується мети з якою будується конкретна моделі, наявності доступних обчислювальних ресурсів і типу завдань щодо опрацювання мови, які покладаються на LLM.

Загальна архітектура LLM формується з багатьох рівнів, зокрема, рівня передавання, інтеграції та уваги. Текст, який вбудовано всередину, об’єднується разом для створення прогнозів.

Важливими компоненти, які впливають на архітектуру LLM є:

– розмір моделі та кількість параметрів;
– внутрішнє представлення даних;
– механізми формування функції уваги;
– цілі навчання;
– ефективність обчислювальних ресурсів;
– декодування та генерація виводу.

Моделі на основі трансформерів (рис. 3.1), які зробили революцію в задачах опрацювання природної мови, зазвичай мають загальну архітектуру, яка включає такі основні компоненти:

– інтеграція вхідних даних;
– позиційне кодування;
– формування енкодера;
– механізм формування функції уваги;
– формування декодера;
– вихідні дані.

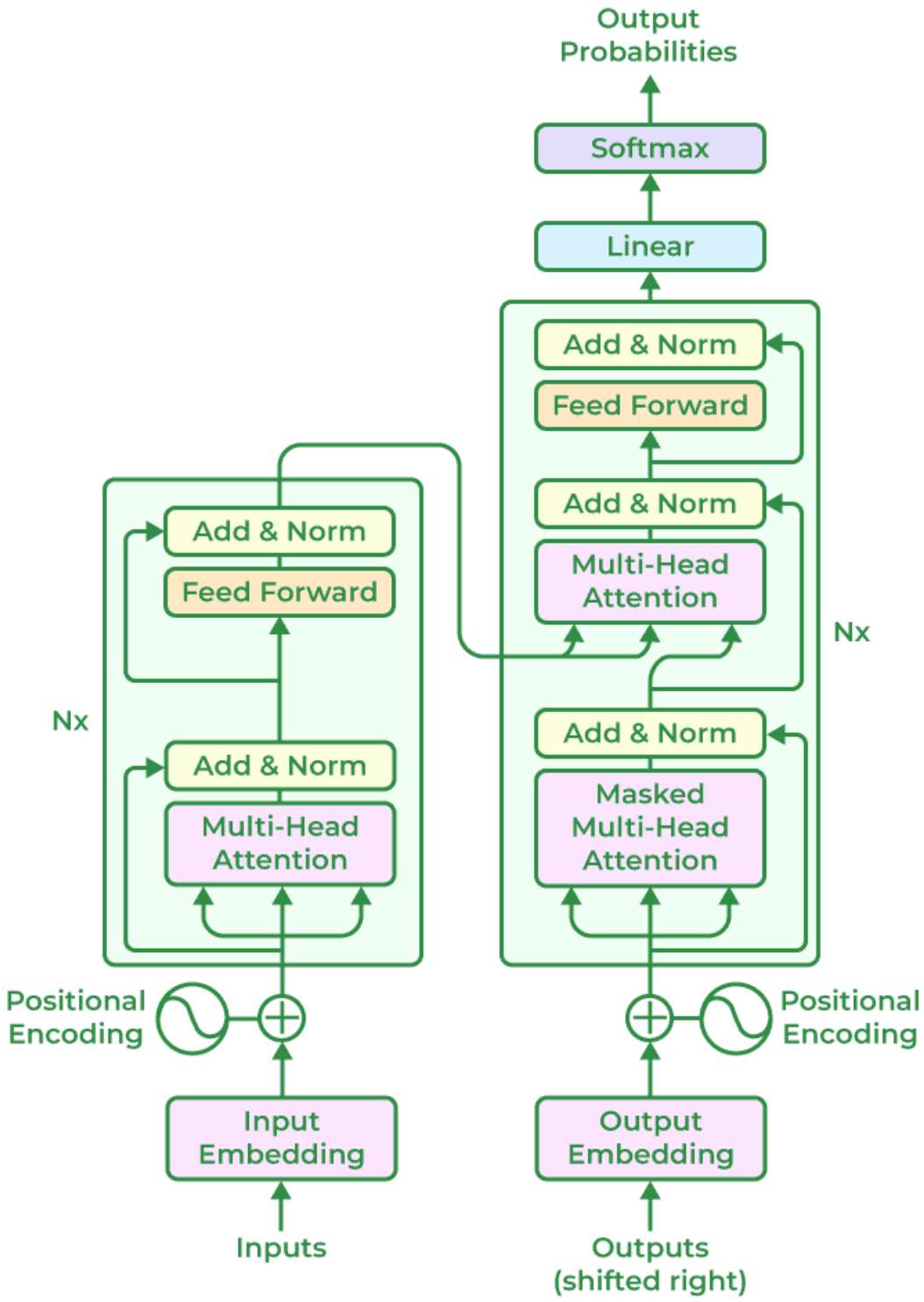

Рис. 3.1. Архітектура LLM моделей на основі трансформерів

Проведемо більш детальний аналіз компонентів архітектури LLM на основі траснформерів, який приведений на рис. 3.1.

При вбудовуванні (інтеграції) вхідних даних передбачається, що вхідний текст розбивається на менші одиниці, такі як слова або символи, і кожен маркер інтегрується в безперервне векторне представлення. Цей крок вбудовування фіксує семантичну та синтаксичну інформацію вхідних даних.

Позиційне кодування застосовується при додаванні та інтеграції вхідних даних для надання інформації про розташування токенів, оскільки трансформери природним чином не кодують порядок токенів. Це дозволяє моделі обробляти слова, враховуючи їх впорядкованість.

Енкодер (кодер), як компонент трансформерів, заснований на техніці нейронної мережі і забезпечує аналіз вхідного тексту, створюючи при цьому множину прихованих станів, які захищають контекст і значення текстових даних. Кілька шарів кодера складають ядро архітектури трансформера. Механізм самоконтролю та нейронна мережа прямого поширення є двома фундаментальними підкомпонентами кожного рівня кодера.

Кодери аналізують і обробляють введений текст, вилучаючи значущі представлення, які передають суть мови. Ці представлення кодують важливу інформацію про семантику, синтаксис і контекст послідовності слів. Аналізуючи вхідний текст на кількох рівнях, кодери вловлюють як локальні, так і глобальні залежності, дозволяючи LLM зрозуміти тонкощі мови. Структуру енкодера показано на рис. 3.2.

Механізм формування функції уваги, або по-іншому самоувага (Self-Attention Mechanism) дозволяє моделі зважувати важливість різних токенів у вхідній послідовності шляхом обчислення рівня уваги. Це дозволяє моделі розглядати залежності та зв’язки між різними токенами з урахуванням контексту.

Самоувага дозволяє LLM зосереджуватися на різних частинах вхідної послідовності під час обробки кожного слова. Під час активації уваги LLM призначають ваги різним словам на основі їх відповідності поточному слову, яке обробляється.

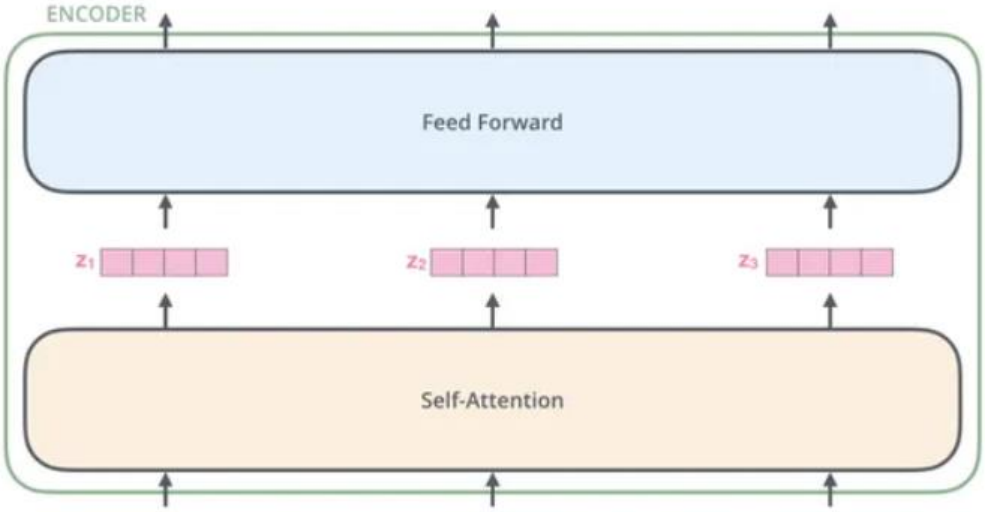

Рис. 3.2. Структура енкодера в архітектурі трансформерів

Цей динамічний механізм привернення уваги дає змогу LLM звертати увагу на важливу контекстну інформацію та ігнорувати нерелевантні або шумні частини вхідних даних. Вибірково звертаючи увагу на релевантні слова, LLMs можуть ефективно фіксувати залежності та добувати значущу інформацію, підвищуючи свої можливості розуміння мови. Організація цього блоку трансформерів приведена на рис. 3.3.

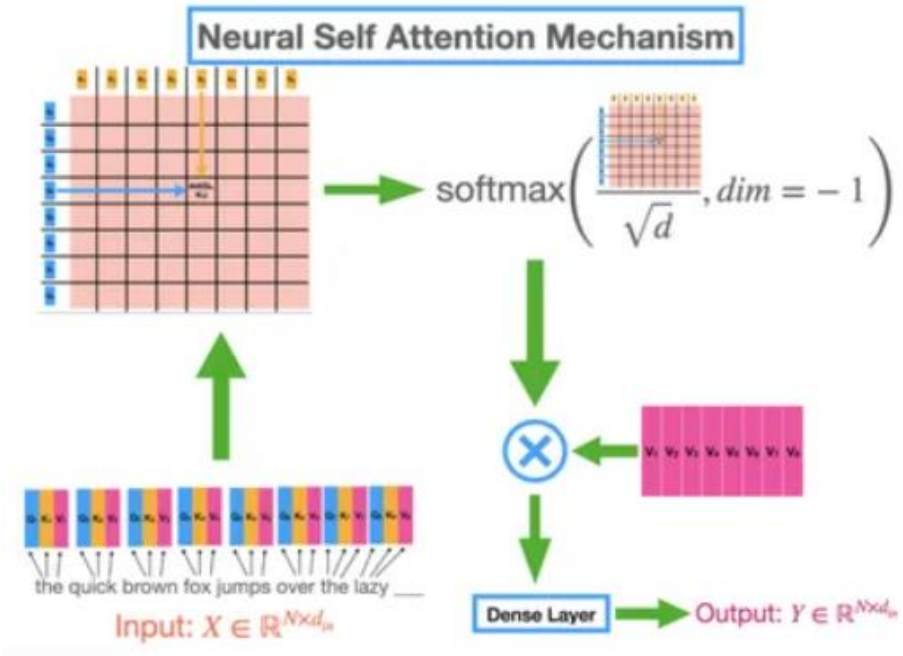

Рис. 3.3. Організація механізму формування уваги

Механізм формування уваги дозволяє трансформерам аналізувати важливість кожного слова в контексті всієї вхідної послідовності. Отже, залежності між словами можуть бути ефективно зафіксовані, незалежно від відстані. Ця здатність є цінною для розуміння нюансів значень, підтримки узгодженості та генерування релевантних контексту відповідей.

Після етапу самоконтролю нейронна мережа прямого поширення застосовується до кожного токена окремо. Ця мережа включає повністю пов’язані рівні з нелінійними функціями активації, що дозволяє моделі фіксувати складні взаємодії між токенами. Рівень декодера в архітектурі LLM імплементується у деяких моделях на основі трансформерів, що дає змогу забезпечувати принцип авторегресії, тобто модель може генерувати послідовні вихідні дані із врахуванням раніше згенерованих маркерів. Коли закодована інформація проходить через шари, вона досягає компонентів декодера. Організація декодера в архітектурі Transfortmers показана на рис. 3.4.

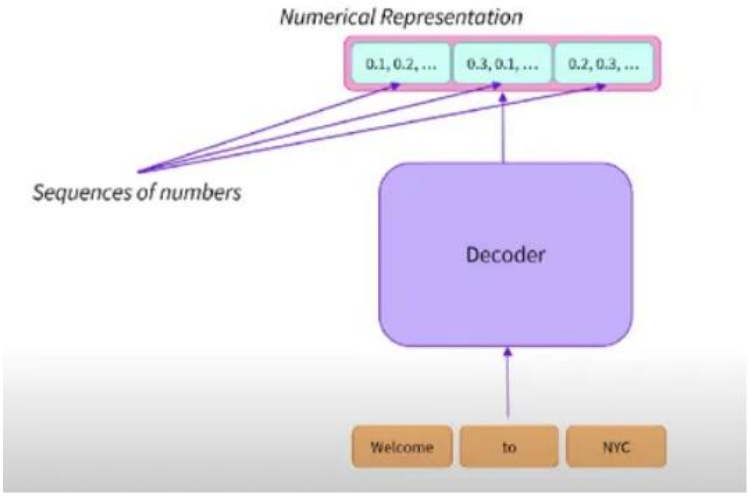

Рис. 3.4. Організація декодера

Механізми уваги в мовних моделях забезпечують динамічне та контекстне розуміння мови. Традиційні мовні моделі, такі як моделі n-грам, трактують слова як ізольовані одиниці, не враховуючи їхні зв’язки в реченні

чи документі. Навпаки, механізми уваги дозволяють призначати різні ваги різним словам, фіксуючи їх релевантність у даному контексті. Зосереджуючись на суттєвих термінах та ігноруючи нерелевантні, механізми уваги допомагають мовним моделям точніше зрозуміти глибинне значення тексту (рис. 3.5).

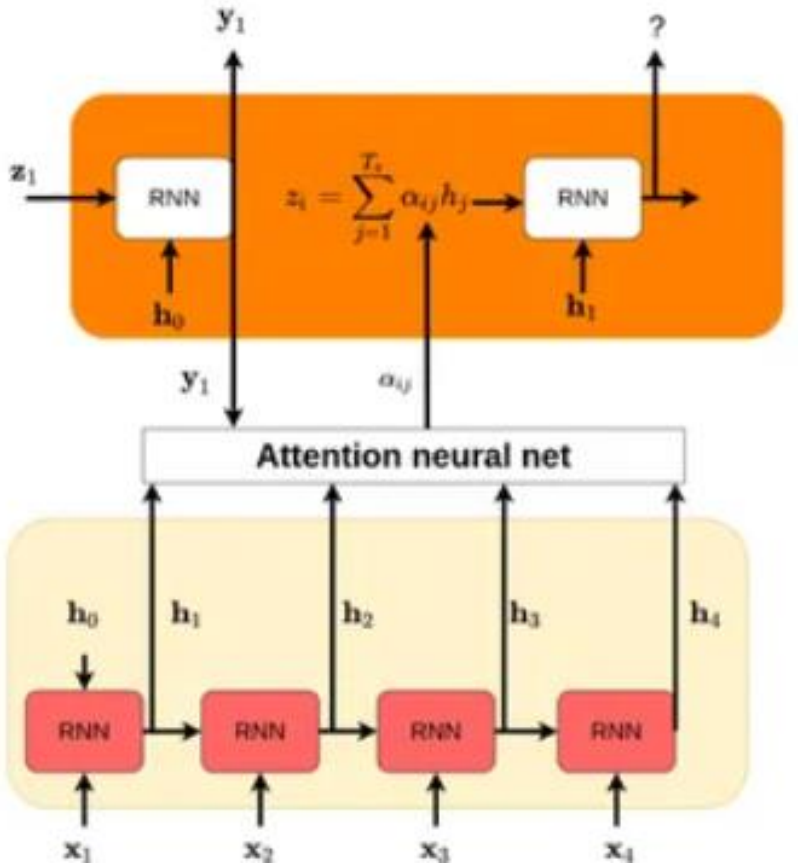

Рис. 3.5. Імплементація механізмів уваги у шарах трансформера

Однією з найважливіших переваг механізмів уваги є їхня здатність призначати різні ваги різним словам у реченні. Під час обробки коментаря мовна модель обчислює його відповідність іншим словам у контексті, враховуючи їхні семантичні та синтаксичні зв’язки. Наприклад, у реченні «Кіт сидів на килимку» мовна модель, що використовує механізми уваги, присвоїть більшу вагу «кіт» і «килимок», оскільки вони більше відповідають дії сидіння. Ця зважена релевантність дозволяє мовній моделі

віддавати пріоритет найважливішій інформації, ігноруючи нерелевантні деталі, що забезпечує більш повне розуміння контексту.

Мова часто включає в себе залежності, які охоплюють кілька слів або навіть речень. Механізми уваги чудово вловлюють ці довгострокові залежності, дозволяючи моделі бездоганно з'єднувати частини мови.

Звертаючи увагу на різні частини вхідної послідовності, мовні моделі можуть навчитися встановлювати значущі зв'язки між словами, які розташовані далеко одне від одного в реченні. Ця здатність є цінною в таких завданнях, як машинний переклад, де вкрай важливо підтримувати узгодженість і розуміння контексту на великих відстанях.

Окрім цього, у трасформерах передбачено визначення рівня нормалізації, процедура визначення якого застосовується після кожного підкомпонента або рівня в його архітектурі. Це дає змогу стабілізувати процес навчання та покращує здатність моделі узагальнювати різні вхідні дані.

Вихідні шари моделі трансформера можуть відрізнятися залежно від конкретного поставленого завдання. Наприклад, в процесі моделювання мови лінійна проекція з подальшою активацією SoftMax зазвичай використовується для створення розподілу ймовірностей за наступним маркером. Важливо мати на увазі, що фактична архітектура моделей на основі трансформерів може змінюватися та вдосконалюватися на основі конкретних досліджень і створення моделей. Для виконання різних завдань і цілей деякі моделі, такі як GPT, BERT і T5, можуть інтегрувати більше компонентів або модифікацій.

На сьогодні реальними прикладами реалізації LLM є продукти, розроблені такими компаніями, як OpenAI, Google, Facebook, Microsoft та групами незалежних дослідників.

GPT – 3 представляє собою попередньо навчений генеративний трансформер, який розроблений компанією Open AI, найбільш популярним застосування цієї моделі є Chat GPT.

BERT є представленням та екземпляром двонаправленого енкодера, що реалізує принцип трансформерів і розроблений компанією Google для розв'язку задач опрацювання природної мови і зазвичай використовується для різноманітних завдань, пов'язаних із опрацюванням природною мовою. Крім того, його можна використовувати для створення вставок для певного тексту, можливо, для навчання іншої моделі.

RoBERTa є розвитком BERT і надійно оптимізованим підходом попереднього навчання. У серії спроб покращити продуктивність архітектури трансформеа RoBERTa, Facebook AI Research реалізував вдосконалену версію моделі BERT.

BLOOM – це перший багатомовний LLM, створений асоціацією різних організацій і дослідників, які об'єднали свій досвід для розробки цієї моделі, яка схожа на архітектуру GPT-3. Для більш глибокого дослідження цих моделей можна скористатися офіційними матеріалами, розміщеними на ресурсах компаній-розробників. Практично усіх їх можна використовувати за допомогою платформ з відкритим кодом, таких як Hugging Face of Open AI.

Основною причиною такого захоплення LLM є їхня ефективність у різноманітних завданнях, які вони можуть виконувати. З наведених вище матеріалів і технічної інформації про LLM, зрозуміло, що Chat GPT також є LLM. Основні функції, які можна покласти на цю модель, а також задачі, які зможе вирішувати Chat GPT наведено нижче.

Генерація коду – один із найбожевільніших варіантів використання служби Chat GPT полягає в тому, що модель може генерувати досить точний код для конкретного завдання, описаного користувачем.

У випадку, коли фахівцям складно налагодити певний фрагмент програмного коду, ChatGPT допоможе у цьому, оскільки він може формувати підказки щодо того, яка стрічка коду створює проблеми, а також спосіб їх усунення.

Окрім цього, тепер не потрібно витрачати години на написання документації до проекту. Цю місію можна перекласти на ChatGPT.

Коли з’явилися персональні віртуальні помічники на основі штучного інтелекту, люди ставили їм неоднозначні запитання і одержували адекватні відповіді на них.

Великі мовні моделі можна застосовувати для перекладу з однієї мови на іншу, зокрема, GPT підтримує понад 50 мов, а також дає змогу виправити граматичні помилки у тексті.

Випадки використання LLM не обмежуються вищезазначеним. Сьогодні спостерігається зародження та розвиток нової сфери Prompt Engineering, що є абсолютно новою темою в академічних колах та для практиків.

Великі мовні моделі створюються на основі складних архітектур трансформерів і розробляються місяцями з мільйонними витратами на їх навчання та забезпечення відповідної платформи для формування результатів. Лише з цих причин рекомендується використовувати попередньо навчені моделі, надані багатьма організаціями з відкритим кодом, щоб використовувати ці моделі для персоналізованих завдань. Тому актуальними є дослідження у напрямку Transfer Learning, зокрема, щодо організації платформ передачі знань.

Сьогодні розроблено кілька платформ, які дають змогу використовувати різні аспекти цих платформ, які пропонують LLM на основі API для легкого доступу та використання.

ChatGPT на сьогодні доступний як веб-додаток із простим у використанні інтерфейсом. Hugging Face також надає API для попередньо навчених моделей у своєму центрі для точного налаштування та формування висновків. BLOOM є прикладом такого LLM, який добре опрацьовує завдання щодо розпізнавання природної мови приблизно 46 природними мовами та 13 мовами програмування.

NVIDIA пропонує різноманітні сервіси для простого керування LLM, які можуть варіюватися від доменно-спеціальних LLM. До них, наприклад, належить NVIDIA BioNemo, що є частиною фреймворку NVIDIA Nemo, яка призначена для створення LLM.

Розглянемо різницю між NLP і LLM, оскільки обидві сфери стосуються опрацювання природної мови і штучного інтелекту, а також використовують подібні алгоритми.

Напрям опрацювання природної мови є більш широкою діяльністю, ніж LLM, яка складається з алгоритмів і технік. NLP керується двома підходами –машинним навчанням і аналізом мовних даних, а також застосовується для вирішення завдань покращення пошуку та його опимізації, аналізу та систематизації великих документів, аналітики соціальних мереж. Оскільки, LLM – це велика мовна модель, яка більш специфічна для людського тексту, то вона забезпечує створення контенту та формування персоналізованих рекомендацій.

Немає жодних сумнівів у можливостях LLM у майбутньому, і ця технологія є частиною більшості додатків на базі штучного інтелекту, які використовуватимуться багатьма користувачами щодня.

Але є й деякі недоліки LLM. Для успішного навчання великої мовної моделі потрібні мільйони доларів, щоб створити таку велику обчислювальну потужність, яка може навчити модель, використовуючи паралельну продуктивність. Для цього потрібні місяці навчання, а потім детального точного налаштування моделей для досягнення кращої продуктивності.

Отримання великого обсягу текстового корпусу може бути складним завданням, оскільки, наприклад, ChatGPT звинувачують лише в навчанні на даних, які були незаконно зібрані, і створенні програми для комерційних цілей. В епоху глобального потепління та зміни клімату не варто забувати про вуглецевий слід LLM. У деяких джерелах [6-8] зазначають, що підготовка однієї моделі ШІ з нуля має вуглецевий слід, який дорівнює

вуглецевому сліду п’яти автомобілів за весь термін їх служби, що є справді серйозною проблемою.

У зв’язку з труднощами, з якими стикаються під час підготовки LLM, перехід навчання значною мірою просувається, щоб позбутися всіх проблем, обговорених вище. LLM має можливість внести революцію в додатки на основі штучного інтелекту, але прогрес у цій галузі здається дещо складним, тому що просте збільшення розміру моделі може підвищити її продуктивність, але через певний час настане насичення в продуктивності та виклики для обробки цих моделей буде більшим, ніж підвищення продуктивності, досягнуте подальшим збільшенням розміру моделей.

3.2. Принципи організації, налаштування і застосування попередньо навчених моделей машинного навчання

Мовні моделі мають унікальний процес навчання, який дає їм змогу розуміти та генерувати мову досконалого рівня. Цей процес складається з двох ключових етапів: попереднього навчання та тонкого налаштування. Програмний код попереднього навченої моделі на основі трансформерів показано на рис. 3.6 у вигляді Python коду.

```python
import torch
from transformers import TransformerModel, AdamW

# Load the pretrained Transformer model
pretrained_model_name = 'bert-base-uncased'
pretrained_model = TransformerModel.from_pretrained(pretrained_model_name)

# Example input
input_ids = torch.tensor([[1, 2, 3, 4, 5]])

# Get the output from the pretrained model
outputs = pretrained_model(input_ids)

# Access the last hidden states or pooled output
last_hidden_states = outputs.last_hidden_state
pooled_output = outputs.pooler_output
```

Рис. 3.6. Програмний код використання попередньо навчених моделей

Після того, як за допомогою програмного коду, представленого на рис. 3.6, з'явилось загальне розуміння мови на основі підходу попереднього навчання, вони переходять на етап тонкого налаштування, де відбувається адаптація до конкретних завдань або областей.

Тонке налаштування передбачає надання LLM розмічених даних, що стосуються конкретного домену, наприклад аналіз емоцій або відповіді на запитання. Ці дані з мітками дозволяють асесорам адаптувати свої попередні знання до конкретних нюансів і вимог завдання. Приклад тонкого налаштування параметрів LLM представлено на рис. 3.7.

```
import torch
from transformers import TransformerModel, AdamW

# Load the pretrained Transformer model
pretrained_model_name = 'bert-base-uncased'
pretrained_model = TransformerModel.from_pretrained(pretrained_model_name)

# Modify the pretrained model for a specific downstream task
pretrained_model.config.num_labels = 2  # Number of labels for the task

# Example input
input_ids = torch.tensor([[1, 2, 3, 4, 5]])
labels = torch.tensor([1])

# Define the fine-tuning optimizer and loss function
optimizer = AdamW(pretrained_model.parameters(), lr=1e-5)
loss_fn = torch.nn.CrossEntropyLoss()

# Fine-tuning loop
for epoch in range(num_epochs):
    # Forward pass
    outputs = pretrained_model(input_ids)
    logits = outputs.logits

    # Compute loss
    loss = loss_fn(logits.view(-1, 2), labels.view(-1))

    # Backward pass and optimization
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Print the loss for monitoring
    print(f"Epoch {epoch+1}/{num_epochs} - Loss: {loss.item():.4f}")
```

Рис. 3.7. Тонке налаштування моделі трансформерів під дані конкретного домену

Під час тонкого налаштування мовні моделі удосконалюють своє розуміння мови та можливості генерації, спеціалізуючись на предметно-специфічних мовних шаблонах і контекстних нюансах. Навчаючись на розмічених даних забезпечується глибше розуміння тонкощів конкретної предметної області, що дозволяє їм надавати більш точні та відповідні контексту відповіді.

Перевага цього двоетапного процесу навчання полягає в його здатності використовувати силу даних. Попереднє навчання величезній кількості текстових даних без міток дає мовній моделі загальне розуміння мови, тоді як тонке налаштування даних з мітками вдосконалює їхні знання для конкретних доменів. Ця комбінація дає змогу володіти широкою базою знань, досягаючи успіхів у певних областях, пропонуючи точне і адекватне розуміння мови та здатність до генерування.

3.3. Імплементація прямого доступу до різних LLM

3.4. Модель GPT-3

GPT-3, Generative Pre-trained Transformer, став новаторською архітектурою мовної моделі, яка революціонізувала розуміння та генерацію природної мови. Архітектура GPT-3 побудована на основі моделі Transformer, що включає багато параметрів для досягнення виняткової продуктивності.

GPT-3 містить стек шарів енкодерів трансофрмера. Кожен рівень складається з розподілених механізмів уваги та нейронної мережі прямого поширення. Механізм уваги дозволяє моделі фіксувати залежності та зв’язки між словами, у той час як мережі прямого поширення обробляють і перетворюють закодовані представлення. Ключова інновація GPT-3 полягає в його величезному розмірі з надзвичайно великою кількістю параметрів, які дозволяють фіксувати величезні знання мови. Архітектура GPT-3 показана на рис. 3.8.

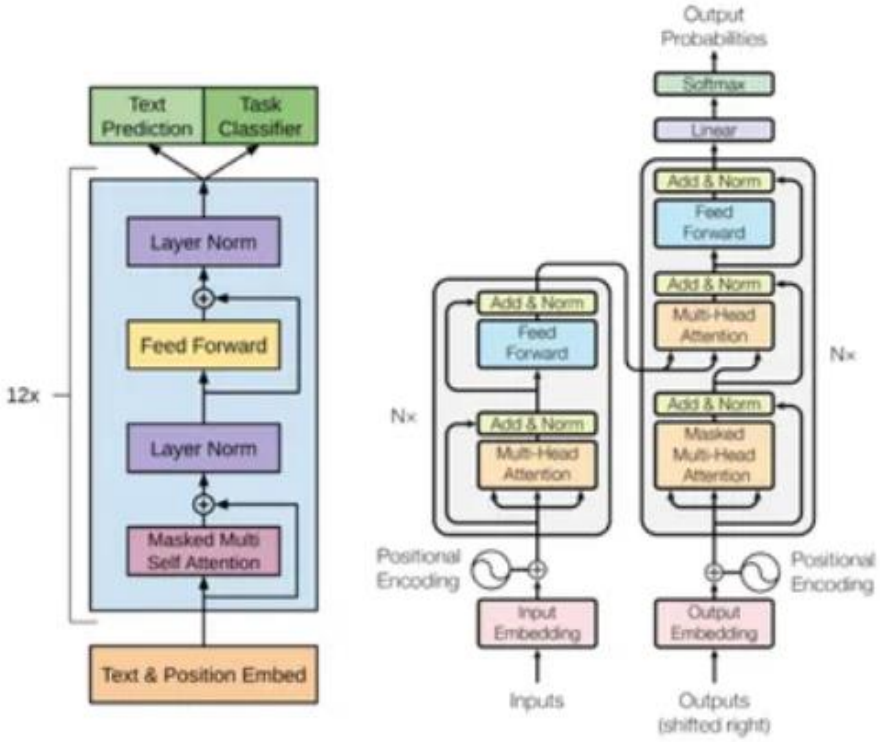

Рис. 3.8. Архітектура GPT-3

Для того, щоб забезпечити взаємодію через прикладний програмний інтерфейс з моделлю openAI GPT-3 та генерацію текстів можна використати програмний код, який представлено на рис. 3.9.

```
import openai

# Set up your OpenAI API credentials
openai.api_key = '[redacted]'

# Define the prompt for text generation
prompt = "Наскільки популярним є Тернопільський національний технічний університет імені Івана Пулюя"

# Make a request to GPT-3 for text generation
response = openai.Completion.create(
  engine="text-davinci-003",
  prompt=prompt,
  max_tokens=500,
  temperature=0.5
)

# Retrieve the generated text from the API response
generated_text = response.choices[0].text

# Print the generated text
print(generated_text)
```

Рис. 3.9. Організація доступу до GPT-3для генерації тексту

У результаті виконання скрипта, представленого на рис. 3.9 одержано результат, який показано на рис. 3.10.

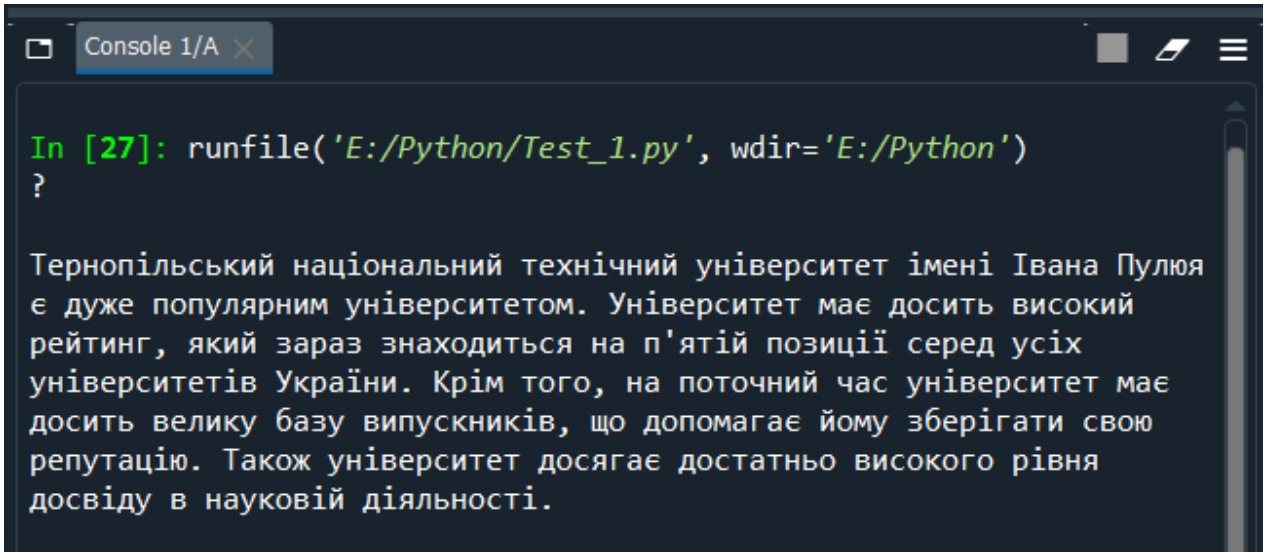

Рис. 3.10. Результат генерації відповіді GPT-3

Далі варто розглянути ще один з варіантів реалізації великих мовних моделей T5 (Text-to-Text Transfer Transformer).

3.5. Модель Text-to-Text Transfer Transformer

T5 представляє новаторський прогрес в архітектурі мовної моделі. Він використовує уніфікований підхід до різних завдань обробки природної мови, створюючи їх як перетворення тексту в текст. Цей підхід дозволяє одній моделі виконувати кілька завдань, включаючи класифікацію тексту, формування висновків та відповідей на запитання. Завдяки об’єднанню архітектур, що відповідають конкретним завданням, в єдину модель, T5 досягає вражаючої продуктивності та ефективності, спрощуючи процес розробки та розгортання моделі.

Архітектуру T5 побудовано на основі трансформера, яка складається зі структури енкодера-декодера. На відміну від традиційних моделей, які точно налаштовуються для конкретних завдань, T5 навчається з використанням багатозадачної цілі, де різноманітний набір функцій виконується як перетворення тексту в текст.

Під час навчання модель вчиться відображати вхідний і вихідний текст, що робить її дуже адаптивною та здатною виконувати широкий спектр завдань природного опрацювання мови, включаючи класифікацію тексту, анотування, переклад тощо. Принцип та алгоритм використання T5 показано на рис. 3.11.

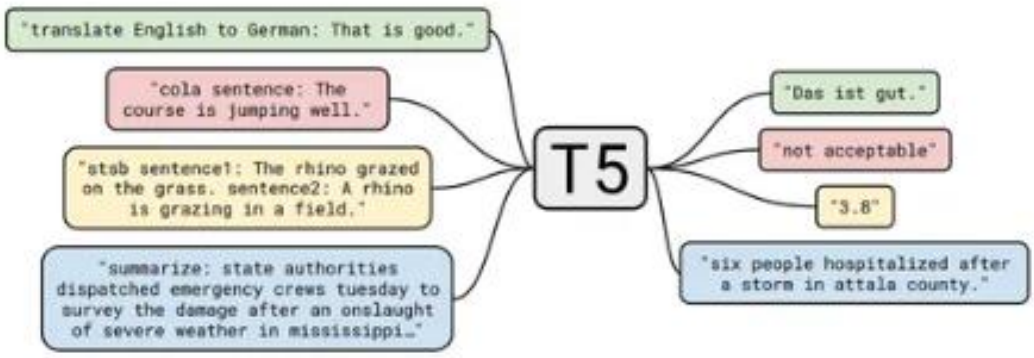

Рис. 3.11. Функціональність T5

Бібліотека transformers, яка пропонує простий інтерфейс для взаємодії з різними моделями трансформерів, включаючи T5, може використовувати модель T5 у Python. На рис. 3.12 проілюстровано приклад, як використовувати T5 для виконання завдань перетворення тексту в текст.

```
 from transformers import T5Tokenizer, T5ForConditionalGeneration

tokenizer = T5Tokenizer.from_pretrained("t5-small")
model = T5ForConditionalGeneration.from_pretrained("t5-small")

input_ids = tokenizer("translate English to German: The house is wonderful.",
      return_tensors="pt").input_ids

# Generate the translation using T5
outputs = model.generate(input_ids)

# Print the generated text
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Рис. 3.12. Застосування T5 при перетворенні текст в текст

3.6. Модель BERT

BERT (Bidirectional Encoder Representations) здійснив революційний зсув у розумінні змісту тексту. Використовуючи двонаправлене навчання, BERT фіксує контекст як зліва, так і справа, що дозволяє глибше зрозуміти семантику тексту.

BERT значно підвищив продуктивність у таких задачах, як розпізнавання іменованих об’єктів, аналіз настроїв і висновків на основі опрацювання природної мови. Його здатність враховувати нюанси мови з тонким розумінням контексту зробила його наріжним каменем у сучасній обробці текстів.

BERT складається зі стеку шарів енкодера трансформатора (рис. 3.13). Він використовує двонаправлене навчання, дозволяючи моделі охоплювати контекст як з лівого, так і з правого контекстів.

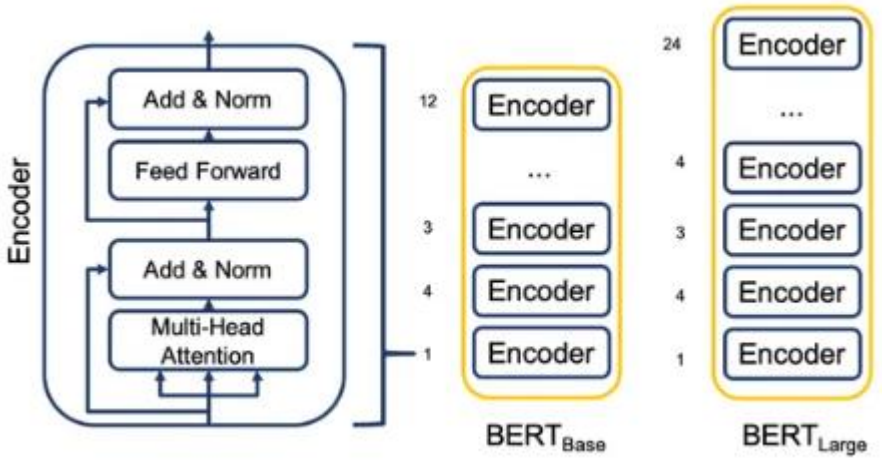

Рис. 3.13. Архітектура BERT

Цей двонаправлений підхід забезпечує глибше розуміння семантики мови. Це також дозволяє BERT досягати успіхів у задачах розпізнавання іменованих об’єктів, аналізу емоцій, відповідей на запитання тощо. BERT також містить унікальні маркери, включаючи [CLS] для класифікації та [SEP] для розділення речень або меж документів.

Бібліотека transformers пропонує простий інтерфейс для взаємодії з різними моделями. Вона також містить BERT і може використовуватися з Python. На рис. 3.14 показано, як можна використовувати BERT.

```
from transformers import BertTokenizer, BertForSequenceClassification

# Load the BERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Define the input text
input_text = "Hello, my dog is cute"

# Tokenize the input text and convert into Pytorch tensor
input_ids = tokenizer.encode(input_text, add_special_tokens=True)
input_tensors = torch.tensor([input_ids])

# Make the model prediction
outputs = model(input_tensors)

# Print the predicted label
print("Predicted label:", torch.argmax(outputs[0]).item())
```

Рис. 3.14. Використання BERT

Внутрішня робота LLMs розкриває складну архітектуру. Таким чином, дозволяючи цим моделям розуміти та створювати тексти з дуже високою точністю та універсальністю.

Кожен компонент має вирішальне значення для розуміння та генерування текстових повідомлень, від трансформерів і механізмів самоконтролю до багатошарових енкодерів і декодерів.

3.7. Висновки до розділу

Основні результати даного розділу полягають в наступному:

1. Проаналізовано архітектуру класу великих мовних моделей на основі трансформерів і встановлено, що основними їх компонентами є нейронні мережі з енкодерами та декодерами, що дало змогу обгрунтувати

доцільність застосування запропонованих показників ефективності та методу оптимального управління інфраструктурою при застосуванні підходу передачі знань з метою донавчання моделей.

2. Досліджено принципи організації, налаштування і застосування попередньо навчених моделей машинного навчання, що дало можливість обирати оптимальних провайдерів хмарних сервісів для подальшої адаптації моделей для користувацьких предметних областей.

3. Програмно імплементовано механізми прямого доступу до моделей з генеративними алгоритмами машинного навчання та експериментально доведено їх ефективність. Визначено необхідність та доцільність проведення додаткових досліджень щодо вибору оптимальних платформ для підтримки підходу transfer learning з врахуванням запропонованих критеріїв та методу оптимізації використання ресурсів програмно-апаратної інфраструктури.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

У кваліфікаційній роботі магістра досліджено методи та засоби створення і розгортання інфраструктури комп’ютерних систем за допомогою генеративних алгоритмів машинного навчання. Під час розв’язання задач дослідження, особливо практичної реалізації, враховано вимоги з охорони праці і техніки безпеки, пожежної та електробезпеки.

Виконання як теоретичної частини роботи, так і практичної, передбачає використання комп’ютерної техніки та обладнання з низькими напругами і силою струму.

В якості регламентуючого документа з пожежної безпеки перед початком роботи над методами і засобами створення і розгортання інфраструктури комп’ютерних систем за допомогою генеративних алгоритмів машинного навчання використано вимоги «Типового положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України», які є діючим на даний час і затверджені постановою Кабінету міністрів України від 26 червня 2013 р. № 444.

Для організації захисту від негативного впливу екранів дотримано вимог Закону України "Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями" та НПАОП 0.00-7.15-18, який затверджений наказом Міністерства соціальної політики України 14.02.2018 N207.

Робоче місце під час виконання кваліфікаційної роботи та проектування рішень щодо інфраструктури для функціонування генеративних алгоритмів облаштовано згідно наведених вимог та відповідає організаційним, ергономічним та вимогам з пожежної безпеки.

Електробезпеку робочого місця регламентують Правила безпечної експлуатації електроустановок споживачів, які затверджені наказом Держнаглядохоронпраці від 09.01.98 N 4, зареєстрованих у Міністерстві юстиції України 10.02.98 за N 93/2533 (НПАОП 40.1-1.21-98) [23].

При виконанні кваліфікаційної роботи магістра використовувана електромережа відповідає правилам [24]:

− живлення електромережі проектовано, як окрему групову трьох провідну мережу з використанням фази, робочого «нуля» та захисного «нуля»;

− захисний «нуль» застосовано для реалізації заземлення електропристроїв;

− усі електричні та електронні пристрої мають захист від короткого замикання та непередбачуваних аварійних ситуацій;

− монтаж та експлуатація електромережі задовольняють вимогам щодо унеможливлення виникнення джерела загоряння через коротке замикання та перевантаження;

− усі лінії електроживлення виконанні не з легкозаймистого матеріалу або з негорючою ізоляцію;

− електричне устаткування підключено до мережі лише за допомогою справних штепсельних з’єднань і розеток заводського виготовлення;

− у розетках і штепселях передбачено контакти заземлення.

Вимоги електробезпеки при виконанні проектних робіт кваліфікаційної роботи дотримано двома шляхами: використання безпровідних технологій передачі даних і напруги живлення в діапазоні 3.3В і 5 В, що дозволяє зменшити можливість ураження струмом при виникненні контакту з мережею чи в аварійних ситуаціях.

Щодо пожежної безпеки будівлі, де виконувався проект і приміщення його потенційної експлуатації, то дотримано вимог державних будівельних

норм ДБН В.1.1-7-2016, а також правилами пожежної безпеки ДСТУ Б.В.1.1-36:2016.

У приміщеннях, де розташовуються робочі місця користувачів ПК потрібно забезпечити відповідність вимогам санітарних норм і правил [23].

Крім цього, на робочих місцях, обладнаних комп'ютерами і периферійною технікою забезпечено оптимальні значення параметрів мікроклімату: температури, руху повітря та відносної вологості, у відповідності до вимог нормативних документів.

Щодо освітлення, то приміщення де експлуатуються ПК, повинно бути обладнаним джерелами штучного освітлення та мати природне освітлення. Нормативний документ, який регламентує вимоги до рівнів природного і штучного освітлення – ДБН В.2.5-28-2018.

Природне освітлення забезпечують прозорі вікна та інші світлові прорізи, що знаходяться на півночі або північному сході. У приміщеннях коефіцієнт природного освітлення повинен бути не нижче ніж 1,5%. Розрахунок коефіцієнта природного освітлення виконують згідно методики, яка наведена у ДБН В.2.5-28-2018.

Штучне освітлення у приміщеннях з ПК забезпечується за допомогою системи загального освітлення, переважно рівномірного. В якості штучного джерела світла застосовуються люмінесцентні лампи типу ЛБ.

При використанні ПК для розробки рішень щодо створення і розгортання інфраструктури на основі моделей з генеративними алгоритмами було дотримано наступних вимог з техніки безпеки:

- не виконувався самостійний ремонт ПК і периферійних пристроїв;
- не вносились конструктивні чи інші зміни в апаратне забезпечення комп'ютера;
- використовувались тільки ті матеріали та предмети, які стосувались розробки комп'ютерної системи оптимізації потоку звернень користувачів.

Для забезпечення вимог щодо безпечної експлуатації інформаційних технологій та мереж дотримано вимог СТУ EN 60950-1:2015 «Обладнання інформаційних технологій. Безпека. Частина 1. Загальні вимоги» (ДСТУ EN 60950-1:2015).

Таким чином, дотримання вимог і норм з охорони праці, а також правил техніки безпеки дали можливість мінімізувати негативний вплив комп'ютерної техніки на здоров'я виконавця і забезпечити імплементацію методів і засобів створення і розгортання інфраструктури на основі моделей з генеративними алгоритмами машинного навчання.

4.2. Застосування основних способів та засобів в ході проведення невідкладних аварійно-рятувальних робіт на промисловому підприємстві

Аварійно-рятувальні роботи (АРР) на промисловому підприємстві – це першочергові заходи на території, де сталася надзвичайна ситуація (НС), з пошуку і рятування персоналу, матеріалів та устаткування, що має суттєву матеріальну цінність, сукупність робіт по обмеженню та гасінню пожеж, аварійного відключення джерел рідкого палива, газу, електроенергії та води, та, в тому числі, надання потерпілим працівникам невідкладної допомоги медичного характеру і в разі потреби їх евакуації в спеціалізовані медичні установи поза зоною проведення АРР [24].

Невідкладні роботи – це заходи першочергового характеру на території, де сталася надзвичайна ситуація, із всебічного забезпечення АРР, усунення окремих вогнищ (причин) підвищеної небезпеки, локалізації аварій і ушкоджень на енергетичних мережах, надання першочергової допомоги медичного характеру, забезпечення мінімальних умов для персоналу, а також роботи по санітарній очистці та знезараженню територій.

Надзвичайна ситуація (НС) на промисловому підприємстві – це подія на виробничому об'єкті, яка сталася внаслідок техногенної аварії,

метеоявища небезпечного характеру, катастрофи, катаклізму, який спричинений природною стихією, що може викликати або вже призвів до смерті людей, погіршення здоров'я персоналу або стану довкілля, суттєвих матеріальних втрат і негативного впливу на життєдіяльність працівників.

Область НС – це територія, де сталася така ситуація [25]. Аварійно-рятувальні та інші невідкладні заходи (АРІНЗ) на підприємстві включають в себе три етапи:

1. Вжиття екстрених заходів:

1.1. Екстрений захист працівників: – своєчасне інформування посадових осіб і уповноважених служб про загрозу настання НС і її розвитку, а також інструктаж працівників про порядок дій у екстреній ситуації;

– використання засобів захисту, впорядковане вилучення працівників із території, де трапилась НС в безпечні місця, введення встановлених режимів поведінки, проведення заходів медичного захисту;

– розшук та вилучення постраждалих та надання їм медичної допомоги.

1.2. Запобігання розвитку і зменшення небезпечних впливів НС:

– локалізація аварії;

– перекриття і глушіння (припинення дії) джерела небезпечних речовин;

– припинення (екстрене відключення) технологічних процесів.

1.3. Підготовчий етап виконання робіт:

– мобілізація служб міської ланки територіальної організації попередження і дій при виникненні НС;

– попереднє оцінювання ситуації і координування комплексного обстеження в зоні НС;

– виїзд оперативних груп сил міської та окружних ланок територіальної підсистеми до місця НС;

– вирішення питання початку АРІНЗ.

2. Виконання аварійно-рятувальних та інших невідкладних заходів:

– переміщення в область, де сталася НС, засобів проведення АРІНЗ відповідно до вирішеного питання;

– безпосереднє виконання робіт аварійно-рятувального характеру і інших невідкладних робіт;

– виведення спецзасобів із зони НС, по завершенні АРІНЗ і переміщення їх до вихідної точки.

3. Ліквідація збитків, спричинених НС:

– роботи по першочерговому життєзабезпеченні постраждалого персоналу;

– роботи з відновлення діяльності об'єктів постраждалих при НС (здійснюються силами об'єктів, постраждалих внаслідок НС).

АРІНЗ вважаються завершеними, коли закінчується розшук потерпілих, медична, психологічна та інша допомога їм, упередження загрози виникнення нових вогнищ уражень. Рятування персоналу при виникненні НС на підприємстві являється одним із найбільш важливим при проведенні АРІНЗ і включає в себе сукупність заходів по виведенню працівників із області, де виявлені шкідливі фактори впливу НС та їх похідні або захищення працівників від дії таких факторів, у т. ч. застосовуючи засоби індивідуального захисту та укриття.

Способами, що використовуються в основному для порятунку працівників, матеріалів і обладнання є:

– переміщення їх у безпечне місце, у тому числі з використанням спеціальних технічних засобів;

– захист від впливу небезпечних факторів надзвичайної ситуації.

Для порятунку працівників потрібно обрати найбезпечніші напрямки і методи. Вивезення потерпілих у безпечну локацію проводиться із розрахунку умов, в яких відбувається ліквідація НС і важкості їх ураження [24].

Засобами, що використовуються в основному для порятунку працівників, матеріалів і обладнання є:

– аварійно-рятувальне обладнання і механізми: гідравлічне аварійнорятувальне обладнання, ремені, обладнані карабінами; різальний інструмент у газовому полум'ї оснащений різаком, напірним рукавом, редуктором і газовим балоном (бензорізи, газозварювальні апарати тощо), ломи, кувалди, лопати, кіркимотики важкі, сокири, пилки, підйомні засоби (включно з лебідками, домкратами тощо), мотузки, окуляри захисної дії, освітлювальні пристрої, бензо- і електропили та ін.

– рятувальне обладнання (рятувальні рукави, канати, плетені драбини та індивідуальні засоби порятунку), засоби захисту, дрони та квадрокоптери, плавзасоби;

– стаціонарні та ручні драбини, що використовуються в якості пожежного інвентаря, тощо; автопідйомники та драбини на базі автомобілів та інші доступні рятувальні засоби.

4.3. Вплив електромагнітного імпульсу (EMI) ядерного вибуху на елементи виробництва та заходи захисту

EMI здатний викликати потужні імпульси струмів і напруг у проводах і кабелях повітряних і підземних ліній зв'язку, сигналізацій, управління, електропередачі, в антенах радіостанцій.

Вплив EMI може призвести до згорання чутливих електронних і електричних елементів, пов'язаних з великими антенами чи відкритими проводами, а також до серйозних порушень в цифрових і контрольних пристроях, зазвичай без необоротних змін.

Для найбільш важливих пристроїв треба застосовувати заходи захисту та підвищувати їх стійкість до EMI. Ступінь ушкодження залежить в основному від амплітуди наведеного імпульсу напруги чи струму і електричної міцності обладнання. Особливо схильна до впливу EMI

радіоелектронна апаратура, виконана на напівпровідникових та інтегральних системах, працюючих на малих струмах і напругах і, чутливих до впливу зовнішніх електричних і магнітних полів.

ЕМІ пробиває ізоляцію, випалює елементи електросхем радіоапаратури, викликає коротке замикання в радіопристроях, іонізацію діелектриків, спотворює або повністю стирає магнітний запис, позбавляє пам'яті ЕОМ.

Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу уражаючих факторів під час надзвичайних ситуацій.

При проведенні цих заходів необхідно враховувати конкретні умови підприємства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

Захистити цінне і унікальне устаткування можна завдяки проведенню інженерно-технічних заходів, щоб зменшити небезпеку пошкодження і руйнування цінного й унікального устаткування, комп’ютерної техніки, станків з програмним керуванням, шліфувальних, токарних, розточних, зубофрезерних, пресових станків, автоматичних конвеєрних ліній та іншого устаткування. Варіантами такого захисту є розміщення зазначеного устаткування в заглиблених приміщеннях, а також використання спеціальних захисних пристосувань, закріплення станків на фундаментах, застосування контрфорсів для підвищення стійкості проти перекидання обладнання

Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні. З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги.

Повітряні лінії електропостачання замінити на підземно-кабельні. Створення необхідних запасів (резервів) паливно-мастильних матеріалів та інших видів палива й організація їх безпечного зберігання. Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ.

В загальному, методи і засоби захисту від електромагнітних полів можна умовно розділити на інженерно-технічні, організаційні та лікувально-профілактичні. Згідно зі встановленою процедурою, захист людини від такого небезпечного впливу повинен здійснюється такими способами:

– зменшення випромінювання від джерела;
– екранування джерела випромінювання та робочого місця;
– встановлення санітарно-захисної зони;

Залежно від джерел випромінювання та того, де саме вони знаходяться – ззовні чи в середині приміщення, можна оптимально підібрати захист від ЕМП. Якщо джерело забруднення електромагнітним випромінюванням знаходиться ззовні приміщення (трансформатора підстанція чи електрощитова, вишка стільникового зв’язку, тощо), надійним засобом захисту від електромагнітного випромінювання у цьому випадку є спеціально розроблена для захисту від ЕМП екрануюча сітка HEG10 фірми «Gigahertz Solutions». Цю сітку кріплять до стіни за допомогою спеціального клею, що також має екрануючі властивості. Сітка HEG10 забезпечує захист від ЕМП високих частот та від малих електричних полів, сітка потребує заземлення.

Якщо ж джерело ЕМП знаходиться в середині приміщення, оптимальним засобом захисту від електромагнітного випромінювання є екрануюча фарба. Надійний захист від електромагнітних полів забезпечує фарба для екранування ЕМП виробництва «Gigahertz Solutions». Вона не токсична, відтак, придатна до використання як в середині приміщення, так і ззовні.

Окрім функції захисту від електромагнітного випромінювання, ці засоби можна також використовувати і для захисту даних з комп’ютерних мереж та комп’ютерів. Окрім того, захист офісу від ЕМП ззовні дасть можливість заблокувати зовнішні мережі WIFI, та захистити від проникнення ззовні власну мережу WIFI, що актуально для офісів, розміщених у великих офісних центрах.

Висновки.

У результаті проведеного аналізу щодо застосування основних способів та засобів в ході проведення невідкладних аварійно-рятувальних робіт на промисловому підприємстві встановлено послідовність дій, які необхідно виконати для збереження життя і здоров’я працівників підприємства, а також необхідні для цього матеріально-технічні засоби. Також проведено аналіз дії електромагнітного імпульсу при ядерному вибухові на людину та господарську діяльність і визначено потенційні шляхи мінімізації його негативного впливу та збереження життєдіяльності підприємств.

ВИСНОВКИ

Основні наукові та практичні результати полягають в наступному.

1. Проведено аналіз базових понять та властивостей великих даних, встановлено особливості їх формування та методи опрацювання, що дало змогу обґрунтувати доцільність визначення критеріїв продуктивності програмно-апаратної інфраструктури для моделей з генеративними алгоритмами.

2. Проаналізовано життєвий цикл великих даних та визначено особливості етапів і процесів, характерних при опрацюванні великих даних, а також функції Big Data інженерів, що дало змогу чітко визначити процеси конвеєра роботи з такими даними, зокрема щодо моделювання даних, якості, безпеки, управління, архітектури та оркестрації.

3. Визначено базові концепції великих мовних моделей з реалізацією генеративних алгоритмів і встановлено, що вони належать до класу інтелектуальних моделей, які дають змогу досліджувати статистичні шаблони, граматику та семантику людської мови та використовують потужні набори даних і відповідно значні апаратно-програмні ресурси.

4. Проведено аналіз процесів генерування відповідей великими мовними моделями і встановлено, що основними з них є попереднє паралельне заповнення токенами і декодування. Це дало змогу визначити найбільш вузькі місця та критерії управління ефективністю використання програмно-апаратних ресурсів.

5. Запропоновано критерії продуктивності у процесі попереднього опрацювання запитів до моделей з генеративними алгоритмами: час формування першого токену, час генерації вихідного токену для кожного користувача, прихована затримка формування відповіді користувачам та пропускної здатності моделі, що дало змогу керувати ресурсами і налаштуваннями інфраструктури апаратно-програмної інфраструктури і оптимально їх розподіляти.

6. Запропоновано метод та показники оптимізації використання ресурсів пам’яті та графічних процесорних ядер при формуванні висновків (відповідей) великими мовними моделями, що дало змогу здійснювати оптимальне налаштування інфраструктури і забезпечити баланс продуктивності та витрат на навчання та донавчання моделей.

7. Проведено порівняльний аналіз різних великих мовних моделей та використовуваної ними інфраструктури і запропоновано використання кешування ключ-значення, що дає можливість оптимізувати використання функції уваги при навчанні та функціонуванні моделей з генеративними алгоритмами.

8. Проаналізовано архітектуру класу великих мовних моделей на основі трансформерів і встановлено, що основними їх компонентами є нейронні мережі з енкодерами та декодерами, що дало змогу обґрунтувати доцільність застосування запропонованих показників ефективності та методу оптимального управління інфраструктурою при застосуванні підходу передачі знань з метою донавчання моделей.

9. Досліджено принципи організації, налаштування і застосування попередньо навчених моделей машинного навчання, що дало можливість обирати оптимальних провайдерів хмарних сервісів для подальшої адаптації моделей для користувацьких предметних областей.

10. Програмно імплементовано механізми прямого доступу до моделей з генеративними алгоритмами машинного навчання та експериментально доведено їх ефективність. Визначено необхідність та доцільність проведення додаткових досліджень щодо вибору оптимальних платформ для підтримки підходу transfer learning з врахуванням запропонованих критеріїв та методу оптимізації використання ресурсів програмно-апаратної інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a large language model (LLM)? URL: https://www.elastic.co/what-is/large-language-models (дата звернення: 10.09.2023).

2. API Reference. URL: https://platform.openai.com/docs/api-reference (дата звернення 10.09.2023).

3. Welcome to the OpenAI developer platform. URL: https://platform.openai.com/docs/overview (дата звернення: 10.09.2023).

4. Smolan R., Erwitt J. The Human Face of Big Data. Against All Odds Productions. 2012. P. 223.

5. Kleppmann M. Designing Data-intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media. 2017. P.590.

6. Popovych N., Lutskiv A., Mitsa O., Lyntvar O., Ivanova A. Ukrainian Redaction of Church Slavonic (URCS): Needs for Digitalization and Text Corpora Platform Generation. Part I. CEUR Workshop Proceedings.2023. pp. 266–278.

7. Lutskiv A., Lutsyshyn R. Corpus-based translation automation of adaptable corpus translation module. CEUR Workshop Proceedings. 2021. pp. 511–527.

8. Lutskiv A., Popovych N. Big data-based approach to automated linguistic analysis effectiveness. Proceedings of the 2020 IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP 2020. 2020. pp. 438–443.

9. Lutskiv A., Popovych N. Big data approach to developing adaptable corpus tools. CEUR Workshop Proceedings. 2020. pp. 374–395.

10. Lutskiv A., Popovych N. Adaptable text corpus development for specific linguistic research. 2019 IEEE International Scientific-Practical

Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 – Proceedings. 2019. pp. 217–223.

11. Технологія BERT від Google: що це, і як із цим жити? URL: https://sprava.ua/blog/tehnologija-bert (дата звернення: 28.09.2023).

12. Devlin J., Ming-Wei Chang, Lee K., Toutanova K.. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 02.10.2023).

13. Reference for built-in BERT algorithm. URL: https://cloud.google.com/ai-platform/training/docs/algorithms/reference/bert (дата звернення 30.09.2023).

14. Qiurui Chen. T5: a detailed explanation. URL: https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51 (дата звернення: 04.10.2023).

15. Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer. URL: https://blog.research.google/2020/02/exploring-transfer-learning-with-t5.html (дата звернення: 10.10.2023).

16. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп’ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

17. Луцків А.М., Островський А.Я. Характеристики та сфера застосування великих мовних моделей. Матеріали XII міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (6-7 грудня 2023 року). Тернопіль: ТНТУ. 2022. С. 452.

18. Луцків А.М., Островський А.Я. Організація доступу до моделі GPT-3 засобами мови Python. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2022. С. 168.

19. Shabliy N., Lupenko S., Lutsyk N., Yasniy O., Malyshevska O. Keystroke dynamics analysis using machine learning methods. Applied Computer Science. 2021. Vol. 17, No. 4. P. 75-83.

20. Лупенко С.А., Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи магістра для студентів спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти усіх форм навчання. Тернопіль, ТНТУ. 2021. 34 с.

21. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. CEUR Workshop Proceedings, 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), Ternopil, Ukraine, November 22–24, 2022. Vol. 3309. P. 194-204.

22. T5 Finetuning Tips. URL: https://discuss.huggingface.co/t/t5-finetuning-tips/684 (дата звернення: 06.10.2023 р).

23. Жидецький В.Ц. Охорона праці користувачів комп´ютерів. Львів: Афіша, 2011. 176 с.

24. Желібо Е.Н. Безпека життєдіяльності: Навчальний посібник/ За редакцією Е.П. Желібо, В.М. Пічі. – Київ: «Караве-ла», Львів: «Новий світ - 2000», 2011. 320с.

25. Стадник І.Я., Зварич Н.М. Оцінка хімічної обстановки при аваріях на хімічно небезпечних об'єктах викидом (виливом) небезпечних хімічних речовин та застосуванні хімічної зброї. ТНТУ. 2020. 36 С.

Додаток А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XII Міжнародної науково-практичної конференції молодих учених та студентів
6-7 грудня 2023 року

УКРАЇНА
ТЕРНОПІЛЬ – 2023

Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів
«АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 6-7 грудня 2023 року

УДК 004.048

А. М. Луцків канд. техн. наук, доцент, А. Я. Островський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ХАРАКТЕРИСТИКИ ТА СФЕРА ЗАСТОСУВАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

A. M. Lutskiv PhD., Assoc. Prof., A. Ya. Ostrovskyi

CHARACTERISTICS AND SCOPE OF LARGE LANGUAGE MODELS

Великі мовні моделі (LLM) представляють собою підмножину моделей глибокого навчання, відомих як моделі «нейронної мови». Їх особливістю є те, що вони навчаються на потужному наборі текстових даних, а це в свою чергу дає змогу їм вивчати статистичні шаблони, граматику та семантику людської мови.

На відміну від попередніх підходів щодо опрацювання природної мови, LLM можуть забезпечувати генерацію зв'язного і змістовно релевантного тексту, що робить їх надзвичайно універсальними для розуміння природної мови та завдань створення.

До ключових характеристик LLM належить:

– масштаб: LLM великі, часто містять від сотень мільйонів до мільярдів параметрів, що дозволяє їм фіксувати складні мовні шаблони.

– попередня підготовка: вони попередньо навчені масивним текстовим корпусам, що дає їм широке розуміння мови.

– «тонка настройка»: LLMs можна точно налаштувати для конкретних завдань, що робить їх адаптованими до широкого спектру застосування.

– генерація: вони можуть створювати текст, схожий на той, який фоомує людина, включаючи статті, код, вірші тощо.

– відповіді на запитання: LLM чудово відповідають на запитання, надаючи стислі відповіді на основі контексту.

Великі мовні моделі відіграють велике значення у різних сферах діяльності. LLMs досягли надзвичайного успіху в задачах розуміння природної мови. Це стосується аналізу емоцій та настроїв людини, мовного перекладу і чат-ботів. Здатність до розуміння контексту і генерації зв'язного тексту на основі LLM відкрила нові можливості для автоматизації та покращення спілкування.

Окрім цього, мовні моделі відіграють вагому роль у створенні завдань, починаючи від творчого написання та підсумовування вмісту до створення коду та написання звітів. Це має застосування в творчих галузях, створенні контенту та автоматизації повторюваних завдань.

LLM забезпечують більш ефективний і точний пошук інформації. Вони забезпечують роботу пошукових систем, систем рекомендацій і персональних помічників, допомагаючи користувачам точно і швидко знаходити адекватний контент. LLM можуть перетворювати текст у звук, допомагати у мовному перекладі та покращувати загальний досвід користувача для тих, хто має особливі потреби.

Великі мовні моделі підтримують дослідження в різних дисциплінах, аналізуючи та узагальнюючи наукову літературу, допомагаючи дослідникам в аналізі даних і надаючи ідеї в таких галузях, як охорона здоров'я та матеріалознавство.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.048

Андрій Луцків канд. техн. наук, доцент, Андрій Островський
Тернопільський національний технічний університет імені Івана Пулюя

ОРГАНІЗАЦІЯ ДОСТУПУ ДО МОДЕЛІ GPT-3 ЗАСОБАМИ МОВИ PYTHON

Andriy Lutskiv PhD., Assoc. Prof., Andriy Ostrovskyi

ORGANIZING ACCESS TO THE GPT-3 MODEL USING PYTHON

GPT-3, Generative Pre-trained Transformer, став новаторською архітектурою мовної моделі, яка революціонізувала розуміння та генерацію природної мови. Архітектура GPT-3 побудована на основі моделі Transformer, що включає багато параметрів для досягнення виняткової продуктивності. GPT-3 містить стек шарів енкодерів трансофрмера. Кожен рівень складається з розподілених механізмів уваги та нейронної мережі прямого поширення. Механізм уваги дозволяє моделі фіксувати залежності та зв'язки між словами, у той час як мережі прямого поширення обробляють і перетворюють закодовані представлення. Ключова інновація GPT-3 полягає в його величезному розмірі з надзвичайно великою кількістю параметрів, які дозволяють фіксувати величезні знання мови. Архітектура GPT-3 показана на рис. 1.

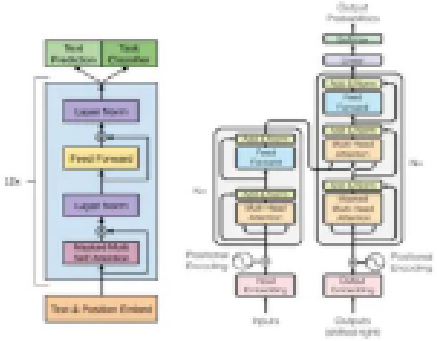

Рис. 1. Архітектура GPT-3

Для того, щоб забезпечити взаємодію через прикладний програмний інтерфейс з моделлю OpenAI GPT-3 та генерацію текстів можна використати програмний код, який представлено на рис. 2.

Рис. 2. Організація доступу до GPT-3для генерації тексту

Для організації доступу до моделі GPT-3 використано скрипт мови програмування Python та бібліотеку Transformers. Це дало можливість сформувати запит та одержати адекватну відповідь, яка показана справа від стрілки на рис. 2.