МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП’ЮТЕРНОЇ

ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КАФЕДРА КОМП’ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Аліна САВЧЕНКО

 «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 р.

**ДИПЛОМНА РОБОТА**

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

**Тема: «Інтеграція платформи Юніті із системою штучного інтелекту інтерактивного голосового помічника»**

**Виконавець:** Мельник Єлизавета Павлівна

**Керівник:**  к.т.н. доц. Зудов Олег Миколайович

**Нормоконтролер:**  к.т.н. доц. Боровик Володимир Миколайович

Київ 2021

|  |  |  |  |
| --- | --- | --- | --- |
| НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ | | | |
|  | | | |
| Факультет | | *Кібербезпеки, комп’ютерної та програмної інженерії* | |
| Кафедра | *Комп’ютерних інформаційних технологій* | | |
| Спеціальність | | | *122 «Комп’ютерні науки»* |

**ЗАТВЕРДЖУЮ**:

Завідувач кафедри КІТ

\_\_\_\_\_\_ \_\_\_\_\_\_\_ Аліна САВЧЕНКО

(підпис)

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

ЗАВДАННЯ

на виконання дипломної роботи

|  |
| --- |
| Мельник Єлизавети Павлівни |
| (ПІБ випускника ) |

1. Тема роботи «Інтеграція платформи Юніті із системою штучного інтелекту інтерактивного голосового помічника» затверджена наказом ректора №

2. Термін виконання роботи: з \_\_\_\_\_\_\_\_\_ по \_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи: інтеграція платформи Юніті з системою штучного інтелекту інтерактивного голосового помічника

4. Зміст пояснювальної записки: Unity3d і система штучного інтелекту інтерактивного помічника та принцип її роботи, огляд обраних методів інтеграції Unity3d і системи штучного інтелекту інтерактивного помічника, вимоги до виду архітектури, інтеграція Unity3d і система штучного інтелекту інтерактивного помічника

5. Перелік обов’язкового ілюстративного матеріалу: мета та задачі, огляд різних штучних інтелектів, аналіз наявних голосових помічників й принцип їх роботи, представлення астеів для реалізації інтеграції, інтеграція Юніті з системою штучного інтелекту, впроавдження синтезу голосу, результат роботи, висновки.

6. Календарний план-графік

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Завдання | Термін виконання | Підпис керівника |
| 1. | Ознайомлення з постановкою задачі та вивчення літератури | 01.09.2021–07.09.2021 |  |
| 2. | Написання 1 розділу, представлення керівнику | 08.09.2021–29.09.2021 |  |
| 3. | Написання 2 розділу, представлення керівнику | 30.09.2021–27.10.2021 |  |
| 4. | Написання 3 розділу, представлення керівнику | 29.10.2021–15.11.2021 |  |
| 5 | Написання 4 розділу, представлення керівнику | 16.11.2021–09.12.2021 |  |
| 6. | Загальне редагування та друк пояснювальної записки, графічного матеріалу | 09.12.2021–12.12.2021 |  |
| 7. | Проходження нормоконтролю, перепліт пояснювальної записки. | 13.12.2021–14.12.2021 |  |
| 8. | Розробка тексту доповіді. Оформлення графічного матеріалу для презентації | 15.12.2021–16.12.2021 |  |

|  |  |
| --- | --- |
| 7. Дата видачі завдання |  |

|  |  |  |
| --- | --- | --- |
| Керівник дипломної роботи |  | Зудов О.М. |
|  | (підпис керівника) | |
| Завдання прийняв до виконання |  | Мельник Є.П. |
|  | (підпис випускника) | |

**РЕФЕРАТ**

Пояснювальна записка до дипломної роботи «Інтеграція платформи Юніті із системою штучного інтелекту інтерактивного голосового помічника»: 99 с., 85 рис., 4 табл., 22 інформаційних джерел.

**Об’єкт дослідження** - Юніті із системою штучного інтелекту інтерактивного голосового помічника.

**Мета роботи** – є дослідження можливостей інтеграції платформи Юніті із системою штучного інтелекту інтерактивного голосового помічника.

**Методи дослідження -** оцінка емпіричних дослідженнь штучного інтелекту (ШІ), різних моделей, їх параметрів та впливу на точність застосування. Дослідження аналогічних прикладів й підводні камені в оцінці та внеску оцінки на всіх етапах циклу дослідження. Методи оцінки протиставляються методам поведінкових наук, і зроблено висновок, що ШІ повинен визначити й удосконалити власні методи, також є необхідним гарне налаштування ШІ. Інтеграція ШІ в Unity3d. Тестування й аналіз роботи програми.

Результати магістерської роботи рекомендується використовувати під час проведення наукових досліджень та в практичній діяльності фахівців із інформаційних технологій проектування, що працюють з новітніми технологіями, штучним інтелектом або діалоговими системами типу голосовий помічнк.

Для інтеграції Unity3d з штучним інтелектом (GPT-J) використана платформа Unity, плагіни TextToSpeech й SpeechToText, пакет з фонами AllSky, ассет для створення реалістичниих аватарів UMA 2 - Unity Multipurpose Avatar, пакет для створення інтерфейсу Dark Theme UI, а також програми для роботи з Python - Jupiter Notebook, Anaconda й для роботи з C# кодом – VisualStudio 2019. Проведено інтеграція Unity з системою штучного інтелекту GPT-J й інтеграція системи з сервісами гугл для відтворення (синтезу) мовлення й для розпізнавання мовлення

ШТУЧНИЙ ІНТЕЛЕКТ, ГОЛОСОВИЙ ПОМІЧНИК, СЕРВІСИ ГУГЛ, РОЗПІЗНАВАННЯ МОВИ, СИСТЕМА ДІАЛОГУ, РОЗУМНИЙ АСИСТЕНТ, МЕТАВЕРС.

**ЗМІСТ**

[**ВСТУП** 10](#_Toc90009468)

[**РОЗДІЛ 1**  **UNITY3D І СИСТЕМА ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА ТА ПРИНЦИП ЇЇ РОБОТИ** 12](#_Toc90009469)

[**1.1.** **Визначення Unity та Системи Штучного інтелекту (ШІ) інтерактивного голосового помічника** 12](#_Toc90009470)

[**1.2.** **Найрозвинутіші штучні інтелекти інтерактивного голосового помічника** 14](#_Toc90009471)

[**1.3.** **Архітектура та використання шучного інтелекту (GPT-J) інтерактивного голосового помічника у інтеграції з Unity** 17](#_Toc90009472)

[**1.4.** **Основні зміни в новій версії GPT-J** 20](#_Toc90009473)

[**ВИСНОВКИ ДО РОЗДІЛУ 1** 27](#_Toc90009474)

[**РОЗДІЛ 2**  **ОГЛЯД ОБРАНИХ МЕТОДІВ ІНТЕГРАЦІЇ UNITY3D І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА** 28](#_Toc90009475)

[**2.1 Аналіз логіки ШІ, специфіки використання й реалізації програм з використанням штучного інтелекту** 28](#_Toc90009476)

[**2.1.1 Як машинне навчання і глибоке навчання відіграють роль у ШІ** 28](#_Toc90009477)

[**2.1.2 Як нейронна мережа функціонує всередині** 31](#_Toc90009478)

[**2.1.3 Чому саме Python використовується для роботи з ШІ** 34](#_Toc90009479)

[**2.1.4 Мова програмування Python** 38](#_Toc90009480)

[**2.2 .NET архітектура** 39](#_Toc90009481)

[**2.2.1 Мова програмування С#** 40](#_Toc90009482)

[**2.3 База даних The Pile** 41](#_Toc90009483)

[**2.3.1** **Включенні бази даних** 42](#_Toc90009484)

[**2.3.2** **Оцінка** 47](#_Toc90009485)

[**ВИСНОВКИ ДО РОЗДІЛУ 2** 49](#_Toc90009486)

[**РОЗДІЛ 3**  **ВИМОГИ ДО ВИДУ АРХІТЕКТУРИ** 50](#_Toc90009487)

[**3.1 Системні вимоги для Unity 2021.2** 51](#_Toc90009488)

[**3.1.1 Мінімальні системні вимоги Unity:** 51](#_Toc90009489)

[**3.1.2 Комп’ютер для Unity:** 51](#_Toc90009490)

[**3.2 Системні вимоги для GPT-J** 54](#_Toc90009491)

[**ВИСНОВКИ ДО РОЗДІЛУ 3** 55](#_Toc90009492)

[**РОЗДІЛ 4**  **ІНТЕГРАЦІЯ UNITY3D І СИСТЕМА ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА** 56](#_Toc90009493)

[**4.1 Встановлення і налаштування GPT-J** 56](#_Toc90009494)

[**4.1.1 Встановлення GPT-J на комп’ютер** 56](#_Toc90009495)

[**4.1.2 Налаштування середовища** 59](#_Toc90009496)

[**4.1.3 Запуск Jupyter Notebook** 61](#_Toc90009497)

[**4.1.4 Встановлення залежностей** 61](#_Toc90009498)

[**4.1.5 Завантаження ваг моделі (The Model Weights)** 62](#_Toc90009499)

[**4.1.6 Перетворення ваг у ваги Pytorch** 62](#_Toc90009500)

[**4.1.7 Завантаження перетвореної моделі** 63](#_Toc90009501)

[**4.1.8 Використання GPT-J-6B для створення вихідних даних** 64](#_Toc90009502)

[**4.1.9 Аутро та майбутня робота** 64](#_Toc90009503)

[**4.2 Розробка додатку в Unity** 64](#_Toc90009504)

[**4.2.1 Встановлення Unity Hub та Unity** 64](#_Toc90009505)

[**4.2.2 Створення проекту й Інтерфейс Unity** 65](#_Toc90009506)

[**4.2.3 Вибір й інсталяція ассетів** 66](#_Toc90009507)

[**4.2.4 Створення сцени й аватару** 69](#_Toc90009508)

[**4.2.5 Налаштування сервісів гугл та їх асетів в Unity** 73](#_Toc90009509)

[**4.2.6 Інтеграція GPTJ з Unity** 81](#_Toc90009510)

[**4.2.7 Створення й налаштування середовища для Аватару** 86](#_Toc90009511)

[**ВИСНОВКИ** 95](#_Toc90009512)

[**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ** 97](#_Toc90009513)

**ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ**

БД – База данних

ОЗП - Оперативна пам'ять (вона ж ОЗУ або RAM)

ОС – Операційна система

ООП – Об’єктно-орієнтоване програмування

ПК - Персональний комп’ютер

ТБ - Тераба́йт

ЦП - Центральний процессор

ШІ - Штучний інтелект

AAA (triple-A) - Читається "трипл-ей"

AІ (Artificial Intelligence) - Штучний інтелект

API (Application Programming Interface) - Інтерфейс прикладного програмування

AR (Augmented Reality) - Доповнена реальність

BPB (Bits Per Byte) - Бітів в байті

BPE (Byte Pair Encoding) - Парне кодування байтів

CLI (Common Language Infrastructure) - Спільна мовна інфраструктура

CLR (Common Language Runtime) - Середовище виконання загальної мови

CTS (Common Type Specification) - Загальна специфікація типу

DCA (Dynamic Character Avatar) - Аватар динамічного персонажа

DCS (Dynamic Character System) - Система динамічних персонажів

DL (Deep Learning) - Глибоке навчання

DLL (Dynamic-Link Library) -Бібліотека динамічних посилань

EN (English) - Англійська мова

GB (Gigabyte) - Гігабайт

HTTP (Hypertext Transfer Protocol) - Протокол передачі гіпертексту

HTML (HyperText Markup Language) - Мова розмітки гіпертексту

IDE (Integrated Development Environment) - Інтегроване середовище розробки

IL (Intermediate Language) - Мова проміжного рівня

JIT (Just in Time) - Вчасно

JSON (JavaScript Object Notation) - Нотація об'єктів JavaScript

LINQ (Language-Integrated Query) - Мовно-інтегрований запит

ML (Machine Learning) - Машинне навчання

NIH (National Institutes of Health) - Національні інститути здоров'я

NLP (Neuro-Linguistic Programming) - Обробка природної мови

NVMe (Non-Volatile Memory Host Controller Interface Specification) - Специфікація інтерфейсу хост-контролера енергонезалежної пам’яті

PMC (PubMed Central) - частина онлайн-сховища PubMed для біомедичних статей

OWT2 (OpenWebText2) - це узагальнений набір даних веб-скрейпінгу, натхненний WebText та OpenWebTextCorpus

RoPE (Nylon rope HS Code) - Кодування з поворотом

SSD (Solid-State Drive) - Твердотільний накопичувач

TPU (Tensor Processing Unit) - Тензорний процесор

TPU-VM (Tensor Processing Units Virtual Machine) - Тензорний процесор на Віртуальній машині

TRC (TPU Research Cloud) - Дослідницька хмара Тенсорного процессору

UI (User Interface) - Користувацький інтерфейс

UMA (Unity Multipurpose Avatar) - Багатофункціональний аватар Unity

URL (Uniform Resource Locator) - Уніфікований покажчик інформаційного ресурсу

UTF-8 (Unicode Transformation Format – 8-bit) - Формат перетворення Unicode – 8 біт

VR (Virtual Reality) - Віртуальна реальність

XML (Extensible Markup Language) - Розширена мова розмітки

# **ВСТУП**

У першій половині 20-го століття наукова фантастика познайомила світ з концепцією штучно створених інтелектуальних роботів. Це почалося з "безсердечної" Залізної людини з "Чарівника країни Оз" і продовжилося людиноподібним роботом, який видавав себе за Марію в "Метрополісі".

З кожним роком мрія про робота з інтелектом, що може виконувати людську роботу, без необхідності постійного контролю ставала все більш схожою на реальність й швидко поширювалася серед мас, набувала все більшого впливу.

Вже у 1950-х роках було ціле покоління вчених, математиків і філософів, в свідомості яких концепція штучного інтелекту (або AІ) стала культурно засвоєною. Одним з таких людей був Алан Тьюринг, молодий британський ерудит, який досліджував математичні можливості штучного інтелекту. Тьюринг припустив, що люди використовують доступну інформацію, а також розум для вирішення проблем і прийняття рішень, так чому ж машини не можуть робити те ж саме? Це стало логічною основою його роботи 1950 року "Обчислювальна техніка та інтелект", в якій він обговорював, як створити інтелектуальні машини і як перевірити їх інтелект.

Тьюринг був одним з учених що дав поштовх для розвитку сучасного інтелекту, зараз же кожна топова компанія використовую штучний інтелект щоб покращити й оптимізувати свою роботу. Тільки так можна впоратися з такою великою кількістю даних й зробити максимально точні передбачення у різних сферах.

Зараз же в епоху "великих даних", в епоху, коли у нас є можливість збирати величезні обсяги інформації, занадто громіздкі для обробки людиною і застосування штучного інтелекту в цьому відношенні вже виявилося дуже плідним в декількох галузях, таких як технології, банківська справа, маркетинг і розваги. Як бачимо навіть якщо алгоритми не сильно поліпшуються, великі дані і масивні обчислення дозволяють штучному інтелекту навчатися методом грубої сили (тобто спроб і помилок). Можливо, є свідчення того, що закон Мура трохи сповільнюється, але зростання обсягу даних, безумовно, не втрачає темпу. Прориви в комп'ютерній науці, математики або нейронауці - всі вони служать потенційним виходом через стелю закону Мура.

У найближчому майбутньому AІ це найбільш визначний винахід людства, особливо в комбінації з квантовими комп’ютерами. Насправді, великі зміни вже відбуваються. Сьогодні вже майже не залишилось людей, що роблять обдзвони, все це зараз виконують звичайнісінькі боти, підключені до баз даних з клієнтськими даними. Взаємодію з експертною системою в плавній бесіді або розмову двома різними мовами, що перекладається в режимі реального часу, вже не треба уявляти, це вже реальність. На дорогах вже авто з автопілотами й планують зробити вертолітне таксі, а пару років тому це було лише фантазією.

У довгостроковій перспективі метою є загальний інтелект, тобто машина, що перевершує людські когнітивні здібності у всіх завданнях. Це схоже на розумного робота, якого ми звикли бачити в кіно. І це 100% буде досягнуто в найближчі 50 років, так само як і довголіття або безсмертя. Вже доволі багато розвинутих проектів, що можуть робити клітини молодшими за їх реальний вік і тим самим покращувати рівень життя. Навіть якщо така можливість існує, етичні питання будуть служити сильним бар'єром на шляху до її реалізації. Коли прийде час (а краще навіть раніше), потрібно буде серйозно поговорити про політику і етику машин (за іронією долі, обидві ці теми є фундаментально людськими), а поки дозволимо ШІ неухильно удосконалюватися і змінювати лад в суспільстві.

# **РОЗДІЛ 1 UNITY3D І СИСТЕМА ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА ТА ПРИНЦИП ЇЇ РОБОТИ**

* 1. **Визначення Unity та Системи Штучного інтелекту (ШІ) інтерактивного голосового помічника**

Unity - це ігровий рушій 3D / 2D й у той же час потужна кроссплатформенна середа розробки.

Як ігровий рушій, Unity забазпечує багато важливих вбудованих функцій, які роблять гру працездатною, а саме: фізика, 3D-рендеринг і виявлення зіткнень. З точки зору розробника це означає, що більшість питань, які йому необхідно вирішити вже вирішенні. Замість того щоб починати новий проект зі створення нового фізичного движка з нуля, прораховуючи кожен рух кожного матеріалу або те, як світло повинне відбиватися від різних поверхонь можна просто використати один з наявних движків, таких як Юніті, де більшість логіки вже створено за нас. Це значно прискорює процес розробки продукту.

Крім ігрового движка, Unity - це ще і IDE. IDE означає "інтегроване середовище розробки", що описує інтерфейс, який дає вам доступ до всіх інструментів, необхідним для розробки, в одному місці. Програмне забезпечення Unity має візуальний редактор, який дозволяє творцям просто перетягувати елементи в сцени, а потім маніпулювати їх властивостями.

Штучний інтелект (ШІ) - це широка галузь комп'ютерної науки, що займається створенням інтелектуальних машин, здатних виконувати завдання, для яких зазвичай потрібен людський інтелект.

Чотири типу штучного інтелекту

* реактивні машини (Reactive Machines)
* обмежена пам'ять (Limited Memory)
* теорія розуму (Theory of Mind)
* самосвідомість (Self-Awarness)

Приклади Штучного Інтелекту: Siri, Alexa та інші розумні помічники; самоврядні автомобілі; Робо-консультанти; розмовні боти; Фільтри спаму в електронній пошті; рекомендації Netflix.

Штучний інтелект зазвичай підрозділяється на дві широкі категорії:

**Вузький ШІ:** іноді званий "слабким ШІ", цей вид штучного інтелекту працює в обмеженому контексті і є імітацією людського інтелекту. Вузький ШІ часто зосереджений на виконанні одного завдання надзвичайно добре, і хоча ці машини можуть здаватися інтелектуальними, вони працюють в умовах набагато більших обмежень і заборон, ніж навіть самий базовий людський інтелект.

**Загальний штучний інтелект (AGI):** AGI, іноді званий "сильним ШІ", - це вид штучного інтелекту, який ми бачимо в кіно, наприклад, роботи з Westworld або Дейта з Star Trek: The Next Generation. AGI - це машина з загальним інтелектом, яка, подібно до людини, може застосовувати цей інтелект для вирішення будь-якої проблеми.

Значна частина вузького ШІ забезпечується завдяки проривам у машинному (machine learning) та глибокому навчанні (deep learning). Розуміння різниці між штучним інтелектом, машинним навчанням та глибоким навчанням може викликати збентеження. Венчурний капіталіст Френк Чен дає хороший огляд того, як їх розрізняти, відзначаючи:

"Штучний інтелект - це набір алгоритмів та інтелекту, які намагаються імітувати людський інтелект. Машинне навчання - одна з них, а глибоке навчання - одна з технік машинного навчання" [5].

AI-помічники або асистенти - це запрограмовані машини, які працюють в області машинного навчання. Персональний помічник ШІ працює так само, як і звичайний людський персональний помічник. Додатки AI assistant можуть здійснювати телефонні дзвінки, встановлювати нагадування, програвати музику, складати список справ і багато іншого [6].

Розробників ШІ для розробки програми в основному використовують мову програмування python. Велика кількість сервісів з розробки зробили взаємодію людини і машини набагато легшою і менш трудоємкою для людини, тобто оптимізувала процесс виробництва продукту або виконання завдання.

* 1. **Найрозвинутіші штучні інтелекти інтерактивного голосового помічника**

AI-помічників на телефонах, ноутбуках або пристроях вже набуди неабиякої популярності. Одними з найпопулярніших і кращих ШІ-асистентів, які працюють по всьому світу, є Siri, Alexa, Google Assistant, Cortana і багато інших.

**Siri**

Siri - це перший в історії смартфонів розумний помічник, розроблений компанією Apple (Рис. 1.1). iPhone 4S був телефоном з Siri, з тих пір все смартфони, розроблені Apple, мали цього розумного помічника. Слід зазначити одну особливість Siri - вона працює тільки з продуктами Apple і не може бути підключена до інших пристроїв. З точки зору популярності Siri довгий час була на вершині, поки люди не почали бачити більш просунуті технології в ІІ-помічниках Google і Amazon.

Незавісімо від конкуренції, Siri має високу адаптивність у відповідь на голосові команди, але відчуває труднощі з улавливанием складних фраз в порівнянні з Google.

Siri володіє високим рівнем інтелекту, оскільки може пропонувати відповіді на текст, нагадувати про заплановані зустрічі і нарадах, а також автоматично завершувати текстові повідомлення на півдорозі.

Siri не обмежується тільки телефонами; Apple впровадила Siri в усі свої пристрої, такі як Apple TV, Apple Smart Watch, а також в Mac Desktop.

**Google Assistant**

Google Assistant є одним з найбільш широко використовуваних користувачами андроїд-пристроїв по всьому світу (Рис. 1.1). Незважаючи на недвозначну назву "Google Assistant", компанія Google доклала всіх зусиль для його створення. Завдяки зусиллям високоосвічених розробників Google, програмне забезпечення є ретельно продуманим.

Google швидко додав в Асистент різні функції, наприклад, здійснення дзвінків за допомогою голосового набору.

Сільние боку Google Assistant полягають головним чином в його гнучкості при підключенні до пристроїв, не схожим на Siri. Assistant відкритий для Smart TV, Smart Watches і AI-пристроїв, що робить роботу з ним більш зручною.

У Індії, а точніше в більшості зарубіжних країн, спостерігається мовний бар'єр, тому Google Assistant має широкий набір мов, що полегшує спілкування і переклад.

**Amazon Alexa**

Amazon починав з сайту інтернет-магазинів, який виявився надзвичайно популярним, і amazon плавно ступив у світ штучного інтелекту і розумних помічників, реалізувавши "ALEXA". Alexa може зробити все, що ви попросите, починаючи від замовлення продуктів і закінчуючи інформацією про ціни на нові мобільні пристрої, що було розумним бізнес-ходом з боку Amazon.

Alexa в основному працює як розумна колонка, яка може бути підключена через ваші телефони, але якщо ви встановите додаток у вашому мобільному телефоні android або iOS, вона, звичайно, не розчарує вас.

Alexa може багато чого запропонувати: вона діє як справжній чоловік, який може змусити ваших дітей заснути, прочитавши їм казку на ніч, або зробити вашу кухню смачно пахне за допомогою покрокових рецептів страв.

Оскільки Amazon має широкий спектр покупок, Amazon Music, Amazon Prime та Amazon Firestick TV, розумна колонка Alexa (Рис. 1.1) найкраще працює з цими середовищами.

Найпотужнішими моделі глибокого навчання на сьогодні:



Рис. 1.1 AI-помічники Siri, Alexa, Google Assistant

**GPT-3**

GPT-3 - це модель глибокого навчання, яка була представлена кілька місяців тому і вразила світ своєю здатністю генерувати текст, переводити, виконувати арифметичні операції і, що найважливіше, виконувати завдання, для яких вона не була спеціально навчена.

Тепер спільнота Discord, EleutherAI, навчило кілька моделей GPT-3. І вони прийшли до двох дуже цікавим версіями, які ми розглянемо тут:

**GPT-NEO**

GPT-Neo - це кодова назва для сімейства мовних моделей на основі трансформаторів, в загальних рисах нагадують архітектуру GPT. Наша основна мета - підготувати модель, еквівалентну повнорозмірною GPT-3, і зробити її загальнодоступною під відкритою ліцензією.

GPT-Neo - це реалізація моделі і даних паралельних GPT-2 і GPT-3-подібних моделей, що використовують Mesh Tensorflow для розподіленої

підтримки. Ця кодова база розроблена для TPU. Вона також повинна працювати на графічних процесорах, хоча ми не рекомендуємо таку конфігурацію обладнання.

В даний час GPT-Neo має дві версії, одна з 1,3 мільярдами параметрів, інша з 2,7 мільярдами.

**GPT-J**

Паралельно була розроблена версія GPT, названа J-6B, яка працює як версія GPT-3 з 6 мільярдами параметрів. В результаті вийде модель з відкритим вихідним кодом, найбільш близька по продуктивності до GPT-3 (Табл. 1.1).

* Продуктивність нульового пострілу знаходиться приблизно на одному рівні з GPT-3 приблизно такого самого розміру, а розрив в продуктивності з GPT-3 приблизно такого самого розміру ближче, ніж у моделей GPT-Neo.
* Пропускна здатність GPT-J з об'ємом 6Б для навчання (151 тис. Токенов / с) вище, ніж GPT-Neo з об'ємом 2,7Б (148 тис. Токенов / с) на тому ж обладнанні (TPU v3-256 pod), що демонструє приблизно 125% -ве поліпшення ефективності.
* При конфігурації 6B на TPU V3-256 pod, GPT-J досягає високої абсолютної ефективності. Теоретичний максимум апаратного забезпечення становить 13,4 PFLOPs, а GPT-J досягає 5,4 PFLOPs, як показано в статті GPT3 (без урахування обчислень уваги, без урахування компромісів між обчисленнями і пам'яттю, таких як градиентная контрольна точка). При обліку цих додаткових факторів використовується 8,1 PFLOPs, або близько 60% від теоретичного максимуму.
* Навчання GPT-J зайняло близько п'яти тижнів на TPU v3-256.
  1. **Архітектура та використання шучного інтелекту (GPT-J) інтерактивного голосового помічника у інтеграції з Unity**

Більшість скриптів в цьому репозиторії призначені для запуску на TPU, які в рамках архітектури TPU-VM є віртуальні машини, здатні виконувати довільний код. Більшість сценаріїв розроблені для запуску TPU, підключення до нього по SSH для установки залежностей і копіювання коду з локальної директорії, а потім запуску робочого Ray, який може приймати виклики RPC.

TPUVMs виконує кроки навчання і оцінки моделі, збереження і завантаження контрольних точок, в той час як python-програма драйвера виконує завантаження даних і загальну оркестровку (наприклад, коли зберігати контрольні точки і т.д.).

TPUVMs виконує кроки навчання і оцінки моделі, збереження і завантаження контрольних точок, в той час як python-програма драйвера виконує завантаження даних і загальну оркестровку (наприклад, коли зберігати контрольні точки і т.д.).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Модель | Тренувальні FLOPs | ЛАМБАДА PPL | ЛАМБАДА Acc | Великомасштабний набір даних із 44 тис. проблем | Виклик набору даних для оцінки NLI здорового глузду | Контрольна перевірка та гарантія якості |
| GPT-2-1.5B | ----- | 10.63 | 51.21% | 59.4% | 50.9% | 70.8% |
| GPTNeo-2.7B | 6.8e21 | 5.63 | 62.2% | 56.5% | 55.8% | 73.0% |
| GPT-3-1.3B | 2.4e21 | 5.44 | 63.6% | 58.7% | 54.7% | 75.1% |
| GPT-3- Babbage | ----- | 5.58 | 62.4% | 59.0% | 54.5% | 75.5% |
| GPT-3-2.7B | 4.8e21 | 4.60 | 67.1% | 62.3% | 62.8% | 75.6% |
| GPT-J-6B | **1.5e22** | **3.99** | **69.7%** | **65.3%** | **66.1%** | **76.5%** |
| GPT-3-6.7B | 1.2e22 | 4.00 | 70.3% | 64.5% | 67.4% | 78.0% |
| GPT-3-Curie | ----- | 4.00 | 69.3% | 65.6% | 68.5% | 77.9% |
| GPT-3-175B | 3.1e23 | 3.00 | 76.2% | 70.2% | 78.9% | 81.0% |
| GPT-3-Davinci | ----- | 3.0 | 75% | 72% | 78% | 80% |

Це означає, що більшість сценаріїв (train.py, eval\_harness.py і т.д.) повинні виконуватися на віртуальній машині GCE в тому ж регіоні, що і TPU, щоб мінімізувати затримки RPC і вартість передачі даних. Інші сценарії (зазвичай ті, які не приймають аргумент --tpu, такі як device\_sample.py, device\_serve.py або device\_train.py) очікують запуску безпосередньо на TPUVM. Скрипти device\_ \* працюють тільки на v3-8, але не на більших стручках.

Таблиця 1.1 Потужність

Крім того, є приклад (resharding\_example.py) того, як перетворити надані контрольні точки (які мають 8 Шардена в разі GPT-J-6B) в менше число, наприклад, для роботи на GPU (s).

Точна настройка

Для точної настройки моделі запустіть файл device\_train.py на віртуальній машині TPU. Використовуючи TPU v3-8, ви можете виконувати тонке налаштування зі швидкістю ~ 5000 токенов в секунду, що має бути досить для невеликих і середніх наборів даних.

Залежність від JAX

Зверніть увагу, що ця бібліотека має деякі специфічні вимоги до версії JAX. Зокрема, для використання моделей v1 (включаючи GPT-J 6B) потрібно jax == 0.2.12. Це, в свою чергу, залежить від jaxlib == 0.1.68. Якщо цього не зробити, ви отримаєте загадкові помилки xmap.

Однак, щоб використовувати код моделі v2 (немає публічно випущених ваг), можна використовувати найновішу версію JAX [4].

Трансформатор сітки JAX

Бібліотека haiku, що використовує оператори xmap / pjit в JAX для модельного паралелізму трансформаторів.

Схема паралелізму аналогічна оригінальній Megatron-LM, яка ефективна на TPU завдяки високошвидкісній 2d mesh-мережі. Існує також експериментальна версія моделі, яка реалізує шардінг в стилі ZeRo.

Ця бібліотека розрахована на масштабованість приблизно до 40B параметрів на TPUv3s, після чого слід використовувати різні стратегії паралелізму. Див. Інші реалізації, такі як GPT-NeoX або DeepSpeed.

Одним з майбутніх напрямків досліджень є інтеграція цієї кодової бази з swarm-jax для досягнення подальшої масштабованості за допомогою конвеєрного паралелізму.

Попередньо навчені моделі

**GPT-J-6B**

Модель генерації тексту з авторегресії і 6 мільярдами параметрів, навчена на The Pile.

The Pile - це різноманітний набір даних з моделювання мов з відкритим вихідним кодом об'ємом 825 Гігабайт, що складається з 22 невеликих високоякісних наборів даних, об'єднаних разом. [8]

Для інтеграції з Unity налаштуємо середовища та тензорного потоку для AgentML у Windows 10 [9], а також використаємо Jupyter Notebook.

Jupyter Notebook - популярний інструмент у спільноті МЛ для навчання моделей (Рис. 1.2). Насправді це досить приголомшливий інструмент. Це щось на зразок REPL, змішаного з word документом. Є можливість написати код прямо в документі без запуску та IDE, а потім виконати його та побачити результати [10].

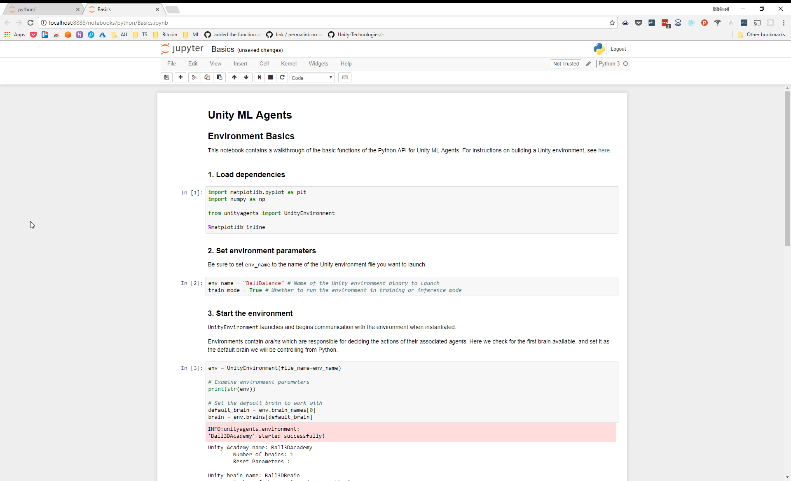


Рис. 1.2 Jupyter Notebook

Python в блокноті спілкується з "мозком" в середовищі. Це свого роду рідної міст між API машинного навчання в Python і "грою" / "середовищем". Коли навчальна сесія запускається з python, запускається середу exe, яка відкривається як окреме вікно на комп'ютері. Відкривається вікно середовища крихітне, імовірно для того, щоб GFX-карта не витрачала цикли на рендеринг і могла замість цього зосередитися на ML.

* 1. **Основні зміни в новій версії GPT-J**

Поява трансформерів в 2017 році повністю змінило світ нейронних мереж. З тих пір основна концепція Transformers була перероблена, переупакованої і перекомпонувати в декількох моделях. Результати перевершили рівень техніки в декількох бенчмарках машинного навчання. Фактично, в даний час у всіх кращих еталонних тестах в області обробки природної мови домінують моделі на базі Transformer. Деякі моделі сімейства Transformer - це BERT, ALBERT і серія моделей GPT.

У будь-якої моделі машинного навчання найбільш важливими компонентами процесу навчання є:

* Код моделі - компоненти моделі і її конфігурація
* дані, які будуть використовуватися для навчання
* доступна обчислювальна потужність

У сімействі моделей Transformer дослідники нарешті знайшли спосіб нескінченно збільшувати продуктивність моделі: Ви просто збільшуєте кількість навчальних даних і обчислювальну потужність.

Саме це і зробила компанія OpenAI, спочатку з GPT-2, а потім з GPT-3. Будучи добре фінансованої компанією (більше 1 мільярда доларів), вона могла дозволити собі навчати одні з найбільших моделей в світі. Для навчання моделі використовувався приватний корпус з 500 мільярдів токенів, а на обчислювальні витрати було витрачено близько 50 мільйонів доларів.

Хоча код більшості моделей мови GPT є відкритим, модель неможливо відтворити без величезної кількості даних і обчислювальних потужностей. А OpenAI вирішив закрити публічний доступ до своїх навченим моделям, надавши їх через API лише кільком обраним компаніям і приватним особам. Крім того, політика доступу не документована, довільна і непрозора.

Буття GPT-Neo

Коннор Ліхі, Стелла Бідерман, Лео Гао, Сід Блек, Філ Ванг та інші створили компанію EleutherAI з ідеєю створення технології ШІ з відкритим вихідним кодом для всього світу. Однією з перших проблем, яку вирішила вирішити команда, було створення GPT-подібної мовної моделі, яка була б доступна для всіх.

Як уже згадувалося, велика частина коду для такої моделі вже була доступна, тому основні проблеми полягали в пошуку даних і обчислювальних потужностей. Команда Eleuther поставила перед собою завдання створити набір даних з відкритим вихідним кодом, який можна порівняти за масштабом з тим, що OpenAI використовував для своїх мовних моделей GPT. Це призвело до створення The Pile. The Pile, випущений в січні 2021 року являє собою набір даних обсягом 825 ГБ, спеціально розроблений для навчання мовних моделей. Він містить дані з 22 різних джерел, включаючи академічні джерела (Arxiv, PubMed, FreeLaw і ін.), Інтернет-сторінки (StackExchange, Wikipedia і ін.), Діалоги з субтитрів, Github і ін. (Рис. 1.3) [7].

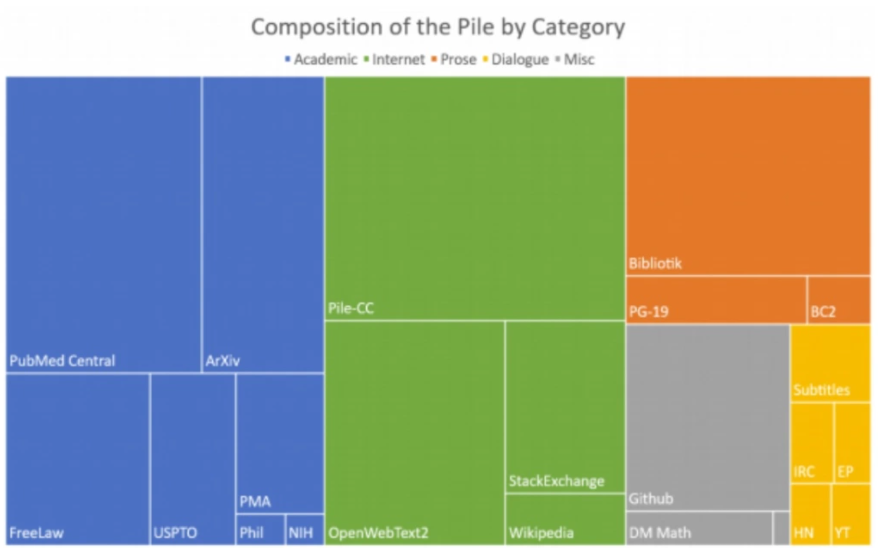


Рис. 1.3 Композиція Pile за категоріями

На Рис 1.3 Синій колір – це академічна література, зелений – література з інтернету, помаранчевий – проза, жовтий – діалоги й сіре – інше.

Для обчислень EleutherAI зміг використовувати незадіяні обчислення з TPU Research Cloud (TRC). TRC - це ініціатива Google Cloud, що підтримує дослідницькі проекти з розрахунком на те, що результати досліджень будуть передані світу через відкритий вихідний код, моделі і т. д.

22 березня 2021 року, після декількох місяців копітких досліджень і навчання, команда EleutherAI випустила дві навчені мовні моделі в стилі GPT, GPT-Neo 1.3B і GPT-Neo 2.7B. Код і навчені моделі знаходяться у відкритому доступі під ліцензією MIT. Моделі можна використовувати безкоштовно за допомогою платформи Transformers від HuggingFace.

Що нового в GPT-J

GPT-J-6B працює майже нарівні з 6,7B GPT-3 (або Curie) в різних завданнях з нульовим потоком.

GPT-J - кращий із загальнодоступних трансформаторів LM з точки зору нульовий продуктивності на різних завданнях низхідного потоку.

GPT-J забезпечує більш гнучкий і швидкий висновок, ніж аналоги на Tensorflow + TPU.

Таблиця 1.2 Деталі моделі

|  |  |
| --- | --- |
| Гіперпараметр | Значення |
| н\_параметри | 6,053,381,344 |
| н\_шари | 28\* |
| д\_моделі | 4,096 |
| д\_фф | 16,384 |
| н\_заголовки | 16 |
| д\_заголовок | 256 |
| н\_ctx | 2,048 |
| н\_словн | 50,257 (той же токенізатор, що і GPT-2/3) |
| Кодування\_позиції | Поворотні кодування положення (RoPE) |
| Розміри RoPE | 64 |

Даний проект зажадав значно менше людино-годин, ніж інші розробки великомасштабних моделей, що свідчить про те, що JAX + xmap + TPU - відповідний набір інструментів для швидкої розробки великомасштабних моделей.

Дизайн моделі:

Дизайн моделі і вибір гіперпараметров в точності повторюють дизайн моделі 6.7B GPT-3 з деякими відмінностями. Зокрема:

* Модель була навчена на 400B лексем з набору даних The Pile з 800GB тексту.
* Ефективне увагу (лінійне, локальне / ковзне вікно і т.д.) не використовувалося для простоти, так як воно не покращило б значно пропускну здатність при такому масштабі.
* Розмірність кожної головки уваги встановлена ​​на 256, що в два рази більше, ніж у GPT-3 приблизно такого самого розміру. Це помітно поліпшило пропускну здатність при мінімальному зниженні продуктивності.

Ми внесли два незначних архітектурних поліпшення:

* Поворотне вбудовування для кращої продуктивності.
* Паралельне розміщення шару уваги і фідфорвардного шару для зменшення кількості комунікацій.

\* кожен шар складається з одного блоку подачі вперед і одного блоку самоуваги (Таблиця 1.2)

Модель складається з 28 шарів з модельним розміром 4096 і розміром прямого переходу вперед 16384 (Таблиця 1.3). Розмір моделі поділено на 16 голів, кожна з яких має розмір 256. Кодування з поворотом (RoPE) було застосовано до 64 розмірів кожної голівки. Модель навчається зі словником токенізації 50257, використовуючи той самий набір BPE, що і GPT-2/GPT-3.

Оцінки нульового пострілу (Zero-Shot Evaluations)

Моделі, грубо відсортовані за продуктивністю, або за FLOP, якщо вони відсутні.

\* Представляє цифри оцінки, повідомлені відповідними авторами, всі інші цифри отримані при запуску lm-evaluation-harness або з випущеними вагами, або з доступом до API. Через тонких відмінностей в реалізації, а також різної постановки задачі нульового пострілу, ці показники можуть бути непорівнянні (Таблиця 1.3).

* Модель Megatron-11B не дає порівнянних метрик, а кілька реалізацій, що використовують випущені гирі, які не відтворюють якість генерації та оцінки. (Табл. 1.3) Таким чином, оцінка не проводилась.
* Ці моделі були навчені на даних, які містять можливе забруднення тестового набору. Моделі OpenAI GPT-3 не вдалося дедупліціровать навчальні дані для деяких тестових наборів, а моделі GPT-Neo, як і ця, навчені на The Pile, яка не була дедупліцірована ні на одному тестовому наборі.

Таблиця 1.3 Оцінки нульового пострілу

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Модель | Ваги | Тренувальні FLOPs | ЛАМБАДА PPL | ЛАМБАДА Acc | Великомасштабний набір даних із 44 тис. проблем | Виклик набору даних для оцінки NLI здорового глузду | Контрольна перевірка та гарантія якості | Розмір набору даних (GB) |
| Chance | ✓ | 0 | lot | -0% | 50% | 25% | 25% | 0 |
| GPT-3- Adal | ✓ | ----- | 9.95 | 51.6% | 52.9% | 43.4% | 70.5% | ----- |
| GPT-2- 1.58 | ✓ | ----- | 10.63 | 51.21% | 59.4% | 50.9% | 70.8% | 40 |
| GPTNeo- 1.38# | ✓ | 3.0e21 | 7.50 | 57.2% | 55.0% | 48.9% | 71.1% | 825 |
| Megatron- 2.58\* | x | 2.4e21 | ----- | 61.7% | ----- | ----- | ----- | 174 |
| GPTNeo- 2.78# | ✓ | 6.8e21 | 5.63 | 62.2% | 56.5% | 55.8% | 73.0% | 825 |
| GPT-3- 1.38\*\* | ✓ | 2.4e21 | 5.44 | 63.6% | 58.7% | 54.7% | 75.1% | -800 |
| GPT-3- Babbage | x | ----- | 5.58 | 62.4% | 59.0% | 54.5% | 75.5% | ----- |
| Megatron- 8.38\* | x | 7.8e21 | ----- | 66.5% | ----- | ----- | ----- | 174 |
| GPT-3- 2.78\*\* | x | 4.8e21 | 4.60 | 67.1% | 62.3% | 62.8% | 75.6% | ~800 |
| Megatron- 118+ | ✓ | 1.0e22 | ----- | ----- | ----- | ----- | ----- | 161 |
| GPT-J-6B# | ✓ | 1.5e22 | 3.99 | 69.7% | 65.3% | 66.1% | 76.5% | 825 |
| GPT-3- 6.78\*\* | ✓ | 1.2e22 | 4.00 | 70.3% | 64.5% | 67.4% | 78.0% | -800 |
| GPT-3- Curie# | x | ----- | 4.00 | 69.3% | 65.6% | 68.5% | 77.9% |  |
| GPT-3- 138\*\* | x | 2.3e22 | 3.56 | 72.5% | 67.9% | 70.9% | 78.5% | -800 |
| GPT-3- 1758\*\* | x | 3.1e23 | 3.00 | 76.2% | 70.2% | 78.9% | 81.0% | -800 |
| GPT-3- Davinci# | x | ----- | 3.0 | 75% | 72% | 78% | 80% | ----- |

## **ВИСНОВКИ ДО РОЗДІЛУ 1**

Під час розробки дипломної роботи було детально розглянуто принципи роботи моделей глибокого навчання, голосових помічників й штучного інтелекту. Було проведено порівняння характеристик найбільш популярних моделей й голосових помічників. Були набуті знання що до архітектури та використання штучного інтелекту (GPT-J) інтерактивного голосового помічника у інтеграції з Unity, знайдено можливості інтеграцій та найбільш ефективні технології для їх реалізації.

Переглянувши приклади коду, доступні в Інтернеті, які показують, як використовувати GPT-J в реальних ситуаціях. Дійшли висновку, що обсяг контролю обмежений, наприклад, можна вирішити, яким має бути параметр під назвою "температура" і якої довжини буде відповідь. У багатьох випадках відповіді звужуються до одного пропозиції / рядки, щоб модель НЛП не йшла від теми. З іншого боку, однорядкові пропозиції можуть бути цілком прийнятні, коли хочеться створити більш прості рішення "питання-відповідь", не надто дбаючи про контекст. Це робить GPT-J набагато розумнішою версією веб-пошуку, який, до речі, також управляється NLP-двигунами, так що нічого дивного в цьому немає. Знову ж таки, цього може бути достатньо або недостатньо у вашій конкретній ситуації.

Основна форма або спосіб управління GPT-J - це введення правильного тексту в якості вхідних даних. І для того, щоб "відчути", як поведе себе рушій, буде потрібно кілька проб і помилок, щоб написати текст таким чином, щоб отримати бажаний результат. Зважаючи на це можна сказати, що майбутнє NLP-двигунів легко передбачити, і гонка озброєнь буде тривати і тривати. Якщо виконавчі комп'ютери будуть визнані економічно або екологічно недоцільними, квантові комп'ютери можуть врятувати становище.

GPT-J - прекрасний приклад адаптації та удосконалення дуже просунутої моделі (GPT-3) в менший, але менш "злий" набір даних для обміну цим зі світом відкритого коду. Тож всі ці амбітні першопрохідці зможуть просунути кордону ще далі, беручи до уваги наші екологічні проблеми, аже це також один з важливих параметрів в епоху прогресу.

# **РОЗДІЛ 2 ОГЛЯД ОБРАНИХ МЕТОДІВ ІНТЕГРАЦІЇ UNITY3D І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА**

## **2.1 Аналіз логіки ШІ, специфіки використання й реалізації програм з використанням штучного інтелекту**

Загалом, мета використання AI — змусити комп’ютери думати так, як люди. Це може здатися чимось новим, але ця галузь зародилася в 1950-х роках.

Уявіть, що потрібно написати програму на Python, яка використовує AI для вирішення проблеми судоку. Спосіб досягнення цього полягає в написанні умовних операторів і перевірці обмежень, щоб побачити, чи можна розмістити число в кожній позиції. Ну, цей скрипт Python вже є додатком AI, тому що комп’ютер запрограмувано для вирішення проблеми.

Машинне навчання (ML) і глибоке навчання (DL) також є підходами до вирішення проблем. Різниця між цими методами та сценарієм Python полягає в тому, що ML і DL використовують навчальні дані замість жорстко закодованих правил, але всі вони можуть бути використані для вирішення проблем за допомогою ШІ. У наступних розділах дізнаємося більше про те, чим відрізняються ці дві методики.

### **2.1.1 Як машинне навчання і глибоке навчання відіграють роль у ШІ**

**2.1.1.1 Машинне навчання**

Машинне навчання – це техніка, за допомогою якої ви навчаєте систему вирішувати проблему замість явного програмування правил. Повертаючись до прикладу судоку в попередньому розділі, щоб вирішити проблему за допомогою машинного навчання, потрібно зібрати дані з розв’язаних ігор судоку та навчити статистичну модель. Статистичні моделі — це математично формалізовані способи наближення поведінки явища.

Дані для навчання

МН алгоритми

Модель

Нові дані для прогнозування

Результати прогнозування

Рис. 2.1 Робочий процес для навчання моделі машинного навчання

Поширеним завданням машинного навчання є навчання з наглядом, у якому у вас є набір даних із вхідними та відомими виходами. Завдання полягає в тому, щоб використовувати цей набір даних для навчання моделі, яка прогнозує правильні вихідні дані на основі вхідних даних. На зображенні нижче представлено робочий процес навчання моделі за допомогою навчання з наглядом:

Поєднання навчальних даних з алгоритмом машинного навчання створює модель. Потім за допомогою цієї моделі можна робити прогнози для нових даних.

*Примітка: scikit-learn — це популярна бібліотека машинного навчання Python, яка надає багато алгоритмів навчання під наглядом і без нагляду.*

Метою навчальних завдань під керівництвом є прогнозування нових, “невидимих даних”. Для цього припускається, що ці невидимі дані відповідають розподілу ймовірностей, подібному до розподілу навчального набору даних. Якщо в майбутньому цей розподіл зміниться, вам потрібно знову навчити свою модель, використовуючи новий набір навчальних даних.

**2.1.1.2 Функціональна інженерія**

Проблеми з прогнозуванням стають складнішими, якщо ви використовуєте різні типи даних як вхідні дані. Проблема судоку відносно проста, оскільки маємо справу безпосередньо з числами. Що робити, якщо потрібно навчити модель передбачати настрої в реченні, або що, якщо у є зображення, і потрібно дізнатися, чи зображена на ньому кішка? Про це зараз і дізнаємося.

Інша назва вхідних даних — функція, а розробка функцій — це процес вилучення ознак із необроблених даних. Маючи справу з різними видами даних, потрібно знайти способи представлення цих даних, щоб витягти з них значущу інформацію.

Прикладом одного з функціональних технік інженерії є лемматизація, під час якої видаляється флексія зі слів у реченні. Наприклад, флективні форми дієслова «watch», як-от «watch», «watching» і «watched», будуть зведені до їхньої леми або базової форми: «watch».

Якщо використовується масиви для зберігання кожного слова корпусу, то, застосовуючи лемматизацію, отримаємо менш розріджену матрицю. Це може підвищити продуктивність деяких алгоритмів машинного навчання. На наступному зображенні представлено процес лемматизації та представлення за допомогою моделі мішка слів:

Дейв - 1

знімав - 1

як - 1

горить - 1

ліс - 1

на - 2

пагорбі – 1

телефон - 1

[1,1,1,1,1,2,1,1]

Рис. 2.2 Створення функцій за допомогою моделі «мішок слів».

Дейв знімав, як горить ліс на пагорбі на телефон

По-перше, флективна форма кожного слова зводиться до його леми. Потім обчислюється кількість зустрічей цього слова. Результатом є масив, що містить кількість зустрічей кожного слова в тексті.

**2.1.1.3 Глибоке навчання**

Глибоке навчання – це техніка, за допомогою якої дозволяється нейронній мережі самостійно з’ясовувати, які функції є важливими, замість того, щоб застосовувати методи розробки ознак. Це означає, що завдяки глибокому навчанню можна обійти процес розробки функцій.

Не мати справу з розробкою функцій — це добре, оскільки процес стає складнішим, оскільки набори даних стають складнішими. Наприклад, як би витяглися дані, щоб передбачити настрій людини, на якій зображено її обличчя? З нейронними мережами вам не потрібно турбуватися про це, оскільки мережі можуть самостійно вивчати функції. У наступних розділах більш глибоко зануримося в нейронні мережі, щоб краще зрозуміти, як вони працюють.

### **2.1.2 Як нейронна мережа функціонує всередині**

**2.1.2.1 Нейронні мережі: основні поняття**

Нейронна мережа – це система, яка вчиться робити прогнози, виконуючи такі кроки:

1. Взяття вхідних даних
2. Складання прогнозу
3. Порівняння прогнозу з бажаним результатом
4. Налаштування внутрішнього стану для правильного прогнозування наступного разу

Вектори, шари та лінійна регресія є одними з будівельних блоків нейронних мереж. Дані зберігаються у вигляді векторів, а за допомогою Python ви зберігаєте ці вектори в масивах. Кожен шар перетворює дані, які надходять з попереднього шару. Ви можете розглядати кожен шар як крок розробки функцій, оскільки кожен шар витягує деяке представлення даних, які були отримані раніше.

Одна цікава особливість шарів нейронної мережі полягає в тому, що ті самі обчислення можуть витягувати інформацію з будь-яких даних. Це означає, що не має значення, чи використовуєте ви графічні дані чи текстові дані. Процес отримання значущої інформації та навчання моделі глибокого навчання однаковий для обох сценаріїв.

На зображенні нижче ви можете побачити приклад архітектури мережі з двома шарами:

Кожен шар перетворює дані, які надійшли з попереднього шару, застосовуючи деякі математичні операції.

**2.1.2.2 Процес навчання нейронної мережі**



Вхідні дані

Шар №1

Шар №2

“Танці”

Рис. 2.3 Нейронна мережа з двома шарами

Навчання нейронної мережі схоже на процес проб і помилок. Уявіть, що ви вперше граєте в дартс. У своєму першому кидку ви намагаєтеся влучити в центральну точку дартсборду. Зазвичай перший постріл — це просто для того, щоб зрозуміти, як висота і швидкість вашої руки впливають на результат. Якщо бачите, що дротик вище центральної точки, то можна налаштовати руку, щоб кинути його трохи нижче, і так далі.

Нижче наведено кроки для спроби потрапити в центр дошки для дартсу:

**Крок 1**

**Киньте** дротик у дартс

**Крок 2**

**Подивіться,** куди попав дротик

**Крок 3**

**Зкорегуйте** положення руки так, щоб наступного разу попасти ближче до центру

Рис. 2.4 Кроки, щоб потрапити в центр дартс

**Повторіть** кроки з 1го по 3й, допоки не навчитеся гарно прицілюатися

Зверніть увагу, що продовжується оцінювання помилки, спостерігаючи, де впав дротик (крок 2). Кроки продовжують йти далі, поки нарешті не потрапимо дротиком в центр дартс дошки.

З нейронними мережами процес дуже схожий: все починається з деяких випадкових ваг і векторів зміщення, робиться прогноз, який порівнюється з потрібним результатом. Після цього коригуються вектори для більш точного прогнозування наступного разу. Процес триває до тих пір, поки різниця між прогнозом і правильними цілями не стане мінімальною.

Знання того, коли зупинити навчання і яку точність встановити, є важливим аспектом навчання нейронних мереж, головним чином через сценарії переобладнання та недостатності.

**2.1.2.3 Вектори та ваги**

Робота з нейронними мережами полягає у виконанні операцій з векторами. Спершу вектори представляються у вигляді багатовимірних масивів. Вектори корисні в глибокому навчанні в основному через одну конкретну операцію: точковий добуток. Добуток двох векторів показує, наскільки вони подібні за напрямком, і масштабується за величиною двох векторів.

Основними векторами всередині нейронної мережі є ваги та вектори зміщення. Простіше кажучи, якщо потрібно, щоб нейронна мережа перевіряла, чи вхід подібний до інших входів, які він уже бачив. Якщо новий вхід подібний до попередніх входів, то і вихідні дані також будуть подібними. Так ви отримаєте результат прогнозу.

**2.1.2.4 Модель лінійної регресії**

Регресія використовується, коли потрібно оцінити зв’язок між залежною змінною та двома чи більше незалежними змінними. Лінійна регресія – це метод, який застосовується, коли потрібно наблизити зв’язок між змінними як лінійний. Цей метод бере свій початок з дев’ятнадцятого століття і є найпопулярнішим методом регресії.

*Примітка. Лінійна залежність – це зв’язок, у якому існує прямий зв’язок між незалежною та залежною змінною.*

Моделюючи зв’язок між змінними як лінійний, можна виразити залежну змінну як зважену суму незалежних змінних. Отже, кожна незалежна змінна буде помножена на вектор, який називається вагою. Окрім ваг і незалежних змінних, також додається ще один вектор: зміщення. Він встановлює результат, коли всі інші незалежні змінні дорівнюють нулю.

Як реальний приклад того, як побудувати модель лінійної регресії, уявіть, що потрібно навчити модель передбачати ціну будинків на основі площі та віку будинку. Тому вирішено змоделювати цей зв’язок за допомогою лінійної регресії. Наступний блок коду показує, як ви можете написати модель лінійної регресії для зазначеної проблеми в псевдокоді:

price = (weights\_area \* area) + (weights\_age \* age) + bias

У наведеному вище прикладі є дві ваги: вага\_площа та вага\_вік. Навчальний процес складається з коригування ваг і зміщення, щоб модель могла передбачити правильне значення ціни. Щоб досягти цього, потрібно буде обчислити помилку передбачення та відповідно оновити ваги.

Це основи того, як працює механізм нейронної мережі. Тепер настав час побачити, як застосувати ці концепції за допомогою Python.

### **2.1.3 Чому саме Python використовується для роботи з ШІ**

Протягом останніх років Python цінується за його невпинне зростання.

Python, який підтримується для додатків, починаючи від розробки веб-сайтів і закінчуючи написанням сценаріїв і механізацією процедур, швидко перетворюється на найкраще рішення серед інженерів у сфері штучного інтелекту, машинного навчання та глибокого навчання.

Комп’ютерний інтелект або штучний інтелект створив всесвіт шансів для інженерів-додатків. Комп’ютерна інформація дозволяє Spotify пропонувати ремікси і мелодії клієнтам, а Netflix — зрозуміти, які шоу подобаються й що пропонувати для перегляду. Крім того, він широко використовується організаціями для допомоги клієнтам, щоб стимулювати само адміністрування та покращувати робочі процеси та ефективність працівників.

Щоб втілити в життя AI, потрібно використовувати мову програмування, яка є стійкою, адаптивною та має доступні інструменти. Python пропонує все це, тому сьогодні групи Python AI розширюються.

Від просування до влаштування та обслуговування Python допомагає інженерам отримувати прибуток і бути впевненим у продукті, який створюється.

Переваги, завдяки яким Python найкраще підходить для AI та AI, включають:

* легкість і послідовність
* Доступ до екстраординарних бібліотек і структур для AI та AI (ML)
* Адаптивність
* Свобода сцени
* розгалужена мережа, що додає загальній славі мови.

**2.1.3.1 Велика кількість підтримуваних бібліотек**

Надзвичайно великий вибір бібліотек є однією з основних причин, чому Python є найпопулярнішою мовою програмування, яка використовується для ШІ. Бібліотека — це модуль або сукупність модулів, що розповсюджуються різними джерелами, такими як PyPi, які включають попередньо складений біт коду, який дозволяє клієнтам отримати певну корисність або виконувати різні дії. Бібліотеки Python виконують функціонал базового рівня, тому дизайнерам не потрібно кодувати їх із самого початку.

ML вимагає безперервної підготовки інформації, а бібліотеки Python дають доступ обробляти та змінювати інформацію. Це найбільші бібліотеки, які можна використовувати для машинного навчання та штучного інтелекту:

1. Pandas для інформаційних структур підвищеного рівня та розслідування. Він дозволяє об’єднувати та відсівати інформацію так само, як соціальна справа з інших зовнішніх джерел, таких як Excel, наприклад.
2. Keras для глибокого навчання. Він дозволяє швидкий підрахунок і створення прототипів, оскільки він використовує графічний процесор, незважаючи на центральний процесор ПК.
3. TensorFlow для роботи з глибоким навчанням шляхом налаштування, підготовки та використання штучних нейронних систем з великими наборами даних.
4. Matplotlib для створення двовимірних графіків, гістограм, графіків та різних типів представлення.
5. NLTK для роботи з обчислювальною етимологією, універсальним мовним визнанням та обробкою.
6. Scikit-картинка для обробки зображень.
7. PyBrain для нейронних систем, самостійного та допоміжного навчання.
8. Caffe для глибокого відкриття, що дозволяє обмінюватися між центральним і графічним процесором і обробляти понад 60 мільйонів зображень на день за допомогою окремого графічного процесора NVIDIA K40.
9. Статистичні моделі для вимірюваних розрахунків та дослідження інформації.
10. У сховищі PyPI можна знайти та переглянути інші бібліотеки Python.

**2.1.3.2 Простий і передбачуваний**

Python пропонує короткий і зрозумілий код. Хоча складні обчислення та гнучкі робочі процеси залишаються позаду AI, Python без зусиль дозволяє інженерам створювати надійні фреймворки. Дизайнери знаходять працездатний темп у своїх зусиллях у вирішенні проблеми ML, а не зосереджуються на спеціалізованих тонкощах мови.

Крім того, Python залучає численних дизайнерів, оскільки навчитися цьому просто. Код Python є розумним для людей, що спрощує створення моделей для ШІ.

Численні інженери програмного забезпечення стверджують, що Python є більш інтуїтивним, ніж інші діалекти програмування. Інші відкривають кілька систем, бібліотек і доповнень, які покращують виконання різних функцій. Загальновідомо, що Python підходить для спільної продуктивності, якщо в ній працюють численні інженери. Оскільки Python є широко корисною мовою, він може виконувати багато складних завдань AI та дає змогу швидко створювати моделі, які дозволять тестувати елемент для цілей AI.

**2.1.3.3 Низька межа ділянки**

Робота в індустрії машинного навчання та штучного інтелекту передбачає управління великою кількістю інформації, яку потрібно обробляти найбільш вигідно та переконливо. Низька перешкода розділу дозволяє більшій кількості дослідників інформації швидко отримати Python і почати використовувати його для розвитку ШІ, не витрачаючи зайвих зусиль на вивчення мови.

Мова програмування Python поступається звичайній англійській мові, і це робить шлях до вивчення легшим. Його проста пунктуація дозволяє швидко працювати зі складними фреймворками, гарантуючи чіткі відносини між компонентами фреймворка.

**2.1.3.4 Широкий вибір бібліотек і систем**

Актуалізація обчислень AI та ML може бути сумнівною і вимагає багато часу. Дуже важливо мати добре організовані та добре випробувані умови, щоб дати можливість дизайнерам продумати найкращі механізми кодування.

Щоб скоротити час покращення, інженери програмного забезпечення звертаються до різних структур і бібліотек Python. Бібліотека продуктів — це попередньо складений код, який дизайнери використовують для розуміння звичайних програмних завдань. Python, з його багатим стеком інновацій, має широкий набір бібліотек для комп’ютеризованих міркувань і ШІ. Ось деякі з них:

1. Keras, TensorFlow і Scikit-learn для AI
2. NumPy для чудової логічної реєстрації та перевірки інформації
3. SciPy для передового малювання
4. Pandas для огляду широко корисної інформації
5. Seaborn для сприйняття інформації

Завдяки таким упорядкуванням можна в рази пришвидшити створення AI. Групі вдосконалення не потрібно більше витрачати час на створення кастомного коду, адже може просто використати поточну бібліотеку для виконання основних завдань.

### **2.1.4 Мова програмування Python**

Python — це інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високо рівневі вбудовані структури даних у поєднанні з динамічним типізацією та динамічним зв’язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання в якості мови сценаріїв або склеювання для з’єднання існуючих компонентів. Простий, легкий у засвоєнні синтаксис Python підкреслює читабельність і, отже, знижує витрати на обслуговування програми. Python підтримує модулі та пакунки, що сприяє модульності програм і повторному використанню коду. Інтерпретатор Python і велика стандартна бібліотека доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Також Python цікавий через підвищену продуктивність, яку він забезпечує. Оскільки немає етапу компіляції, цикл редагування-тестування-налагодження неймовірно швидкий. Налагоджувати програми Python легко: помилка або неправильне введення ніколи не призведе до помилки сегментації. Натомість, коли інтерпретатор виявляє помилку, він створює виняток. Якщо програма не вловлює виняток, інтерпретатор друкує трасування стека. Налагоджувач рівня вихідного коду дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупину, перебирати код по рядку тощо. Налагоджувач написаний на самому Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидший спосіб налагодити програму — це додати кілька операторів print до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

## **2.2 .NET архітектура**

Програми на C# працюють на .NET, віртуальній системі виконання, яка називається загальномовним середовищем виконання (CLR) і наборі бібліотек класів. CLR — це впровадження Microsoft спільної мовної інфраструктури (CLI), міжнародного стандарту. CLI є основою для створення середовищ виконання та розробки, в яких мови та бібліотеки безперебійно працюють разом.

Вихідний код, написаний на C#, компілюється на проміжну мову (IL), яка відповідає специфікації CLI. Код IL та ресурси, такі як растрові зображення та рядки, зберігаються в збірці, як правило, з розширенням .dll. Збірка містить маніфест, який надає інформацію про типи, версію та культуру збірки.

Коли програма C# виконується, збірка завантажується в CLR. CLR виконує компіляцію Just-In-Time (JIT) для перетворення коду IL у власні машинні інструкції. CLR надає інші послуги, пов’язані з автоматичним збиранням сміття, обробкою винятків і керуванням ресурсами. Код, який виконується CLR, іноді називають «керованим кодом». «Некерований код» скомпільовано на рідну машинну мову, яка націлена на певну платформу.

Сумісність мов є ключовою особливістю .NET. Код IL, створений компілятором C#, відповідає загальній специфікації типу (CTS). Код IL, згенерований на C#, може взаємодіяти з кодом, згенерованим із версій .NET F#, Visual Basic, C++. Існує понад 20 інших мов, сумісних із CTS. Одна збірка може містити кілька модулів, написаних різними мовами .NET. Типи можуть посилатися один на одного, як ніби вони написані однією мовою.

На додаток до служб часу виконання, .NET також включає великі бібліотеки. Ці бібліотеки підтримують багато різних робочих навантажень. Вони організовані в простори імен, які забезпечують широкий спектр корисних функцій. Бібліотеки включають все: від введення та виведення файлів до маніпуляції з рядками до аналізу XML, до фреймворків веб-додатків до елементів керування Windows Forms. Типова програма C# широко використовує бібліотеку класів .NET для виконання звичайних «сантехнічних» справ [[11].](#_ВИСНОВКИ)

### **2.2.1 Мова програмування С#**

C# (вимовляється як «See Sharp») — сучасна, об’єктно-орієнтована та безпечна для типів мова програмування. C# дозволяє розробникам створювати багато типів безпечних і надійних програм, які працюють у .NET. C# має свої коріння в сімействі мов C і буде відразу знайомий програмістам C, C++, Java та JavaScript. Цей огляд містить огляд основних компонентів мови в C# 8 і раніше. Якщо ви хочете вивчити мову на інтерактивних прикладах, спробуйте ознайомитися з підручниками C#.

C# — це об'єктно-орієнтована, компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, роблячи C# природною мовою для створення та використання програмних компонентів. З моменту свого зародження C# додав функції для підтримки нових робочих навантажень і нових методів проектування програмного забезпечення. За своєю суттю C# є об'єктно-орієнтованою мовою. Ви визначаєте типи та їх поведінку.

Кілька функцій C# допомагають створювати надійні та довговічні програми. Збір сміття автоматично відновлює пам’ять, зайняту недоступними невикористаними об’єктами. Типи, які допускають значення NULL, захищають від змінних, які не посилаються на виділені об’єкти. Обробка винятків забезпечує структурований і розширений підхід до виявлення та відновлення помилок. Лямбда-вирази підтримують методи функціонального програмування. Синтаксис мовного інтегрованого запиту (LINQ) створює загальний шаблон для роботи з даними з будь-якого джерела. Підтримка мови для асинхронних операцій забезпечує синтаксис для побудови розподілених систем. C# має уніфіковану систему типів. Усі типи C#, включаючи примітивні типи, такі як int і double, успадковуються від одного кореневого типу об’єкта. Усі типи мають набір загальних операцій. Цінності будь-якого типу можна зберігати, транспортувати та використовувати узгоджено. Крім того, C# підтримує як визначені користувачем типи посилань, так і типи значень. C# дозволяє динамічно розподіляти об'єкти та вбудовано зберігати полегшені структури. C# підтримує загальні методи та типи, які забезпечують підвищену безпеку та продуктивність типів. C# надає ітератори, які дозволяють реалізаторам класів колекції визначати користувацькі способи поведінки для клієнтського коду.

C# робить акцент на версійності, щоб програми та бібліотеки могли розвиватися з часом сумісним чином. Аспекти дизайну C#, на які безпосередньо вплинули міркування щодо версій, включають окремі віртуальні модифікатори та модифікатори перевизначення, правила вирішення перевантаження методів і підтримку явного оголошення членів інтерфейсу.

## **2.3 База даних The Pile**

Нещодавні роботи показали, що збільшення різноманітність навчальних наборів даних покращує загальні міждоменні знання та здатність до узагальнення для великомасштабних мовних моделей. Pile – це 825 гігабайт англійських текстів, призначений для навчання великомасштабних мовних моделей. Pile створена з 22 різноманітних високоякісних підмножин - як існуючих, так і новостворених - багато з яких отримані з академічних чи професійних джерел.

Оцінка ненастроєної продуктивності GPT-2 та GPT-3 на Pile показує, що цим моделям важко справляються з багатьма компонентами, наприклад, з академічним листом. Навпаки, моделі, навчені на Pile, значно перевершують моделі Raw CC і CC-100 за всіма компонентами Pile, покращуючи при цьому результати подальших оцінок.

Основні внески цього розділу є:

1. Представлення англомовного набору даних 825.18 GiB для мовного моделювання, що поєднує 22 різноманітні джерела.

2. Представлення 14 нових наборів даних моделювання мови, які, становлять незалежний інтерес для дослідників.

3. Оцінки, що демонструють значні покращення в багатьох областях за допомогою GPT-2-розмірної моделі, навчені на цьому новому наборі даних, порівняно з навчанням на CC-100 і необробленому Common Crawl.

4. Розслідування та документування цього набору даних, який, краще інформує дослідників про те, як його використовувати, а також мотивуватиме проводити подібні дослідження.

* + 1. **Включенні бази даних**

Pile складається з 22 складових піднаборів даних (БД), як показано в Таблиці 1.

Таблиця 1 – це огляд наборів даних у Pile перед створенням витягнутих наборів.

Значення кожного з стовбців таблиці:

* *Необроблений розмір* *(Raw Size)* – це розмір перед будь-яким збільшенням або зменшенням вибірки.
* *Вага* *(Weight)* — це відсоток байтів у кінцевому наборі даних, зайнятих кожним набором даних.
* *Епохи (Epochs)* – це кількість проходів над кожним складовим набором даних протягом повної епохи над купою.
* *Ефективний розмір (Effective size)* — це приблизна кількість байтів у купі, яку займає кожен набір даних. Набори даних, позначені †, використовуються з мінімальною попередньою обробкою з попередньої роботи.

Як можна побачити збільшено вагу компонентів вищої якості, при цьому певні набори високоякісних даних, такі як Вікіпедія, відображаються до 3 разів («епохи») для кожної повної епохи над Купою (Pile). Це дозволяє зробити навчання більш змістовним і ефективним.

Таблиця 2.1 Бази данних, що входять до Pile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Компонент | Необроблений Розмір | Ваги | Епохи | Ефективний розмір | Середній розмір документа |
| Pile-CC | 227.12 ГБ | 18.11% | 1.0 | 227.12 ГБ | 4.33 кБ |
| PubMed Central | 90.27 ГБ | 14.40% | 2.0 | 180.55 ГБ | 30.55 кБ |
| Books3' | 100.96 ГБ | 12.07% | 1.5 | 151.44 ГБ | 538.36 кБ |
| OpenWebText2 | 62.77 ГБ | 10.01% | 2.0 | 125.54 ГБ | 3.85 кБ |
| ArXiv | 56.21 ГБ | 8.96% | 2.0 | 112.42 ГБ | 46.61 кБ |
| Github | 95.16 ГБ | 7.59% | 1.0 | 95.16 ГБ | 5.25 кБ |
| FreeLaw | 51.15 ГБ | 6.12% | 1.5 | 76.73 ГБ | 15.06 кБ |
| Stack Exchange | 32.20 ГБ | 5.13% | 2.0 | 64.39 ГБ | 2.16 кБ |
| USPTO Backgrounds | 22.90 ГБ | 3.65% | 2.0 | 45.81 ГБ | 4.08 кБ |
| PubMed Abstracts | 19.26 ГБ | 3.07% | 2.0 | 38.53 ГБ | 1.30 кБ |
| Gutenberg (PG-19) | 10.88 ГБ | 2.17% | 2.5 | 27.19 ГБ | 398.73 кБ |
| OpenSubtitles | 12.98 ГБ | 1.55% | 1.5 | 19.47 ГБ | 30.48 кБ |
| Wikipedia (en) | 6.38 ГБ | 1.53% | 3.0 | 19.13 ГБ | 1.11 кБ |
| DM Mathematics | 7.75 ГБ | 1.24% | 2.0 | 15.49 ГБ | 8.00 кБ |
| Ubuntu IRC | 5.52 ГБ | 0.88% | 2.0 | 11.03 ГБ | 545.48 кБ |
| BookCorpus2 | 6.30 ГБ | 0.75% | 1.5 | 9.45 ГБ | 369.87 кБ |
| EuroParl | 4.59 ГБ | 0.73% | 2.0 | 9.17 ГБ | 68.87 кБ |
| HackerNews | 3.90 ГБ | 0.62% | 2.0 | 7.80 ГБ | 4.92 кБ |
| YoutubeSubtitles | 3.73 ГБ | 0.60% | 2.0 | 7.47 ГБ | 22.55 кБ |
| PhilPapers | 2.38 ГБ | 0.38% | 2.0 | 4.76 ГБ | 73.37 кБ |
| NIH ExPorter | 1.89 ГБ | 0.30% | 2.0 | 3.79 ГБ | 2.11 кБ |
| Enron Emails | 0.88 ГБ | 0.14% | 2.0 | 1.76 ГБ | 1.78 кБ |
| The Pile | **825.18 ГБ** |  |  | **1254.20 ГБ** | **5.91 кБ** |

**2.3.2 Порівняльний аналіз мовних моделей за допомогою Pile**

Хоча Pile був задуманий як навчальний набір даних для великомасштабних мовних моделей, його охоплення кількох різнорідних доменів робить його також придатним як набір даних для оцінки. У цьому розділі описується, як Pile можна використовувати як набір даних широкого охоплення для порівняльних мовних моделей.

* + - 1. Рекомендації з порівняльного аналізу

Pile надається як розділи для навчання, перевірки та тестування. Компоненти перевірки та тестування містять по 0,1% даних, відібраних рівномірно випадковим чином. Хоча це набагато менший відсоток ніж більшість наборів даних, самий розмір набору даних призводить до понад 1 ГіБ даних перевірки та тестування кожен. Тому підкреслимо, що, хоча було докладено зусиль для видалення дублікатів документів у купі, все ще можливо, що деякі документи дублюються в розділах тенування/підтвердження/тестування.

Бажаним показником є біти на байт, що закодовано в UTF-8 (BPB) формі. Під час використання Pile як метрики перевага над бітами на байт надає перевагу перед бітами на символ або незрозумілістю через його незмінність до різних схем токенізації та неоднозначність вимірювання символів у Unicode. Щоб обчислити біти на байт із даного негативного *log* втрати ймовірності *l*, обчислимо *BPB = (LT /LB) log2 (e ` ) = (LT /LB)`/ ln(2*), де *LT* – довжина набору даних у токенах, а *LB* – довжина набору даних у байтах, закодованих UTF-8. Було виявилено, що *LT /LB* становить 0,29335 GPT-2-токенів/байт по всій купі;

* + - 1. Тестування Perplexity за допомогою GPT-2 і GPT-3

Обчислимо тестову заплутаність складових наборів даних Pile за допомогою GPT-2 (Radford et al., 2019) і GPT-3 (Brown et al., 2020), показаних на малюнку 2. Використаємо всі доступні версії GPT- 2, і всі чотири версії GPT-3 доступні через OpenAI API. Через витрати, пов’язані з використанням OpenAI API, оцінимо одну десяту відповідних наборів тестів для більшості складових наборів даних. Повідомляємо про незрозумілість, перетворену в біти на байт кодування UTF-8 (BPB). Важливо те, що обчислюючи заплутаність, оцінюють кожен документ незалежно від кожного набору даних, на відміну від конкатенації всіх документів, як це є звичайною практикою для обчислення незрозумілості на великих корпусах. Повну інформацію про обчислення плутанини можна знайти в Додатку.

Не дивно, що більші мовні моделі, як правило, викликають меншу заплутаність порівняно з меншими моделями. Останні роботи показали підвищену увагу до емпіричних законів масштабування мовних моделей (Kaplan et al., 2020; Henighan et al., 2020). Таким чином, було досліджено закон масштабування для сімейств моделей GPT-2 і GPT3 щодо оцінки незрозумілості на Pile. Відношення закону масштабування для сімейства моделей GPT-3 показано на рисунку 2. Лінія найкращого підходу, показана на малюнку, має коефіцієнт -0,1674 і відрізок 2,5516.

Цікаво, що хоча GPT-2 і GPT-3 не були навчені на Палі, все ще існує чіткий закон масштабування без зменшення віддачі. Було припущено, що це пов’язано з властивою цим моделям здатністю до узагальнення. Тож залишимо більш ретельний аналіз законів масштабування нульового режиму для майбутньої роботи.

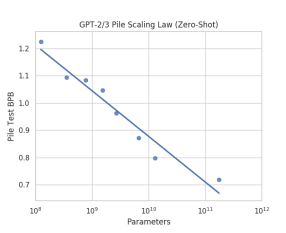


Рис. 2.5 Закон масштабування для виконання моделей GPT-2/3. «Нульовий знімок» відноситься до того факту, що жодна з моделей не була точно налаштована на даних з Пали.

Параметри.

GPT-2/3 Pile Низьке масштабування (нульовий постріл).

Pile тест Бітів в байті.

* + - 1. Відносна по компонентна продуктивність GPT-3 Pile

Визначення того, на яких компонентах GPT-3 не працює, надає інформацію про те, які компоненти Pile найбільше відрізняються від розподілу тексту (веб-сторінки та книги), на якому навчався GPT-3. Таким чином, ці компоненти стануть особливо хорошими кандидатами для доповнення навчальних даних GPT-3. Ці результати також є цінними для визначення типів наборів даних для майбутніх ітерацій Pile.

Зображення, що містить текст

Автоматично згенерований опис Через різницю в ентропії різних наборів даних, пряме порівняння збентеження GPT-3 на різних компонентах Pile не є точним показником відносної продуктивності. В ідеалі ми б тренували модель GPT-3 з нуля на Pile і порівнювали б різницю втрат на набір даних з оригінальною GPT-3. Через обмеження ресурсів ми замість цього використовуємо модель GPT-2, навчену з нуля на Палі, щоб побудувати проксі-захід. Щоб створити проксі-сервер, спочатку було виміряно покращення від моделі GPT2-Pile до GPT-3 для кожного компонента. Потім було нормалізовано результати, встановивши зміну OpenWebText2 на нуль. Цей розрахунок показано в рівнянні нижче:

Оскільки GPT2-Pile навчався як на OWT2, так і на наборі даних, який було оцінено, очікується, що другий член у наборі ∆ відображатиме різницю у внутрішній складності двох наборів даних. Таким чином, загальне значення ∆set відображає, наскільки складнішим був набір даних, який було оцінено, для GPT-3, ніж OWT2, мінус відносна складність двох завдань. Оскільки GPT-3 навчався на даних, дуже подібних до OWT2, це дає нам проксі-сервер щодо того, наскільки краще GPT-3 був би, якби його навчали на Pile.

Результати показані на малюнку 3. Для перевірки працездатності було помічено, що набори даних, які містяться в навчальному наборі GPT-3 (Books3, Wikipedia (en), Pile-CC і Project Gutenberg) або дуже схожі на нього, або близькі до нуля за нашою метрикою.

Схоже, що GPT-3 погано працює з наборами даних, що стосуються досліджень або академічних писем, таких як PubMed Central, PubMed Abstracts і ArXiv; специфічні для домену набори даних, такі як FreeLaw, HackerNews та USPTO Backgrounds; і на наборах даних, що містять переважно текст, відмінний від природної мови, як-от GitHub і DM Mathematics.

Крім того, більшість наборів даних мають менше покращень, ніж OpenWebText2. Таким чином, очікується, що модель розміру GPT-3, навчена на Pile, буде працювати значно краще з дослідницькими завданнями, програмними завданнями та завданнями маніпулювання символами, ніж базова модель. Крім того, цей експеримент надає докази того, що більшість компонентів Pile не є зайвими з переважно веб-базованими навчальними даними GPT-3.

Зазначимо, що ця метрика є лише проксі-сервером для подібності, і що вона може бути заплутана ефектами масштабування конкретних наборів даних. Хоча вихідні результати значною мірою відповідають очікуванням, є деякі незрозумілі результати, як-от набори даних, за якими GPT-3 перевершив GPT-2 Pile. Припускається, що GPT-3 вчиться настільки добре володіти цими наборами даних, що навчання на них явно не впливає на продуктивність моделі.

* + 1. **Оцінка**

Щоб підтвердити ефективність Pile для покращення якості моделювання мови, архітектурно ідентичних моделей навчалася з 1,3 мільярдами параметрів на основі моделей Brown et al. (2020) на різних наборах даних та оцінювала завдання WikiText та LAMBADA як еталони здібностей мовного моделювання.

* + - 1. Методологія

Щоб забезпечити справедливе порівняння між наборами даних різного розміру, було знезаражено будь-які екземпляри оцінювальних наборів, використовуючи 13-грамову фільтрацію перекриття, як у Brown et al. (2020 р.) , а також була зменшена вибірка до 40 ГБ, щоб контролювати розмір набору даних. Оскільки розмір набору даних можна контролюємо, то оцінка для CC-100 (en) є більш ніж щедрою, і становить приблизно 1/3 від розміру купи (Pile).

Було порівняно такі набори даних: Pile, англійський компонент набору даних CC-10011 (Wenzek et al., 2019; Conneau et al., 2020) і зразок необроблених файлів CC WET, відфільтрованих лише для англійської мови.

* + - 1. Результати

У тестах традиційного мовного моделювання Pile значно покращується щодо WikiText і демонструє незначні зміни в LAMBADA. Однак моделі, навчені на Pile, значно покращуються як у Raw CC, так і в CC-100 на всіх компонентах Pile, як показано в таблиці. Це вказує на те, що моделі, навчені на Pile, мають кращі можливості міждоменного узагальнення без шкоди для продуктивності на традиційних тестах. Величина покращення порівняно з CC-100 за комплект показана на малюнку. Як не дивно, у Pile-CC майже немає покращень. Однак модель, навчена на Pile, працює значно краще, ніж будь-яка з інших моделей на академічних наборах даних, таких як ArXiv, Pubmed Central, FreeLaw і PhilPapers. Він також значно покращує набори даних, пов’язані з програмуванням, такі як Github і StackExchange, на EuroParl через відсутність багатомовного тексту в будь-якому іншому наборі даних, а також на DM Mathematics, що вказує на значне покращення в математичних здібностях.

Дивно, але необроблений Common Crawl працює краще на Pile BPB, ніж CC-100, незважаючи на те, що він програє зі значним відривом від LAMBADA та WikiText. Ми припускаємо, що це пов’язано з фільтрацією на основі збентеження, яка використовується в CC-100, коли мовна модель тренується у Вікіпедії, а всі дані із занадто високим або надто низьким значенням плутанини відкидаються. Це ефективно відкидає будь-які дані, надто схожі чи занадто відрізняється від Вікіпедії, що суттєво обмежує різноманітність зібраних даних. Цей результат свідчить про те, що в подальшій роботі з використанням Common Crawl слід бути обережними з фільтруванням, щоб зберегти його різноманітність.

# **ВИСНОВКИ ДО РОЗДІЛУ 2**

Комп’ютерний або штучний інтелект глибоко впливає на світ, у якому живемо, і кількість нових додатків неухильно зростає. Блискучі дизайнери вибирають Python як мову програмування завдяки різноманітним перевагам, які роблять його особливо придатним для AI та глибокого навчання.

Широкий вибір відкритих бібліотек і структур у Python AI роз’єднує процедуру вдосконалення та скорочує час просування. Основна граматика і зрозумілість Python сприяють швидкому тестуванню складних обчислень і роблять мову відкритою для не розробників.

Це також зменшує психологічні витрати на інженерів, відкриваючи їхні розумові активи з метою, щоб вони могли зосередитися на критичному мисленні та досягненні цілей підприємства. Нарешті, проста пунктуація спрощує спільну роботу або переміщення між дизайнерами.

Крім того, Python хизується широкою, динамічною мережею дизайнерів, які раді запропонувати допомогу та підтримку, що може бути важливо при управлінні такими складними підприємствами.

У той час як інші діалекти програмування також можуть бути використані в підприємствах з штучним інтелектом, неможливо уникнути того, що Python знаходиться на передовій, і про нього слід критично подумати. Це причина, чому ви повинні розглянути Python для свого підприємства з штучним інтелектом.

# **РОЗДІЛ 3 ВИМОГИ ДО ВИДУ АРХІТЕКТУРИ**

Дотримання вимог до архітектури допоможе зберегти багато часу й нервів під час розробки, адже обладнання може як прискорювати розробку, роблячи її надзвичайно швидкою, так і завдати клопоту, коли комп’ютер просто не може впоратися з занадто складними задачами для його архітектури.

Якщо портібно щось, що можна легко брати з собою на роботу і назад, або взяти з собою в університет, тоді підійде легкий ноутбук. Вони важать всього пару фунтів, мають батареї, які можуть витримати повний робочий день понад 8 годин, і мають більш ніж достатньо потужності, щоб вирішувати звичайні завдання, такі як перегляд веб-сторінок та редагування документів Microsoft Office. Деякі моделі, відомі як 2-в-1, можна використовувати як свого роду планшет, відігнувши клавіатуру назад і за екраном за допомогою шарніра на 360 градусів.

Настільний комп’ютер може бути розумнішим і ергономічним вибором, однак, якщо говорити про обладнання домашнього офісу і не турбуєтеся про мобільність. Дивлячись у дисплей, буде значно менше навантаження на шию, ніж дивлячись вниз, як на ноутбук. Деякі моделі оснащені вбудованим дисплеєм, а в інших доведеться поставити власний. На щастя, комп’ютерні монітори не дорогі. Крім того, хоча багатофункціональні пристрої (де екран і комп’ютер є єдиним блоком) естетично привабливі і можуть займати менше місця на столі, вони можуть коштувати набагато дорожче, ніж покупка настільного комп’ютера та монітора окремо.

І хоча велика дискусія між Mac і ПК навряд чи закінчиться, Mac може зробити приблизно стільки ж, скільки і ПК (хоча вам знадобиться ПК, якщо говорити про роботу над великими проектами, архітектурою й AAA-контентом). Обидва мають доступ до програмного забезпечення найвищого рівня, мають надійний захист від шкідливих програм і прості у використанні: наприклад, Google Chrome і Microsoft Word в основному однакові на комп’ютерах Mac і ПК.

Тим не менш, можливість з’єднати Mac з iPhone або телефон Android з ПК може бути надто гарною, щоб відмовитися від цього. Це дозволить відповідати на текстове повідомлення за допомогою клавіатури та обмінюватися даними копіювання та вставки між комп’ютером і смартфоном. І це лише кілька прикладів.

Для випадку ж розробки AI та Unity доцільніше обрати стаціонарний ПК з двома моніторами. Про системні вимогу поговоримо нижче.

## **3.1 Системні вимоги для Unity 2021.2**

Розробка ігор та програм під Unity не потребує великих потужностей, особливо при розробці під мобільну платформу, або веб. Якщо ж працювати з комп'ютерними іграми в 3D, то доведеться витратитися на відповідну відеокарту - її потужність залежить від "тяжкості" графіки, яка планується використовуватися у проекті. Також для VR й AR проектів потрібно заздалегідь потурбуватися про відповідні вимоги до архітектури персонального комп’ютера (ПК).

### **3.1.1 Мінімальні системні вимоги Unity:**

**Графічний процесор (GPU):** відеокарта з можливостями DX10 (шейдерна модель 4.0), , DX11, або DX12.

**Центральний процесор (CPU):** Потрібен багатоядерний процесор Intel або AMD із підтримкою набору інструкцій SSE2. Але Pentium 4 – останню модель, яку їх не підтримувала – перестали випускати ще 2008 року.

**Операційна Система:**

* лише версії Windows 7 SP1+, 8, 10, 64-розрядні версії;
* macOS 10.12+;
* Ubuntu 16.04, 18.04;
* CentOS 7.

### **3.1.2 Комп’ютер для Unity:**

**Процесор для Unity**

Найвитратніші за часом процеси – це обробка файлів, робота зі світлом та підсумкова компіляція. У всіх них бере участь процесор і накопичувач. Чим швидше все це працюватиме, тим менше часу доведеться витрачати даремно, тому процесор – один із найважливіших компонентів робочого комп'ютера для розробки в Unity. Для виконання інтенсивних завдань, яким потрібно багато часу, знадобиться процесор з великою кількістю ядер.

Для більшості користувачів чудовим вибором буде 12-ядерний AMD Ryzen 9 5900X. Цей процесор має 12 ядер та 24 потоки, що забезпечує відмінну продуктивність. Якщо дозволяє бюджет, можна придивитися до 16-ядерного Ryzen 5950X.

Тим, кому потрібна максимально можлива продуктивність для багатопотокових завдань, таких як робота з освітленням або компіляція, ми рекомендуємо такі процесори. AMD Threadripper 3960X на 25% швидше, ніж Ryzen 5950X у таких завданнях, а Threadripper 3970X та 3990X будуть ще швидше. Зверніть увагу, що ці процесори можуть бути трохи повільнішими в інших завданнях, тому рекомендуємо Threadripper тільки тим користувачам, які витрачають значно багато часу на створення освітлення або компіляцію.

Якщо ж говорити про Intel процессорти, то гарним рішенням для розробки на Unity є:

Нова серія Intel Core i5-10600K має 6 ядер з 12 потоками, частота ядра 4,1 GHz у штатному режимі та 4,8 GHz при включеному Turbo Boost.

Більш просунута модель Intel Core i7-10700K, яка працює на 8 ядрах з 16 потоками, частота 3,8/5,1 GHz. Досить потужний помічик, що швидко впорається з будь-якими завданнями.

**Відеокарта для Unity**

Unity використовує відеокарту виключно для відображення графіки на екрані. Багато професійних програм в інших областях вже почали використовувати відеокарту для інших завдань, але в Unity це ще не реалізовано. Через це більш потужніша відеокарта дасть вам вищий FPS на екрані монітора, але не покращить продуктивність у розробці.

Рекомендовано наступні відеокарти для робочого комп'ютера, залежно від бюджету та того, чи плануєте розробку VR-контенту:

NVIDIA GeForce RTX 3070 8 ГБ – ця відеокарта пропонує відмінну продуктивність і має достатню потужність для роботи з кількома дисплеями одночасно.

NVIDIA GeForce RTX 3090 24 ГБ на даний момент є однією з найкращих відеокарт для розробки ігор, VR та архітектурної візуалізації. Великий обсяг відео пам’яті робить цю карту придатною для робочої станції з трьома або навіть чотирма 4K-моніторами, а додаткова потужність відмінно підходить для ігор зі слабкою оптимізацією.

**Оперативна пам'ять для Unity**

Хоча точний обсяг необхідної оперативної пам'яті залежатиме від конкретних проектів і від того, чи виконуються завдання, які потребують великого обсягу оперативної пам'яті, наприклад, освітлення будівлі. Але рекомендовано:

* 32 ГБ оперативної пам'яті для більшості користувачів
* 64 ГБ + ОЗУ, для тих, хто створює освітлення

Тож загальні вимоги це 16-32 Gb, якщо плануєте працювати з маленькими та середніми іграми та 32-64 для великих ігор й проектів AAA.

Майте на увазі, що це базові рекомендації і вони охоплюють лише обсяг оперативної пам'яті, необхідної для Unity. Якщо при розробці часто працює одночасно й інші програмами паралельно з Unity, то може знадобитися ще більше ОЗП у вашій системі, тому що кожній програмі буде потрібно окремий фрагмент ОЗП.

**Сховище (жорсткі диски) для Unity**

Unity може бути не найважчим додатком для зберігання, але все ж таки важливо мати швидке і надійне сховище, щоб не відставати від решти системи.

Який тип накопичувача слід використовувати для Unity:

Існує три основні типи накопичувачів, які можна використовувати: SSD, NVMe і традиційні жорсткі диски.

* З цих трьох традиційні жорсткі диски є найповільнішими, але вони дешеві та доступні з набагато більшою ємністю, ніж твердотільні накопичувачі або накопичувачі NVMe. Завдяки цьому виходять чудові накопичувачі для тривалого зберігання файлів.
* Твердотільні накопичувачі SATA у кілька разів швидше за жорсткі диски, але при цьому вони дорожчі. Ці диски чудово підходять для широкого кола завдань, таких як зберігання вашої операційної системи та програм, а також зберігання проектів, над якими ви працюєте.
* NVMe-накопичувачі бувають двох видів (M.2 та U.2), і вони значно швидше, ніж SATA-SSD. Вони, як правило, дорожчі, ніж твердотільні накопичувачі SATA, але зате можуть бути в дванадцять разів швидше! У більшості випадків ви не побачите значного збільшення продуктивності з диском NVMe, оскільки сучасний стандартний твердотільний накопичувач вже досить швидкий і рідко є вузьким місцем продуктивності, але оскільки вартість цих дисків продовжує знижуватися, їх можна використовувати як диск з операційною системою та програмами, щоб вони запускалися трохи швидше.

Тож підсумовуючи можна сказати, що SSD m.2 останнього покоління зробить процес обробки даних дуже комфортним, у тандемі з потужним процесором завантаження/вивантаження файлів, компіляція, запікання світла відбуватиметься практично миттєво.

## **3.2 Системні вимоги для GPT-J**

Для висновку моделі потрібно приблизно 12,1 ГБ. Щоб запустити на графічному процесорі потрібна карта NVIDIA з принаймні 16 ГБ оперативної пам’яті, а також принаймні 16 ГБ оперативної пам’яті процесора для завантаження моделі. Також можна запустити лише на центральному процессорі (ЦП), тоді вам знадобиться лише 16 ГБ звичайної оперативної пам’яті, але час виконання буде набагато довшим.

# **ВИСНОВКИ ДО РОЗДІЛУ 3**

Наше покоління бачило комп’ютери різних типів і розмірів. Від суперкомп’ютерів, які використовуються для зберігання, керування та декодування даних супутникової передачі, до мейнфреймів і веж, які утворюють основу великих корпоративних онлайн- та офлайн-організацій, а потім до міні-веж, які надають послуги на сервері меншим організаціям, аж до мікрокомп’ютерів (інакше відомих як персональні комп’ютери), які здатні задовольнити індивідуальні інформаційні та комунікаційні потреби (будь то настільна комп’ютерна система чи ноутбук). Комп’ютери — це саме те, за чим стоять майбутні професій і вибір чого відіграє велике значення у ефективності й зручності виконання справ у повсякденному житті.

Хоча настільні комп’ютери сьогодні менш дорогі та набагато менш популярні, оскільки попит на ноутбуки (нетбуки та ноутбуки) вищий, правда полягає в тому, що «все ще можна знайти чудовий настільний комп’ютер у ці дні, якщо знати, на що звернути увагу ».

Зростаючі потреби в інформації, комунікації та технологіях постійно підтверджують той факт, що ПК більше не є розкішшю, а неминучою необхідністю. А тепер велике питання – «Як вибрати комп’ютерну систему, яка відповідає повсякденним потребам?» Багато приватних осіб та бізнес-організацій витратили статки на придбання в різний час комп’ютерних систем, які у довгостроковій перспективі, не були пристосовані до їхніх потреб. Але хороша новина полягає в тому, що незалежно від того, ким кінцевий користувач є чи графічним дизайнером, програмістом, науковцем, бухгалтером чи будь-якими організаційними чи корпоративними потребами, у нього є відповідний комп’ютер. І вибирати ПК потрібно з розумом, розраховуючи де й за яких обставин він буде використаний, скільки оперативної пам’яті буде потрібно і яка відеокарта й процесор.

В випадку ж роботи над ШІ з Unity потрібен найбільш прогресивний ПК: 32 гб ОЗУ, процесор Intel i5 покоління, відеокарта gtx 1650 й 1 тб SSD пам’яті. Це дозволить максимально зручно працювати над проектом.

# **РОЗДІЛ 4 ІНТЕГРАЦІЯ UNITY3D І СИСТЕМА ШТУЧНОГО ІНТЕЛЕКТУ ІНТЕРАКТИВНОГО ПОМІЧНИКА**

## **4.1 Встановлення і налаштування GPT-J**

### **4.1.1 Встановлення GPT-J на комп’ютер**

Першим кроком буде завантаження репозиторія з GPT-J на комп’ютер. Для цього використаємо GitTurtoise і посилання на Git репозиторій GPT-J-6B Notebook [[13](#_ВИСНОВКИ_1)].

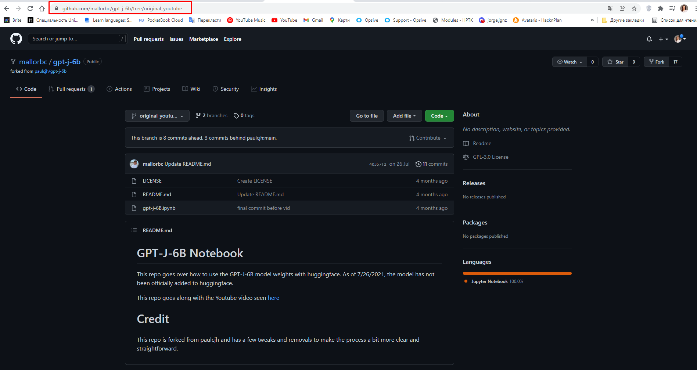


Рис. 4.1 Копіювання посилання на репозиторій

У цьому репозиторії розповідається про те, як використовувати гирі моделі GPT-J-6B з huggingface. Спочатку модель офіційно не підтримувалася huggingface, але ваги все ще можна було використовувати з huggingface іншими методами. Модель зараз офіційно підтримується, набагато легша у використанні, і ноутбук був змінений відповідно до цього.

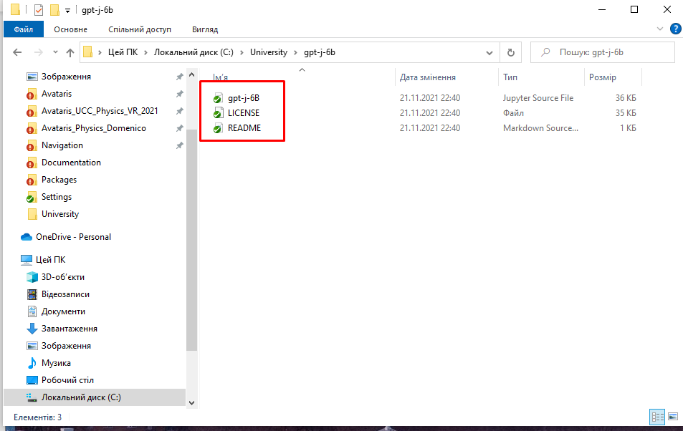
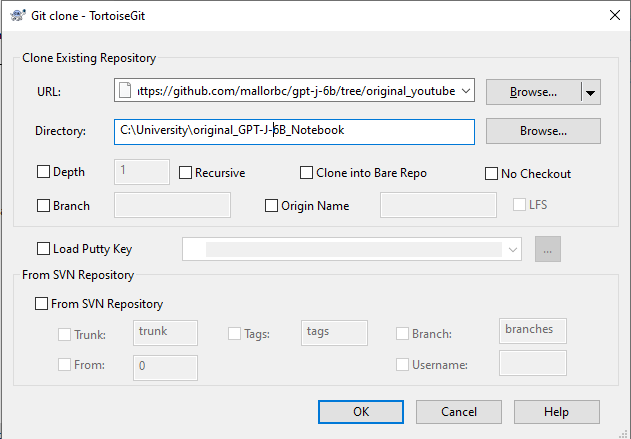
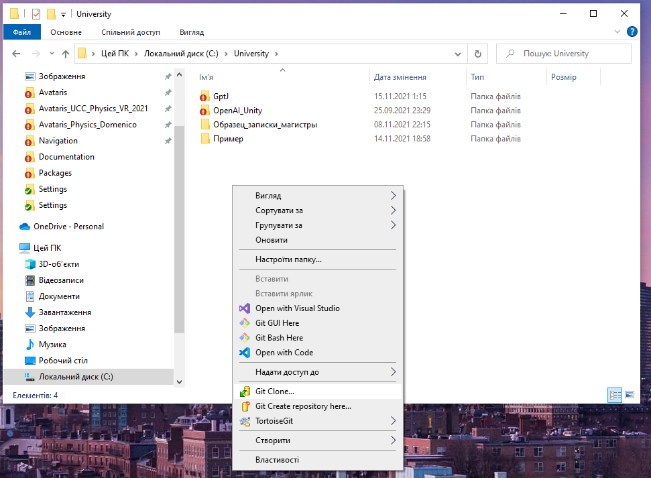


Рис. 4.2 Клонування репорзиторію за допомогою GitTurtois

Для цього клікнемо праву клавішу миші у папці,де розмістимо вміст репозиторію й з випадаючого списку виберемо “Git Colne…”. Вставимо зкопійований адрес Git репозиторія, а також назву папки, куди ми помістимо данні. Натиснемо ок й зачекаємо поки данні завантажаться.

Тепер ми маємо 3 файли в папці.

* License – це ліцензія на використання моделі.
* Gpt-j-6B – модель данних GPT-J
* Readme – опис проекту git

**Встановлюємо Python 3.9 на комп’ютер**

Встановимо Python 3.9 на комп’ютер. Для цього в пошуку напишемо python й відкриємо офіційний сайт [[14](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]. Перейдемо у розділ для windows програм й оберемо 64 бітну версію. Натиснемо на з’явившийся посилання для завантаження файлу, після завантаження встановимо Python на диск С, в заздалегідь створену папку з відповідним ім’ям.

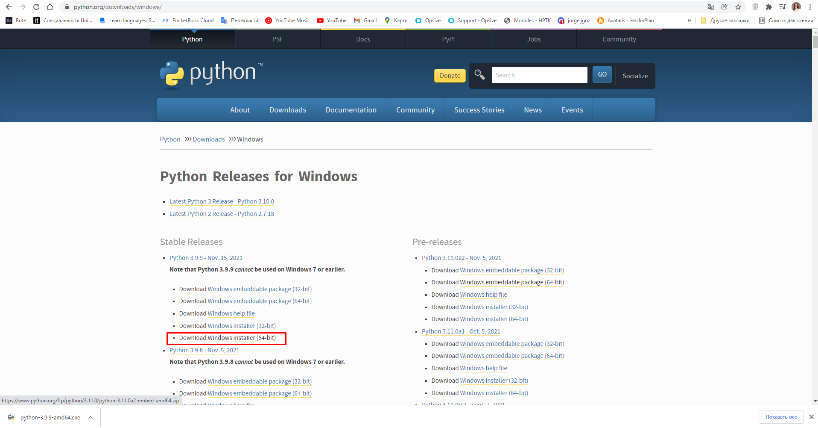
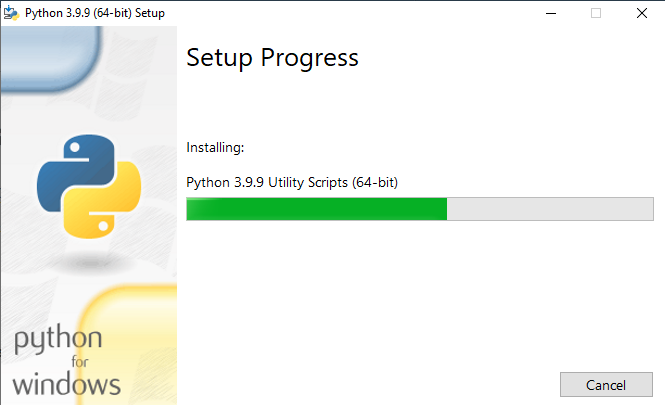


Рис. 4.3 Завантаження у встановлення Python 3.9



Перевіримо чи працює Python. Для цього выдкриємо командний рядок й напишемо python. Бачимо, що консоль вивела версію Python, що зараз встановлена на ПК. Спробуємо вивести щось за допомогою функції ptint(‘’), щоб перевірити чи точно все працює. Як бачимо консоль виводить результат, отже все встановлено правильно. Вийдемо за допомогою команди exit().

Тепер перевіримо чи прописано шлях до python у Системних змінних оточення. Для цього відкриємо вікно з інформацією про Змінні оточення, вибиремо Path й клікнемо 2 рази по ньому. З’явиться інформація про наявні шляхи, перевіримо чи є тут Python. Отже, бачимо, що все як портібно бути й шлях наявний. Тож встанвлення й усі змінні що потрібні для роботи з Python є.

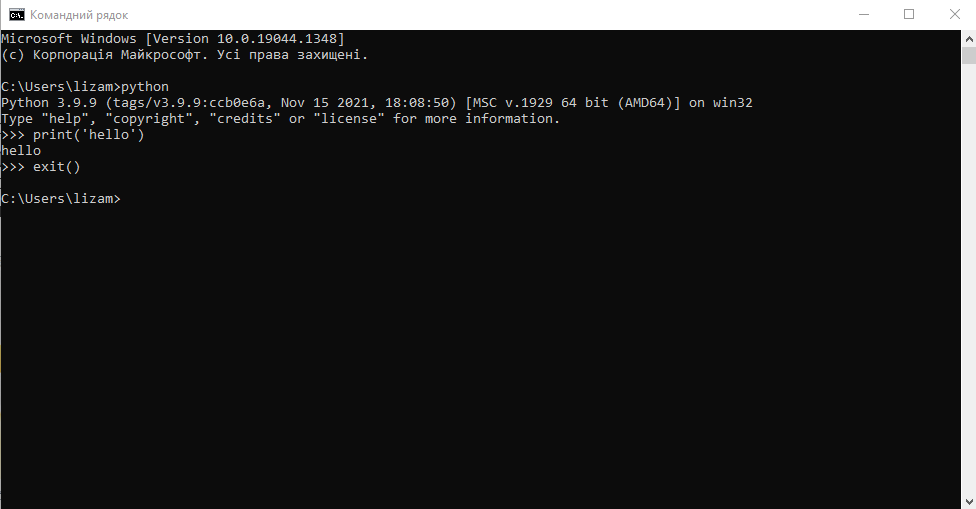


Рис. 4.4 Перевірка роботи Python

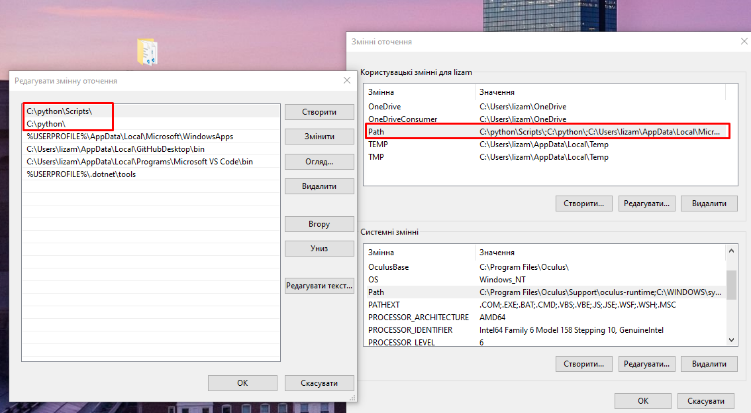


Рис. 4.5 Перевірка Змінних оточення

**Встановлення й налаштувння Anaconda Python, Jupyter Notebook і Spyder**

Перейдемо до офіційного сайту Анаконда [[15](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)] й встановимо додаток на комп’ютер.

Після встановлення Anaconda, відкриємо Anaconda Navigator й перевіримо чи встановлено Jupyter Notebook і Spyder. Бачимо, що під відповідними фконками напс Launch, а онже фони наявні. Також перевіримо командний рядок, ввівши Anaconda Prompt в пошуковому рядку й натиснемо кнопку ентер. Бачимо, що відкрилося комадне вікно. Введемо команду conda info для перевірки працездатності додатку. Бачимо що після застосування комани ми отримали інформацію про наявну версію Anaconda, її середовище, місцеположення, посилання та багато іншої інформації, а отже бачимо, що все працює як треба

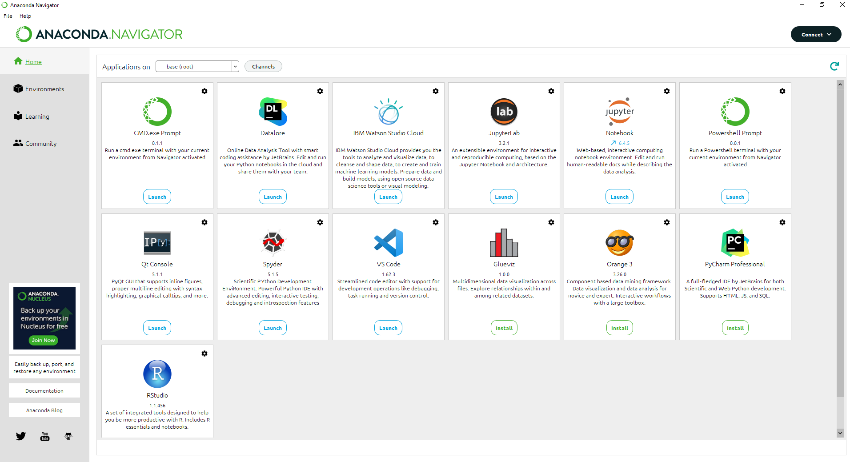


Рис. 4.6 Anaconda Navigator

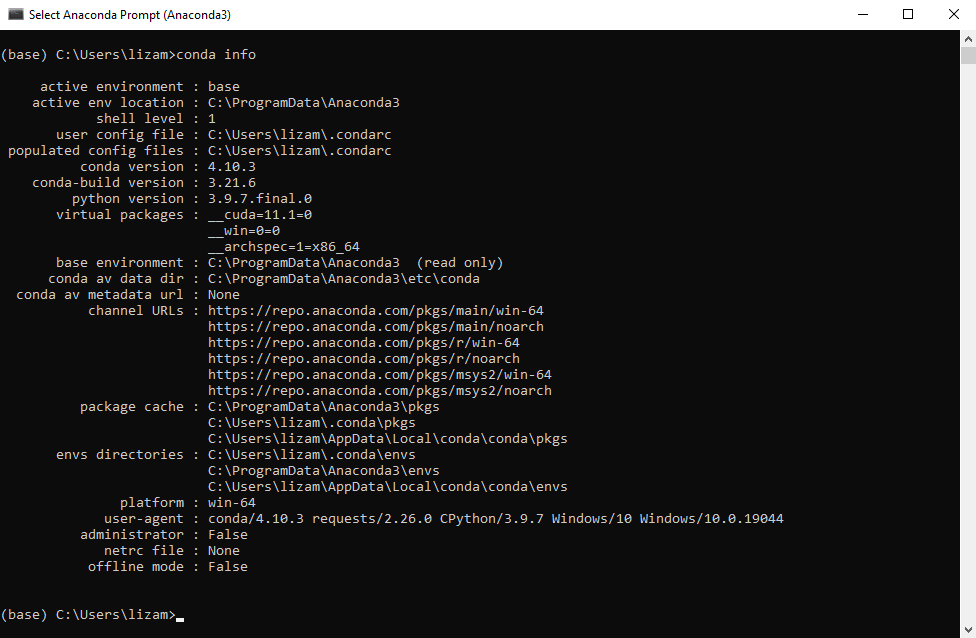


Рис. 4.7 Anaconda командний рядок

### **4.1.2 Налаштування середовища**

Запустимо командний рядок Anaconda й створимо нове середовище для роботи з python 3.9. Для цього введемо команду:

conda create -n gptneo6B python=3.9

Буде створено середовище для Python 3.9 з ім’ям gptneo6B. Після закінчення інсталяції – відповімо y для того щоб продовжити далі.

Далі активуємо середовище за допомогою команди:

conda activate gptneo6B

Після активації, встановимо Jupyter Notebook, за допомогою команди:

conda install jupyter

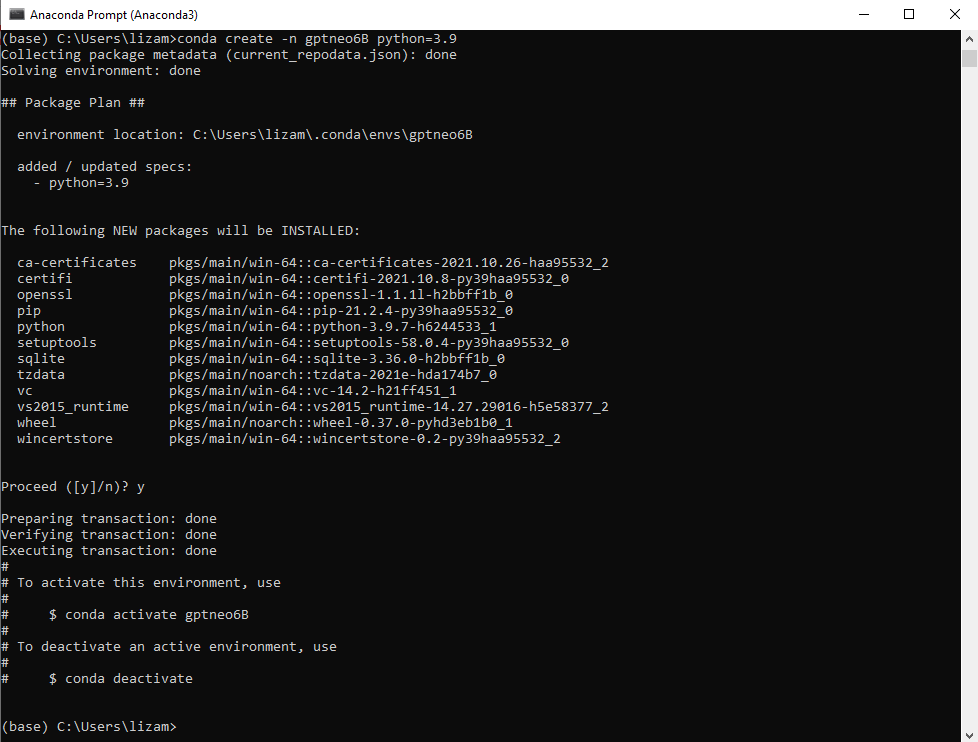


Рис. 4.8 Створення середовища для GPT-J

На питання *“Proceed([y]/n)”*? Відповідаємо *y*, і чекаємо на встановлення необхідних пакетів.

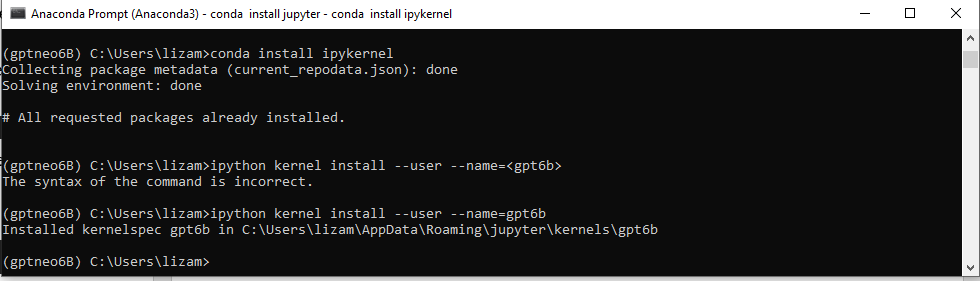


Рис. 4.9 Встановлення Kernel в середовищі Conda

Далі встановимо ноутбук Kernel в середовищі Conda:

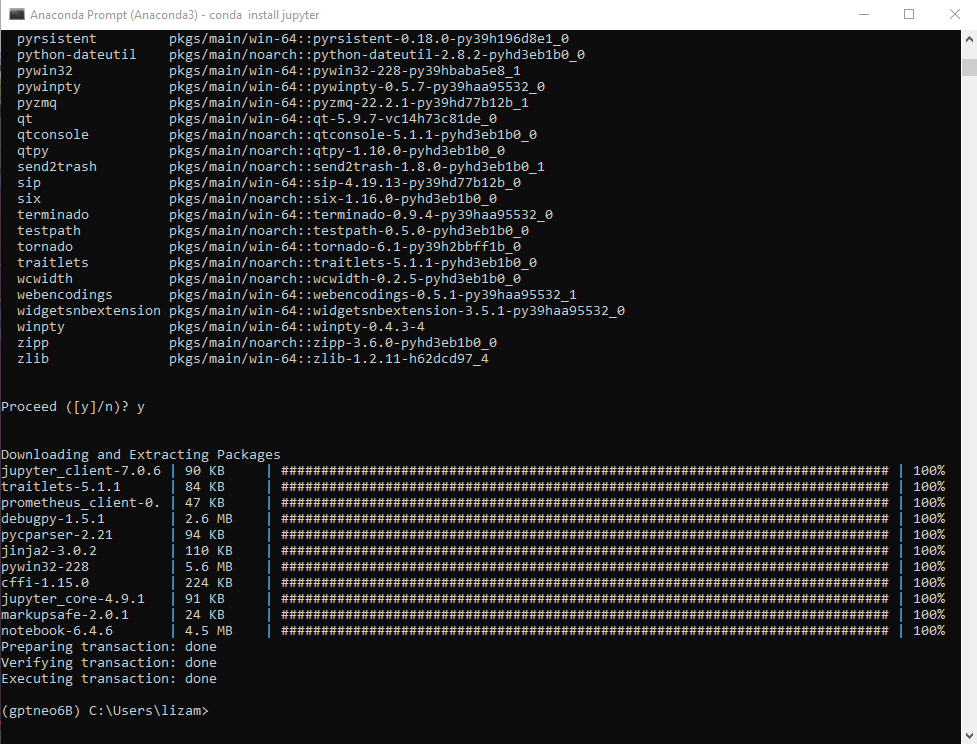
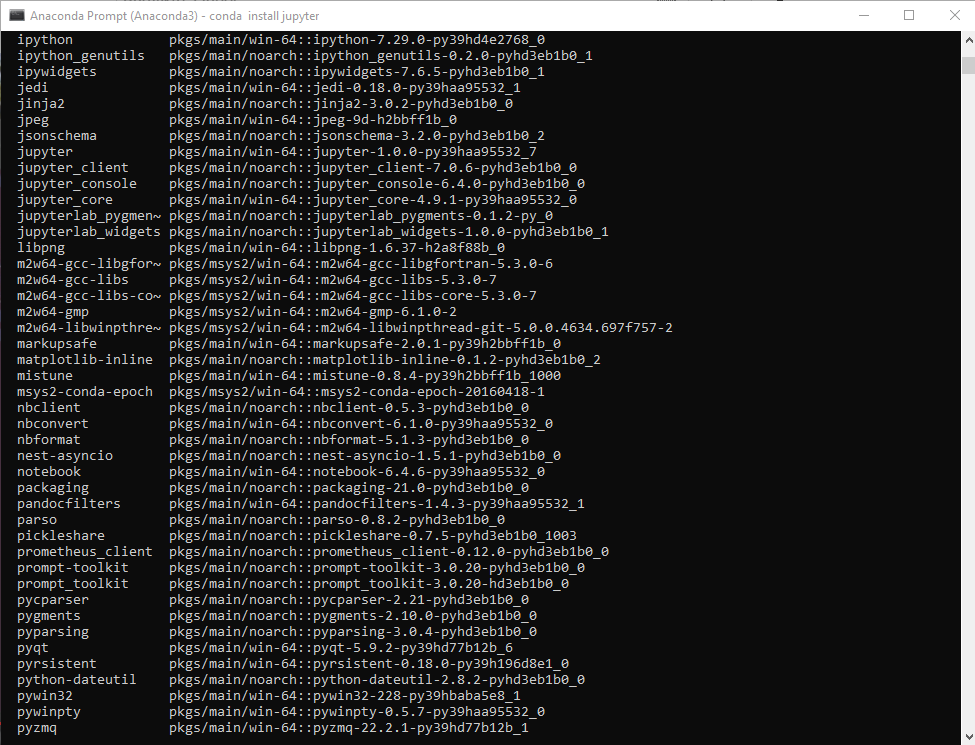
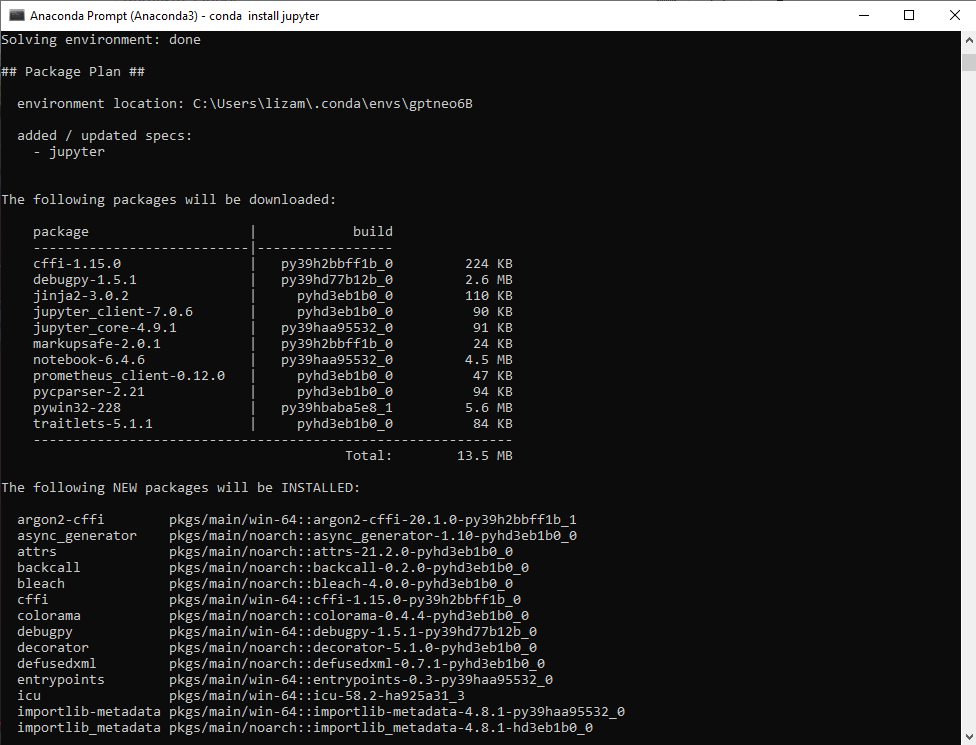


Рис. 4.10 Встановлення Jupyter Notebook й необхідних пакетів

conda install ipykernel

Й задамо ім’я користувача:

ipython kernel install --user --name=<any\_name\_for\_kernel>

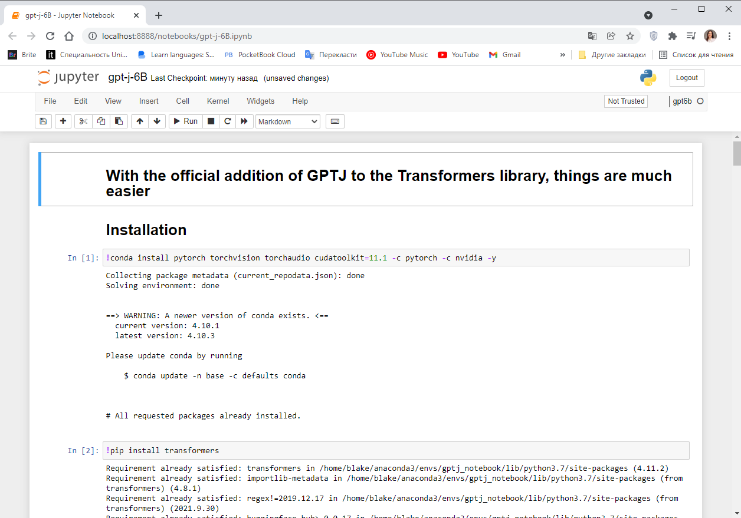


Рис. 4.11 gptj-6B відкритий у Jupyter Notebook

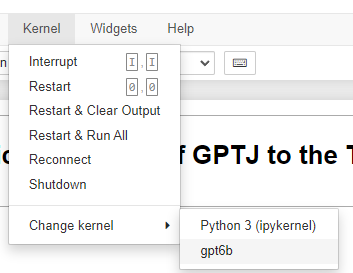


Рис. 4.12 Налаштування Kernel

### **4.1.3 Запуск Jupyter Notebook**

Для запуску Jupyter Notebook використаємо наступний рядок:

Jupyter notebook

Далі вибираємо файл, що було завантажено з репозиторія Git (gpt-j-6B) й клікаємо по ньому. Для Kernel вибираємо gpt6b.

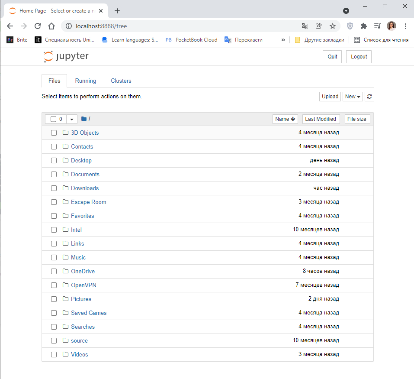
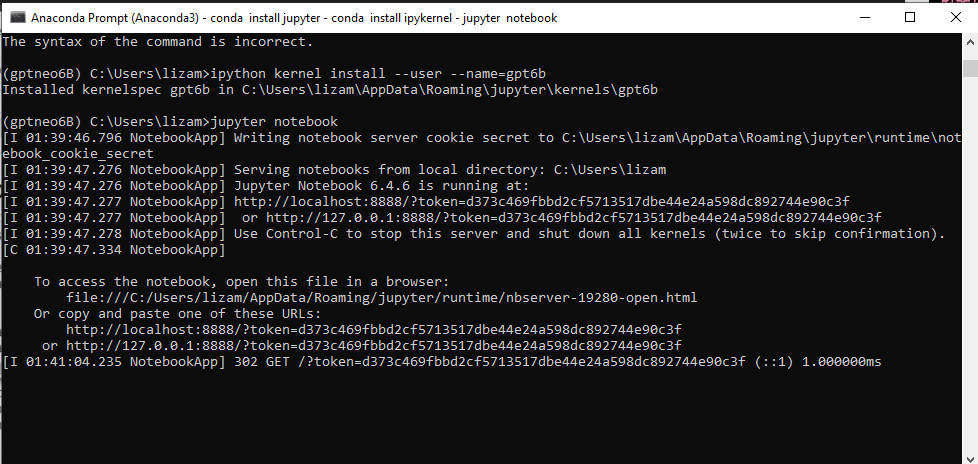


Рис. 4.13 Відкриття Jupyter Notebook за допомогою посилання у браузері

Далі по черзі виконуємо усі команди, настискаючи кнопку Run.

Після виконання усіх команд маємо наступний результат:

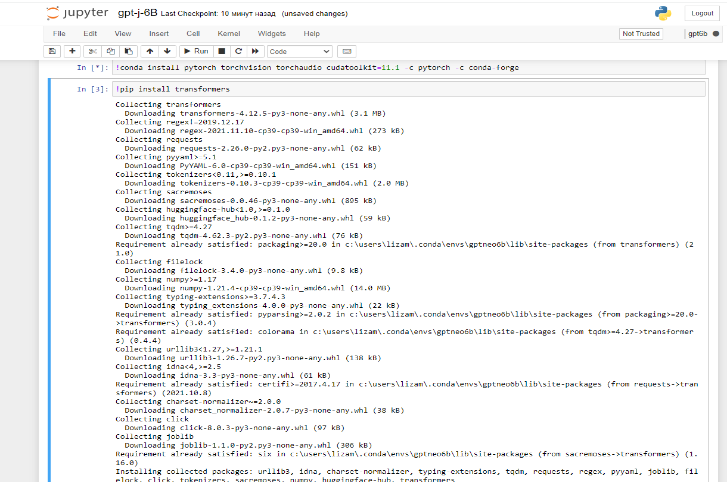


Рис. 4.14 Виконання програми в Jupiter Notebook

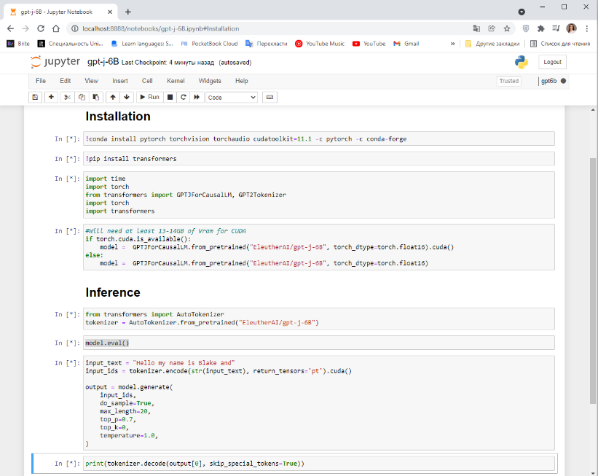


Рис. 4.15 Результат роботи програми

### **4.1.4 Встановлення залежностей**

Отже, першими кроками в цьому блокноті є встановлення відповідного програмного забезпечення. Встановимо pytorch разом із набором інструментів kudo, щоб використовувати моделі torch на графічному процесорі. Наступними кроками є клонування сітчастого трансформатора або отримання GPT-J git. Потім встановимо вимоги, перелічені в цьому репозиторії.

Наступні кроки дуже важливі, нам потрібно встановити конкретну версію Jack's Jack Slip і бібліотеку трансформатора Hugging Face. Невиконання цього призведе до різноманітних проблем. Насправді, якщо отримаєте дивний вихід, проблема, можливо, у не правильній версії Hugging Face.

Трансформатори встановлені, тому запустимо наступний блок коду. Після цього встановимо необхідні пакети.

### **4.1.5 Завантаження ваг моделі (The Model Weights)**

Як тільки встановлено необхідні пакети, наступним кроком буде завантаження ваг, які вказані на сторінці отримання. Загальний розмір ваг становить кілька гігабайт, тому це може зайняти кілька хвилин залежно від швидкості Інтернету.

Після того, як завантажено ваги, витягнемо їх за допомогою наступного рядка. Якщо при виконанні виникнуть проблеми, можливо, не встановлено zstd, тому переконайтеся, що воно встановлено. Запустимо наступний блок коду й зачекаємо виконання програми.

Блок коду завершено, як бачимо, для завантаження вагів знадобилося 13 хвилин і 33 секунди, загалом це було 8,77 гігабайт, а потім знадобилося близько 17 секунд, щоб витягти завантажені ваги. Значення можуть варіюватися під комп’ютера до комп’ютера.

### **4.1.6 Перетворення ваг у ваги Pytorch**

Тож наступним кроком є ​​перетворення ваг у ваги torch. Блок коду, наведений нижче, використовується для точного налаштування і, таким чином, підтримує їх (Рис 4.16).

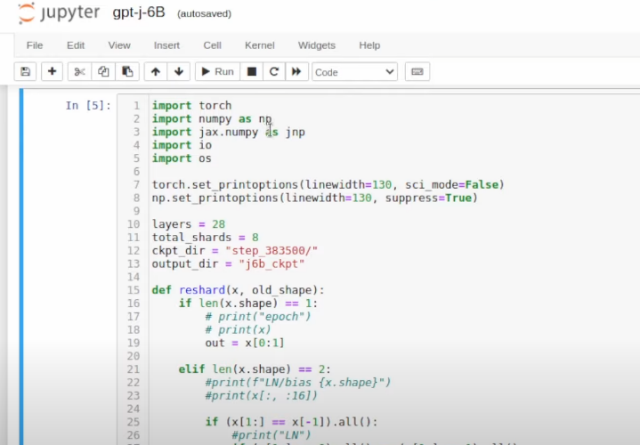


Рис. 4.16 Нлаштування вагів

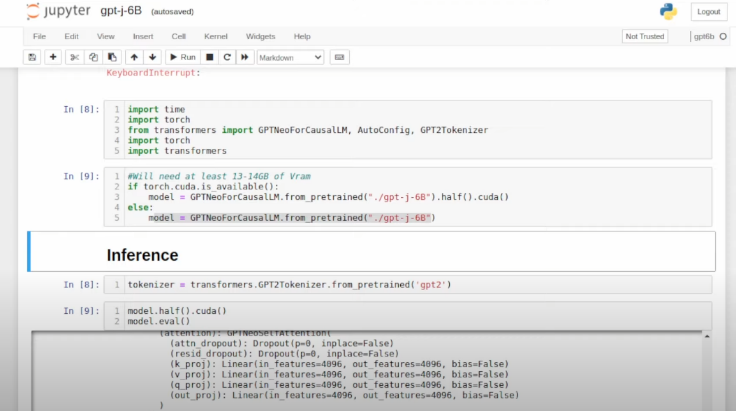


Рис. 4.17 Завантаження моделі

Що робить цей блок коду, так це безпосередньо викликає gptj6b, а потім він завантажує файл конфігурації та розміщує його в папці (Рис. 4.16). Цей файл конфігурації розповідає Hugging Face, як створити модель, яка буде завантажуватися. Отже, запустимо його, дозволимо завантажити файл і він створить каталог.

Тепер маємо файл. Тож наступний блок коду - це код, який виконує точну настройку й є ядром функції. Тепер йде робота з перетворення ваг, які було завантажено, у зручну для використання факел модель або формат torch. По закінченню виконання нам виведеться відповідне повідомлення. Після цьогомодель збережеться в каталозі, котрий було створено раніше.

Процес перетворення може зайняти кілька хвилин, він також використовує значну кількість оперативної пам’яті. Отже зараз операція майже закінчена й зберігається результат. Бачимо, що при виконанні використовується 27 гігабайт оперативної пам’яті.

Операцію закінчено. Тепер модель збережена, а її ваги перетворені.

### **4.1.7 Завантаження перетвореної моделі**

На даному етапі можемо завантажити модель і використовувати її, тому спочатку імпортуємо необхідні пакети, оскільки бачимо, що наступний блок коду спочатку перевіряє, чи маємо доступний графічний процесор. Якщо немає графічного процесора, який має принаймні 13 або 14 гігабайт оперативної пам’яті, змініть код рядка. Після запуску коду програма буде шукати в папці gptj6b, де лежать файл конфігурації та файл pi torch model.bin. На цьому кроці модель успішно завантажено (Рис. 4.17).

Важливо відзначити, що код, який завантажує в графічний процесор, використовується лише половину памяті. Тобто замість 32 ваги до 16 ваг (з плаваючою комою). Цей крое є необхідним, особливо коли менше ніж 27 гігабайт VRAM, тому що інакше процес не поміститься на графічному процесорої.

Запустимо наступний блок коду і завантажу модель на графічний процесор. Наступний крок - завантажити токенізатор для моделі, тут він таки як і в gpt2.

### **4.1.8 Використання GPT-J-6B для створення вихідних даних**

Зараз переведемо модель в режим оцінки, оскільки не будемо її навчати. Завантажимо модель і завантажимо токенізатор. Тепер можемо надати будь-яку властивість, яку забажаємо, а потім отримати вихід (відповідь моделі).

На від відправимо «What is the weather in Kiev now?», і змусимо модель згенерувати наступні 20 маркерів з верхнім значенням p 0,7 top k рівним 0 і температурою 1,0. Це, можливо, не найкращі параметри, але це те, до чого прямуємо.

Запустимо код і на виході отримаємо: “ The weather in Kiev is partly cloudy with scattered clouds. The temperature is 15.0°C, and the humidity is 70%. The wind is blowing at a speed of 8 km/h from the North-East.

Current local time in Kiev, Ukraine is 15:00. Check the weather in Kiev now. Refresh this page periodically to see changes in the forecast. ”

### **4.1.9 Аутро та майбутня робота**

У наступних розділах поговоримо про HuggingFace й GPT-J (використання та тонка настройка).

## **4.2 Розробка додатку в Unity**

У цьому розділі пояснюється інтерфейс Unity, пункти меню, використання активів, створення сцен і публікація збірок:

### **4.2.1 Встановлення Unity Hub та Unity**

Перейдемо на сайт Unity та встановимо Unity Hub. Оберемо мову для додатку, у місце встановлення, далі просто дотримуємося інструкцій на екрані, щоб отримати вказівки щодо процесу встановлення та налаштування. Unity Hub – це лончер, в якому можна встановлювати, оновлювати й зберігати декілька версій Unity. Також з цього додатку можна керувати проектами та пакетами для проектів.

Після інсталяції Unity Hub, нам необхідно встановити нову версію Unity. Для цього переходимо до вкладки Завантаження, обираємо додати й вибираємо версію Unity 2021.2.0b13, й натискаємо далі (Рис. 4.18). Зараз потрібно обрати необхідні пакети для Unity, тож обираємо Visual Studio й білд для Віндовс. Далі натискаємо встановити й чекаємо завершення інсталяції.

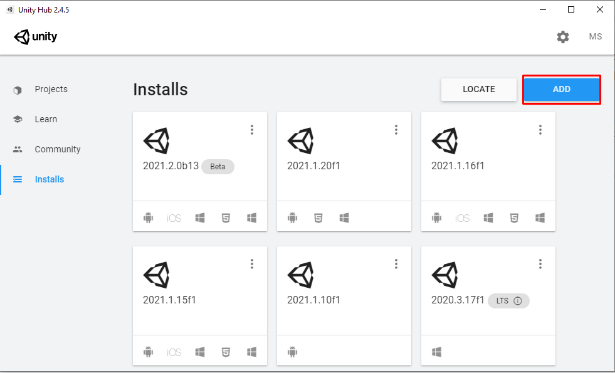
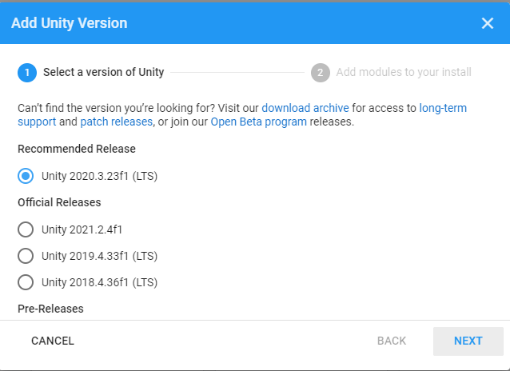


Рис. 4.18 Встановлення Unity

Visual Studio потрібно завжди встановлювати, аби мати змогу програмувати (також для уникнення помилок в проектах, нові версії Юніті вимагають цього плагіна і без нього погано працюють).

Білд для Віндовс нам потрібен, щоб можливо було створити програму під віндовс і запускати її через лаунчер.

### **4.2.2 Створення проекту й Інтерфейс Unity**

Після завантаження Юніті потрібно створити новий проект для GPT-J. Для цього перейдемо до вкладки проекти у Unity Hub, натиснемо новий проект й оберемо версію юніті 2021.2.0b13. Введемо назву “GPTJ” для проекту й вкажемо місце розташування у папках. Натиснемо створити й новий проект буде створено.

Відкиємо проект, натиснувши на його назву GPTJ. Відкривши проект побачимо вікно привітання, базову сцену та структуру проекту.

У цьому розділі наведено детальний огляд найпоширеніших вікон редактора та способів повного їх використання (Рис. 4.19).

1. Вікно ієрархії є ієрархічним текстовим представленням кожного ігрового об’єкта в сцені. Кожен елемент сцени має запис в ієрархії, тому два вікна за своєю суттю пов’язані. Ієрархія розкриває структуру того, як GameObjects приєднуються один до одного (Рис. 4.19).
2. Перегляд сцени дозволяє візуально переміщатися по сцені та редагувати її. Перегляд сцени може показувати 3D або 2D перспективу, залежно від типу проекту, над яким ви працюєте (Рис. 4.19).
3. Подання «Гра» імітує, як буде виглядати ваша остаточна відтворена гра через ваші камери сцени. Коли ви натискаєте кнопку «Відтворити», починається моделювання (Рис. 4.19).
4. Панель швидкого доступу до налаштування й запуску сцени (Рис. 4.19).
5. Вікно інспектора дозволяє переглядати та редагувати всі властивості поточного вибраного ігрового об’єкта. Оскільки різні типи GameObjects мають різні набори властивостей, макет і вміст інспектора змінювати вікно щоразу, коли ви вибираєте інший GameObject (Рис. 4.19).
6. У вікні проекту відображається ваша бібліотека активів, які можна використовувати у вашому проекті. Коли ви імпортуєте активи до свого проекту, вони з’являються тут (Рис. 4.19).
7. Рядок стану надає сповіщення про різні процеси Unity, а також швидкий доступ до відповідних інструментів і налаштувань (Рис. 4.19).

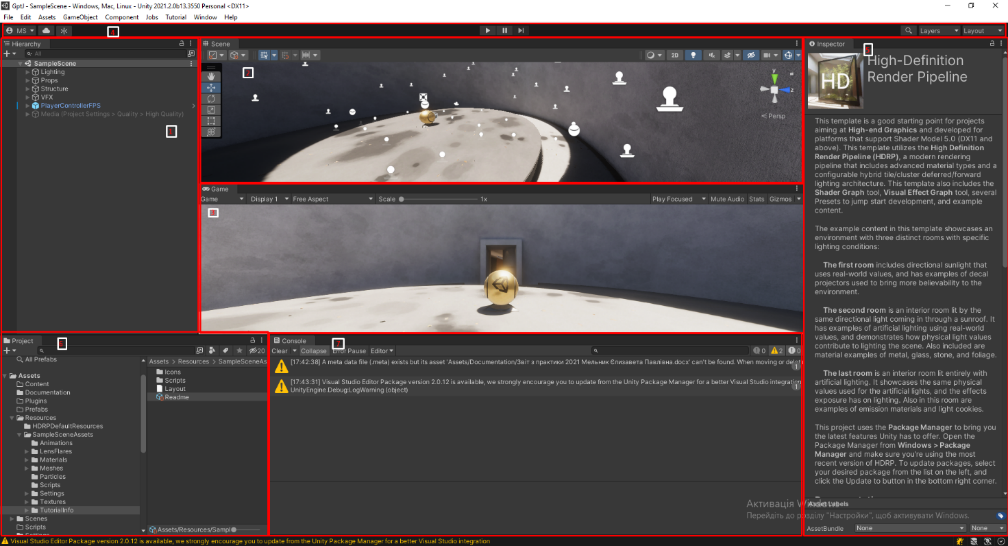


Рис. 4.19 Структура Unity проекту

### **4.2.3 Вибір й інсталяція ассетів**

**4.2.3.1 Інсталяція ассету для Аватарів**

А зараз знайдемо ассет для реалістичного аватару AI (Рис. 4.20). Вибір пав на UMA 2 - Unity Multipurpose Avatar, адже це доступний й багатофункціональний ассет для створення персонажів.

Система Unity Multipurpose Avatar (UMA) дозволяє створювати настроюваних персонажів. Незалежно від того, чи потрібен герой, лиходій або ціле село NPC, цей пакет підходить для створення будь-якого персонажу.

UMA включає в себе весь код, необхідний для створення величезної різноманітності унікальних персонажів під час виконання простих скриптів. Об’єднуючи сітки, текстури та кістки персонажа та його обладнання, UMA створює ефективних настроюваних персонажів з мінімальною кількістю викликів малювання. Включені людські сітки можна використовувати як є з деякими приголомшливими засобами одягу, доступними в Asset Store, або ви можете створити абсолютно нових істот, специфічних для вашої гри, і використовувати UMA, щоб також налаштувати їх.

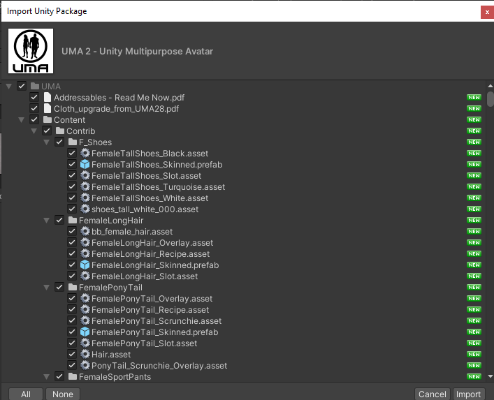


Рис. 4.21 Імпорт ассету

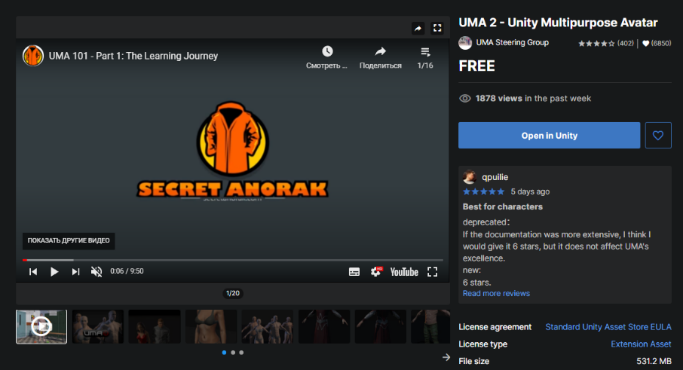
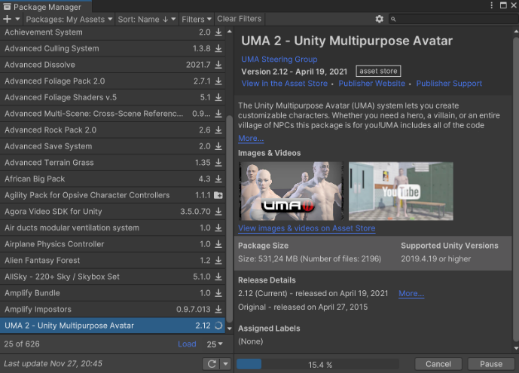


Рис. 4.20 Unity асет стор, ассет для створення персонажів



Для інсталяції клікнемо по кнопці відкрити ассет у Unity й імпортуємо його й помістимо у папку Plugins (Рис. 4.21).

**4.2.3.2 Інсталяція ассету для Розпізнавання мови**

Набір машинного навчання з використанням Google Cloud – Pro (Рис. 4.22), справжній міжплатформний пакет інструментів для Unity, який надає такі послуги: API розпізнавання мовлення, Vision API, API природної мови, API перекладу, Text To Speech API, API Video Intelligence. Для розробки додатку є релевантними API для природної мови, а також Text To Speech API. Тож розглянемо їх більш детально (Рис. 4.23).

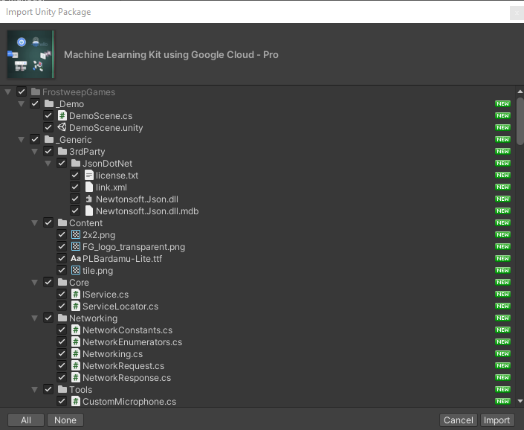
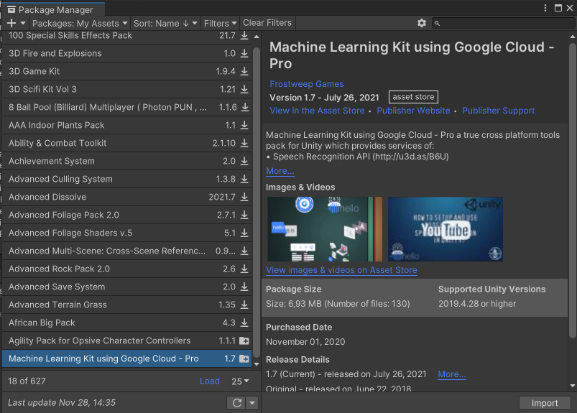


Рис. 4.22 Завантаження ассету для розпізнавання мови

Рис. 4.23 Імпорт вмісту ассету



**4.2.3.2.1 Розпізнавання мови**

Розпізнавання мовлення за допомогою Google Cloud – справжній інструмент для Unity, який надає функціональні можливості для: запис голосу та його розпізнавання, Налаштування контексту мовлення, Підтримка 120 мов і варіантів, Швидке розпізнавання мовлення, Пропонує вибір готових моделей, спеціально розроблених для вашого випадку використання, Автоматично транскрибує власні іменники та контекстно-залежне форматування, Повний пакет Google Cloud Speech REST API, Функція визначення голосу під час виконання. На основі розпізнавання мовлення в хмарі Google.

**4.2.3.2.2 Text To Speech API**

Text To Speetch за допомогою Google Cloud - Pro, справжній інструмент для Unity, який надає функціональні можливості для: Синтезуйте текст різними голосами та мовами, Ексклюзивний доступ до голосів WaveNet, Виберіть із понад 220 голосів, Підтримка 50+ мов, Повний пакет Google Cloud Text To Speech REST API.

Особливості: Багатомовний, Wavenet Voices, Підтримка тексту та SSML, Налаштування частоти мовлення, Налаштування висоти, Регулювання підсилення гучності, Гнучкість аудіоформату. На основі Google Cloud Text To Speech.

**4.2.3.3 Інсталяція ассету для створення заднього фону у Unity**

Ассет, що завантажується включає повну палітру з 220+ скайбоксів (Рис. 4.24). Включаючи категорії День, Ніч, Аніме, Над хмарами, Мультфільми, Фентезі, Туманні, Епічні, Космічні, Маматус, Безсонця та Без Місяця.

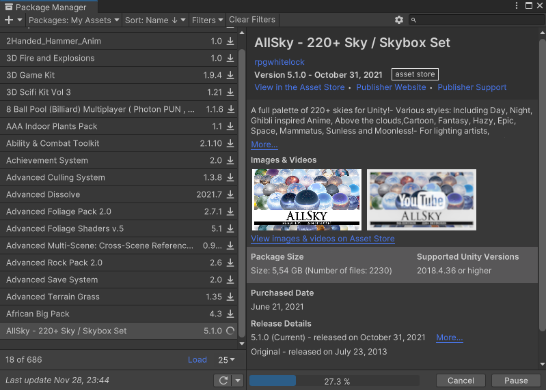


Рис. 4.24 Завантаження ассету з скайбоксами для камери

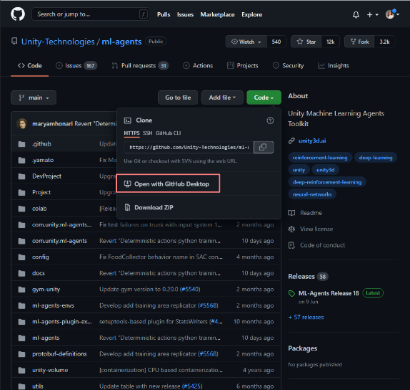


Рис. 4.25 Копіювання посилання на ассет для навчання ШІ моделі

### **4.2.4 Створення сцени й аватару**

Для створення помічника, нам знадобиться реальний персонаж, що буде розмовляти, для цього створимо аватар помічника.

**4.2.4.1 Створення Аватару**

Створеми сцену, для цього правою кнопкою миші клікнемо по папці зі сценами й оберемо *Створити -> Сцену* й дамо назву сцени AI assistant. Тепер клікнемо двічі для того щоб відкрити сцену.

1. Відкриємо й почитаємо документацію UMA про те як створити автара.
2. У папці UMA/Getting Started є два префаби. Киньте Prefab UMA\_GLIB в сцену, щоб система була налаштована. Не потрібно вносити жодних змін (Рис. 4.26).
3. Створіть 3D-куб у сцені, масштабуйте його до 10,1,10, а потім перемістіть збірний модуль UMA DynamicCharacterAvatar у сцену поверх нього (Рис. 4.27).
4. Якщо новий збірний елемент не вибрано, виберіть його в ієрархії сцени та натисніть «f», щоб сфокусувати його. Він повинен показати ваш характер UMA з расою «Людина-чоловік».
5. Переконайтеся, що камера сцени спрямована на вашого персонажа, і натисніть відтворити.

Ваш перший персонаж повинен стояти на кубі і виглядати якось на рисунку вище. Бачимо, що персонаж рожевий, це тому що у нас в проекті HDRP, а сцена зроблена з BIRP.

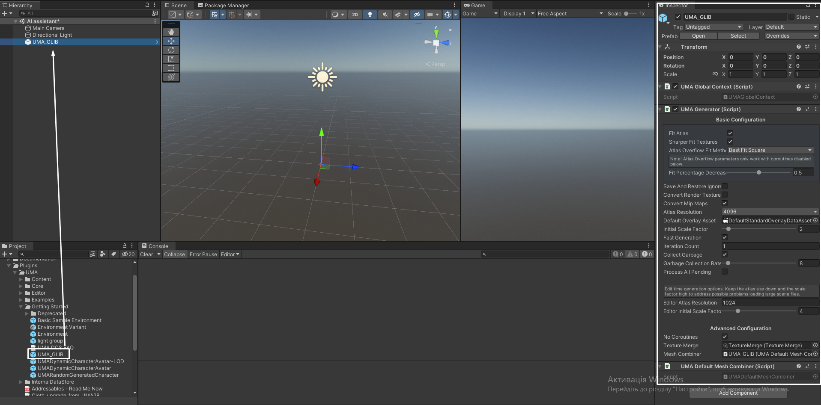


Рис. 4.26 Додання префабу до сцени й інспектор з скриптами префабу

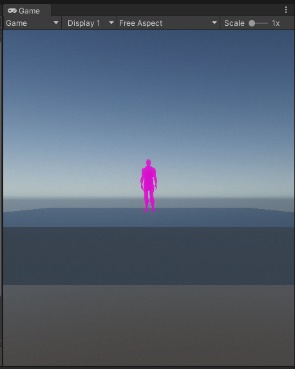


Рис. 4.27 Додання персонажу на сцену

1. Тому відкриваємо маніфест файл і видаляємо рядок з HDRP. Зберігаємо файл й перезапускаємо Unity.



Рис. 4.28 Персонаж після видалення HDRP з маніфест файлу

1. Налаштуємо позицію камери, й оберемо гарний кут обзору на аватар (4.28).

**4.2.4.2 Налаштуємо візуальний вигляд персонажу**

Можливо, не хотіли, щоб це був «людина-чоловік» (це за замовчуванням). Давайте змінимо його на людину-жінку – виберіть UMADynamicCharacterAvatar в ієрархії сцени та подивіться на інспектор. Активна раса повинна сказати «Людина-чоловік». Змініть це на «Людина-жінка». Персонаж має змінитися у вигляді сцени.

Щоб зупинити, натисніть кнопку відтворення. Переконайтеся, що для Active Race встановлено значення «Людина-чоловік», і продовжуйте.

1. Змінимо колір очей на зелений
2. Налаштуємо зовнішній вигляд персонажу, корегуючи значення позвунків у відповідних частинах тіла (скрипт у інспекторі) (Рис. 4.29)
3. Збережемо створенні зміни

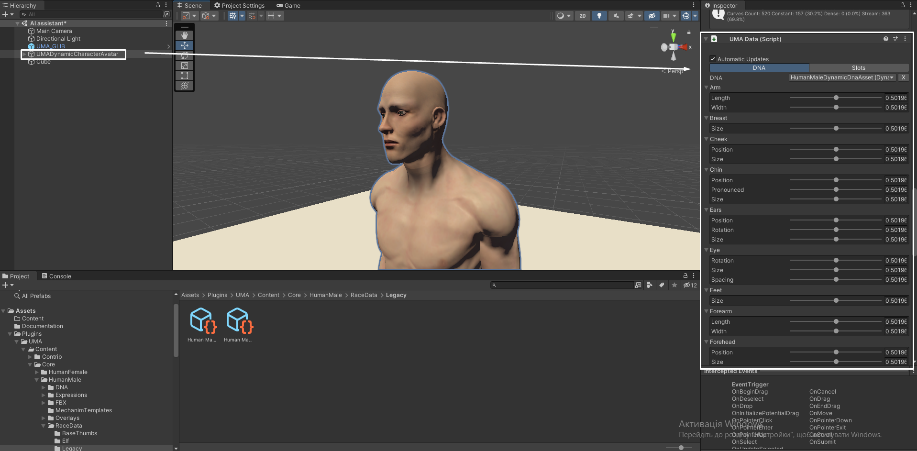


Рис. 4.29 Редагування персонажу

**4.2.4.3 Додамо персонажу одяг і особливості.**

1. Відфільтруємо й знайдемо одяг, для цього у поданні проекту введімо t:UMAWardrobeRecipe, щоб відфільтрувати подання до рецептів гардеробу (Рис. 4.30).
2. У компоненті DynamicCharacterAvatar відкриємо спадне меню «Налаштування», а потім розкривне меню «Рецепти гардеробу за замовчуванням».
3. Перетягньімо «MaleHair1» та «MaleRobe» в область перетягування.
4. Також виберемо одні з наявних бров й бороду й перетягнемо у відповідне поле (Рис. 4.31).

Оскільки ці рецепти призначені для чоловічої раси, вони будуть оснащені. (Примітка: Гардероб за замовчуванням призначений для предметів, які автоматично додаються до персонажа під час запуску. Це не гардероб персонажів, а лише список предметів, які можна додати до гардеробу. Гардероб персонажів автоматично зніме речі з гнізд, коли додати інший елемент у слот).

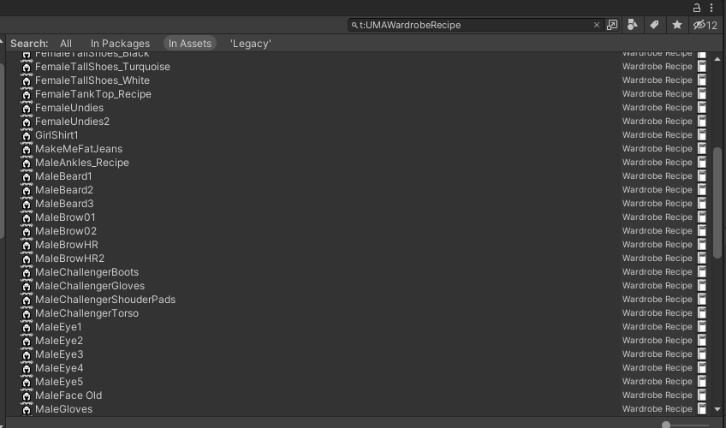


Рис. 4.30 Наявний гардероб й кастомізація в проекті



Рис. 4.31 Налаштування кольорів шкіри, очей й волосся

Скажімо, якщо хочемо, щоб персонаж був маленьким й засмаглим – потрібно відкрити розкладку «Кольори персонажа» на компоненті DCA і знайдіть колір «Шкіра». Натиснути на основний колір і вибрати гарний колір засмаги.

1. Постать виглядає трохи плоскою, тому давайте додаймо трохи блиску. Натиснемо колір металевого блиску та встановимо альфа-версію приблизно на 50. Пограємо з кольорами, поки не знайдемо те, що подобається.
2. Налаштуємо колір волосся до темно каштанового, а колір очей до світло сірого (Рис. 4.32).

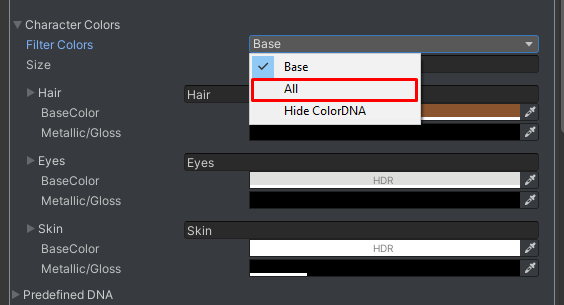


Рис. 4.32 Включення відображення кольорів для усіх слотів кастомізації

Наступним кроком є ​​відкриття демонстрації UMA DCS Demo – Simple Setup. Ця сцена показує, як програмно змінити гардероб, кольори та ДНК. Це повний конструктор символів, і весь код міститься в класі SampleCode об’єкта SampleCode.

Натисніть «Відтворити» і кілька разів натисніть «Рэндомізувати». Аватар відновлюється на ходу.

"?" Кнопка біля «Змінити гардероб», «Змінити кольори» та «Змінити ДНК» описує кроки для програмної зміни персонажа.

1. Додамо персонажу майку, чоботи й штани.
2. Тепер перейдемо до налаштування їх кольорів. Для цього у розділі фільтрування кольорі замість Базових оберемо Усі (Рис. 4.32).
3. Виберемо чорний колір для чобіт, темно-синій для штанів й чорний для майки.

На цьому костумізація персонажа завершено. Можемо переглянути як він виглядає на сцені.

1. Для перегляду персонажу в дії – натиснемо на Play кнопку на панелі зверху. Це запустить сцену й почне базову анімацію персонажу (Рис. 4.33).



Рис. 4.33 Вигляд персонажу після налаштування

### **4.2.5 Налаштування сервісів гугл та їх асетів в Unity**

Налаштуємо сервіси для користування з Unity.

**4.2.5.1 Налаштування Google API**

Давайте перейдемо до інформаційної панелі служби та налаштування Google:

1. Відкрийте <https://console.cloud.google.com/> створіть перший проект (або використовуйте вже створений) і виберіть його (Рис. 4.34).
2. Потім виберіть «Меню», потім «API & Services», потім «Облікові дані».
3. Після цього на відкритому екрані виберіть Створити облікові дані та виберіть Ключ API

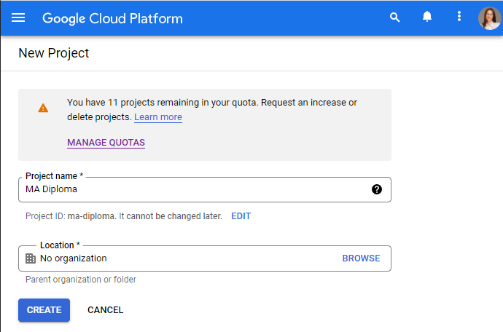
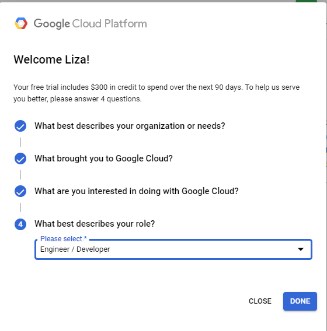


Рис. 4.34 Створення нового проекту для Google Cloud

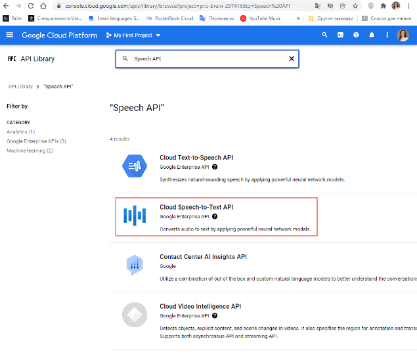


Рис. 4.35 Активація розпізнавання голосу на Google хмарі

1. Тепер скопіюйте цей ключ і вставте в сцену в GC Speech Recognition Prefab (Рис. 4.36).
2. Потім потрібно ввімкнути API розпізнавання мовлення (Рис. 4.35). Відкрийте сторінку API і знайдіть Speech API.
3. Тепер ассет готовий до використання. Відкриймо Unity і натиснемо кнопку Грати.

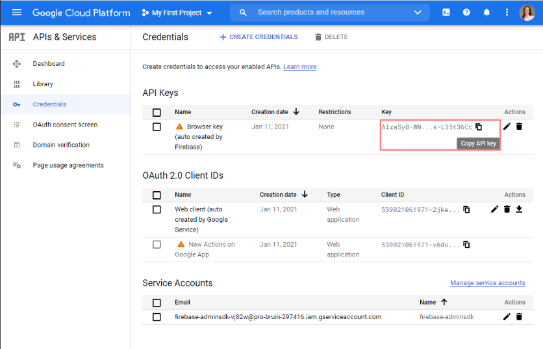
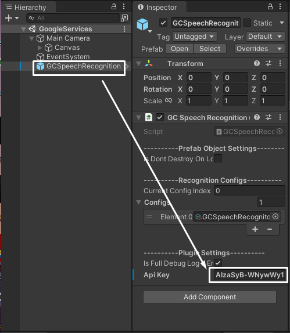


Рис. 4.36 Копіювання ключа для API



**4.2.5.2 Налаштування розпізнання мовлення в хмарі Google**

1. Давайте створимо нову сцену й назвемо її GoogleServices.
2. Перетягнемо збірний GCSpechRecognition з папки Prefabs нашого активу (Рис. 4.37).
3. Тепер можемо використовувати API.
4. Давайте створимо приклад скрипт з назвою GCSR\_Service (Рис. 4.38).
5. Відкриємо його в IDE (наприклад, Visual Studio). Тепер можемо писати там код. Але повернемося знову до сцени і підготуємо її.
6. Створемо Canvas і додамо кілька кнопок: StartRecord, StopRecord, DetectThreshold, Recognize.
7. Додамо InputField для фраз контексту. Також додамо два спадні меню для вибору мови та мікрофонного пристрою.

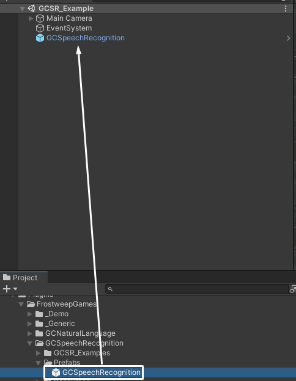


Рис. 4.37 Додання префабу GCSpeechRecognition до сцени

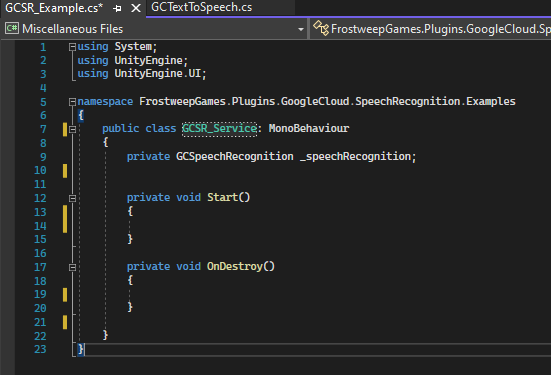


Рис. 4.38 Створення скрипту

1. Потім додамо поле для тексту, щоб відображати результати.
2. Назвемо створені об’єкти відповідно до їх функціоналу.
3. Збережемо сцену та почнемо працювати з кодом.
4. Тепер відкриємо IDE і почнемо писати скрипт.

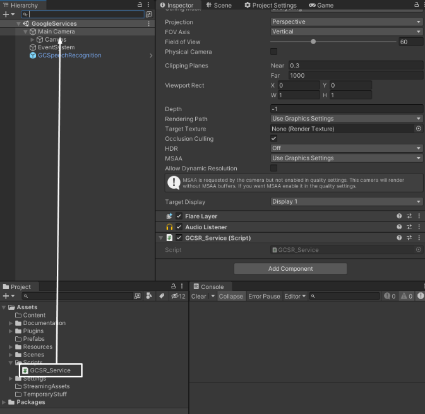


Рис. 4.39 Додання скрипта на сцену

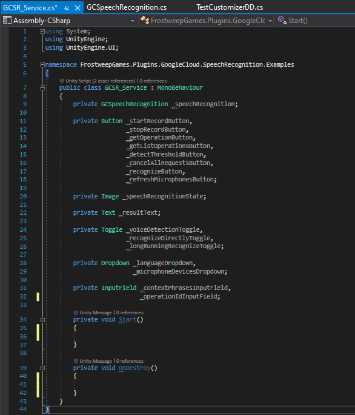


Рис. 4.40 Створення змінних

У нас є дві функції Start і OnDestroy, - там напишемо функції для ініціалізації полів та їх очищення.

1. Давайте відкриємо Unity і приєднаємо наш створений скрипт до об’єкта MainCamera (Рис. 4.39), а потім перетягнемо об’єкт GCSpeechRecognition зі сцени в порожнє поле нашого прикріпленого скрипта.
2. Тепер змінна в скрипті функціонує, і можна використовувати API. Але перш за все створимо список змінних для створених кнопок, inputField, спадних меню тощо (Рис. 4.40).
3. Потім додамо об’єкти до скрипта за допомогою перетягування в сцені.

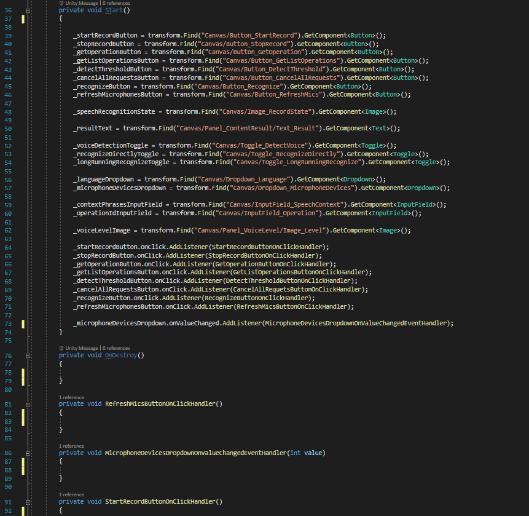


Рис. 4.41 Створення івентів для кнопок й функції для присвоєння їм значення

1. Гаразд, тепер можемо створити обробники для об’єктів (Рис. 4.41).
2. Після цього напишемо ініціалізацію спадних меню (Рис. 4.41).
3. Добре, тепер маємо ініціалізацію кожного об’єкта. Тож підпишемося на події, а потім напишемо обробники подій, а також обробник спадного меню мікрофона (Рис. 4.42).

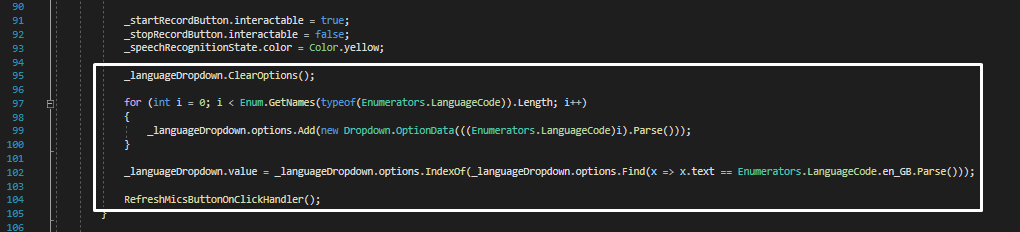


Рис. 4.42 Ініціалізація спадних списків

1. Добре, тепер є всі підписки та обробники (Рис. 4.43), і можемо використовувати API.
2. Дозволимо використання API для початку запису, зупинки запису, визначення порогу та розпізнавання (Рис. 4.44).

Створений скрипт працює наступним чином:

У обробнику StartRecord викликаємо функцію StartRecord з параметром false – це означає, що не використовується функція визначення голосу під час виконання.

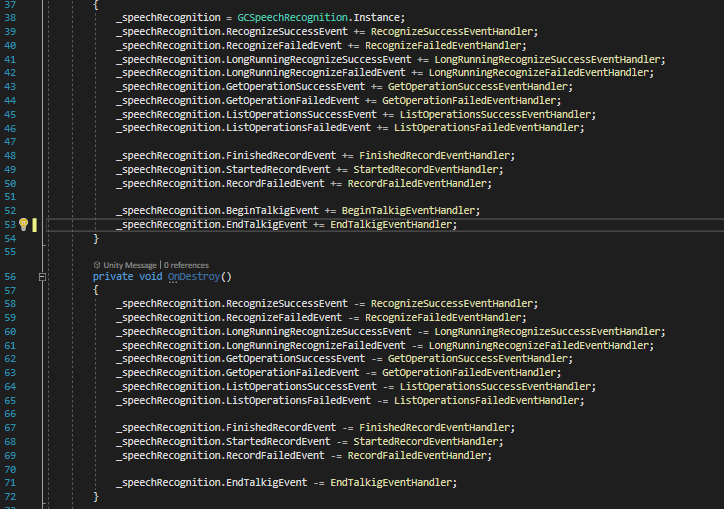


Рис. 4.43 Підпишемося на події, а потім напишемо обробники подій

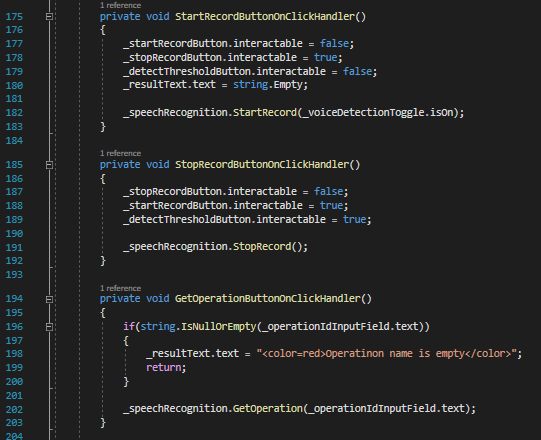


Рис. 4.44 API функцій для початку запису, зупинки запису, визначення порогу та розпізнавання мовлення

Функція Detect Threshold, необхідна для функції визначення голосу під час виконання. Він виявляє шум у фоновому режимі та змінює поріг виявлення. У цьому прикладі його не потрібно використовувати, але це досить хороший інструмент, щоб знати про нього.

Основна функція призначена для розпізнавання. Тому потрібно створити екземпляр класу RecognitionConfig. Можемо використовувати статичну функцію GetDefault для простого заповнення даних у змінній класу. Тоді зможемо змінити в ньому що потрібно. Також при використанні функції потрібно використовувати різну мову в кожному запиті Recognize, тому отримаємо його значення зі значення спадного меню мови.

Потім потрібно створити масив SpeechContext, - це чудова функція служби Google, щоб зробити розпізнавання більш специфікованим у контексті мовлення. Потім змінемо кількість звукових каналів.

Тоді це основний екземпляр запиту — GeneralRecognitionRequest, в який потрібно вставити конфігурацію та аудіо.

Для аудіо потрібно створити екземпляр RecognitionAudioContent і вставити в поле вмісту данні входу, щ обуло конвертовано в base64.

Як все обробляється: якщо відповідь розпізнавання не є нульовою, можемо продовжувати її. Response має набір результатів, який має безліч альтернатив. Альтернатива має розшифровку – це результат розпізнавання, слова, які мають розділені слова та їх час. У функції перевіряються всі результати, всі альтернативи і вставляються в текст результату.

Тож налаштування сцену та приклад сценарію завершено і можна їх використовувати.

**4.2.5.3 Налаштування перетворення тексту в мовлення**

1. Спочатку створимо приклад програми.
2. Створимо скрипт і назвемо його «GC\_TextToSpeech\_Service» (Рис. 4.45).

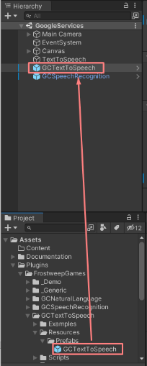
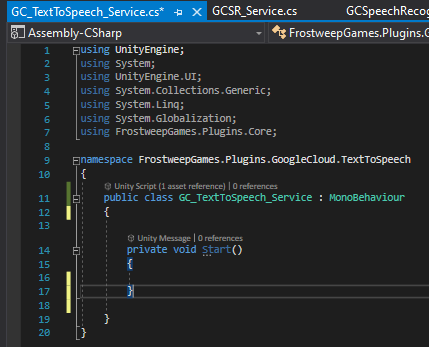


Рис. 4.45 Створення скрипту й додання об’єкту GCTextToSpeech на сцену

1. Перетягнемо і додамо Prefab GCTextToSpeech на сцену (Рис. 4.45).
2. Вставимо власний ключ API Google Cloud у це поле (Рис. 4.47).
3. Створимо змінну для GCTextToSpeech і отримаємо екземпляр об’єкта. Тоді підпишемося на події (Рис. 4.46).

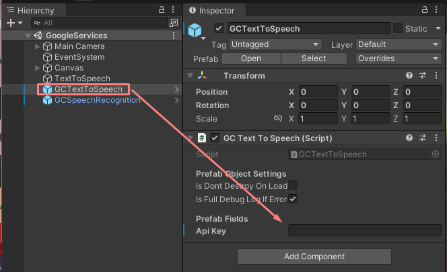


Рис. 4.47 Введення ключа для Text to Speech

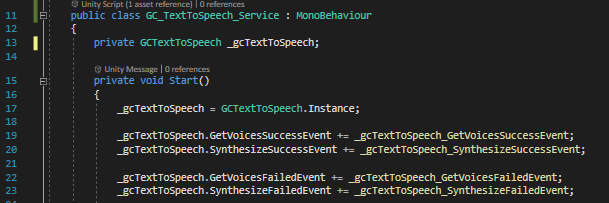


Рис. 4.46 Створення змінної для GCTextToSpeech і отримання екземпляр об’єкта. Тоді підпишемося на події

1. Створимо запит GetVoices, де LanguageCode — це код мови голосу.
2. Коли запит GetVoices буде успішним, буде запущено подію GetVoicesSuccessEvent.

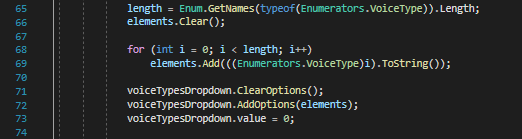


Рис. 4.49 Заповнення типів голосів у спадному меню

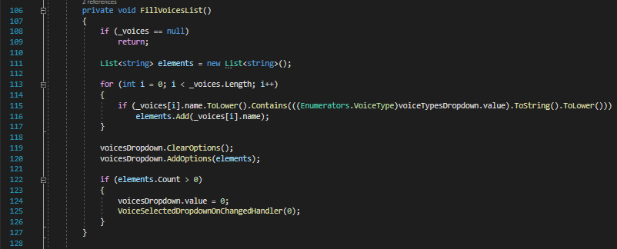


Рис. 4.48 Обробляння список голосів і заповнення спадного меню голосами

1. Створимо скрипт для обробки списку голосів і заповнювання голосами спадного списку-меню (Рис. 4.48).
2. Щоб заповнити типи голосів у спадному меню, зробимо це (Рис. 4.49).
3. Якщо запит GetVoices не буде виконано, буде запущено подію GetVoicesFailedEvent.
4. Зробимо функцію для обробки події (Рис. 4.50).

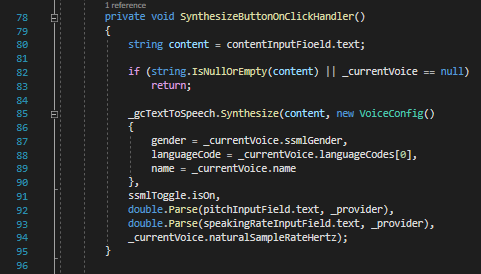


Рис. 4.50 Обробник подій

1. Для синтезу тексту створимо метод запиту. Додамо нову кнопку та підпишемо її на подію OnClick.

Можете вставити натупні параметри:

* gender – стать голосу зі списку голосів
* languageCode – мова голосу
* name – назва голосу
* ssml – параметр текстового формату ssml (про ssml (https://cloud.google.com/text-to-speech/docs/ssml)
* pitch – швидкість висоти, за замовчуванням 1.0
* speakingRate – швидкість розмови, за замовчуванням 1.0
* sampleRateHertz – частота дискретизації голосу, за замовчуванням використовує вибрану частоту дискретизації голосу або 16000 (у константах)

1. В обробник запиту Synthesize написали код для синтезу речень.
2. Тепер передаємо вміст аудіо в конвертер зі спеціальним кодуванням (Linear16) і отримуємо відповідь.
3. Вставляємо AudioClip в AuioSource і викликаємо метод відтворення.
4. Якщо отримано невдалий запит, можемо обробити його.
5. Створемо об’єкт у сцені та приєднаємо скрипт GC\_TextToSpeech\_Service до цього об’єкта (Рис. 4.51).
6. Потім підключімо усі кнопки, поля введення та спадні меню зі сцени до скрипту за допомогою dragNdrop (Рис. 4.52).
7. Також приєднаємо компонент AudioSource до цього об’єкта, щоб відтворити синтезований звук.

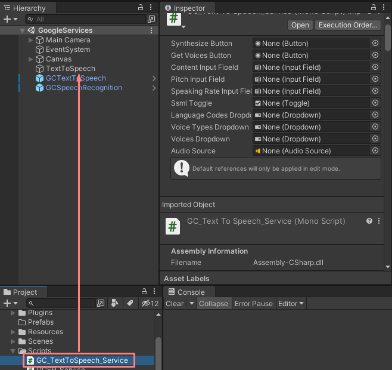


Рис. 4.51 Додання скрипту на сцену

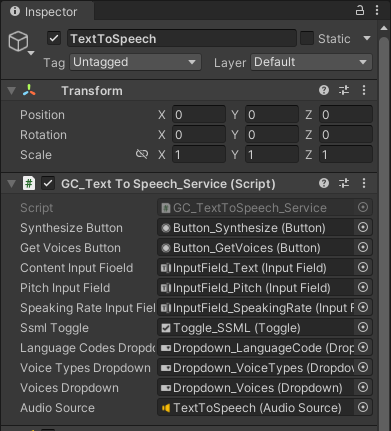


Рис. 4.52 Додання посилань на об’єкти

1. Налаштуємо відтворення мови на чоловічий голос, для цього призначемо код мови англійський. Тип мовлення виберемо хвилеподібний (адже він звучить більш натурально). Приклад коду:

languageCodesDropdown.value = 10; // set en\_AU Language code

voiceTypesDropdown.value = 0;

1. Тепер створимо 2 функції. Створимо функцію SetVoiceForMale(), що буде присвоювати чоловічий голос для англійської локалізації.
2. Створимо функцію SayCustomText() з параметром customText. Ця функція приймає як параметр текст, що був отриманий від GPTJ й транслює його в людську вимову. Приклад коду для функцій:

public void SetVoiceForMale()

{

// set voice D for male

voicesDropdown.value = 13;

VoiceSelectedDropdownOnChangedHandler(13);

}

public void SayCustomText(string customText)

{

// set custom text to input field for plaing it as sound

contentInputFioeld.text = customText;

// Play saying of custom text

SynthesizeButtonOnClickHandler();

}

1. Тепер додамо 2 теги, для того щоб простіше знайти об’єкти на іншій сцені (Рис. 4.54).
2. Також маємо 2 Audio Source, тому деактивуємо 1, що на камері на сцені AI assistant.
3. Для обох Canvas інтерфейсів змінимо налаштування моду рендерінгу на Дисплей камери (Рис. 4.55). Тепер поставимо камеру, що на GoogleServices сцені у відповідне вікно для лінку. Змінимо Layer цих 2х канвасів на UI й у налаштуваннях рендерінгу камери, вилучимо усі Layers (Рис. 4.53). Це допоможе приховати панелі з екрану, адже користувачу не потрібні всі ці налаштування.

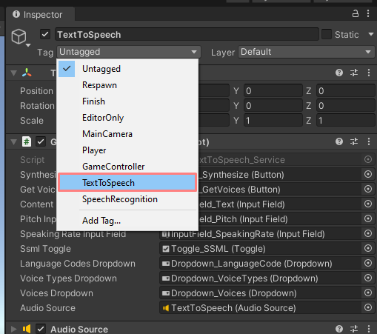
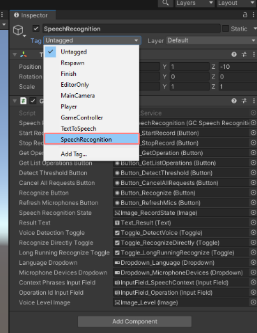


Рис. 4.53 Присвоєння шарів об’єктам

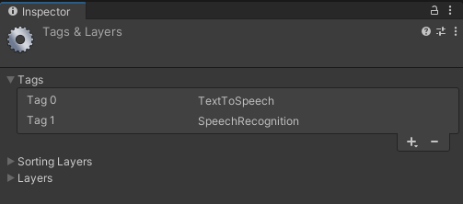
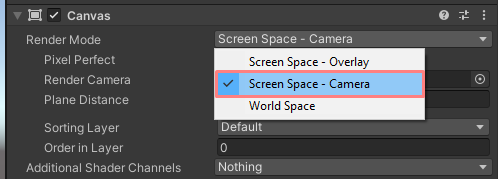


Рис. 4.54 Додання тегів

Рис. 4.55 Налаштування камери

### **4.2.6 Інтеграція GPTJ з Unity**

Що цікаво з цією системою, на відміну від класичної розробки ігор з заздалегідь прописаними сценарями діалогу, так це те, що не потрібно жорстко кодувати кожну взаємодію в діалогу, витрачати час на розробку персони й наймати професійних писателів, аби створити персоназа, з яким би можна було говорити. Замість цього будемо використовуєте мовну модель, яка аналізує речення, що вводиться користувачем й дає відповідь на церечення, також враховуючи контекст розмови.

Для інтеграції можна використати Hugging Interface, щоб пришвидшити відповідь NPC. Для цього виконаємо наступні дії:

**4.2.6.1 Початок роботи з HuggingFace**

HuggingFace містить багато дивовижних мовних моделей і API (API прискореного висновку), щоб безпосередньо підключати їх до проектів (Рис. 4.56).

1. Спочатку створитимо обліковий запис.
2. Коли обліковий запис буде створено, перейдемо на інформаційну панель API Accelerated Inference, натиснемо профіль (у верхньому правому куті) і маркер API [[19](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)] (Рис. 4.57).
3. Скопіюймо маркер API, це ключ, який знадобиться, щоб мати можливість використовувати API (Рис. 4.58).

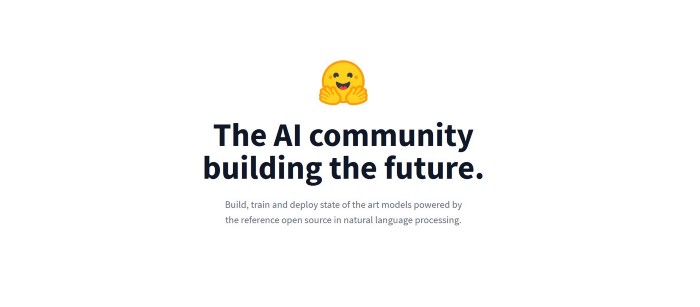
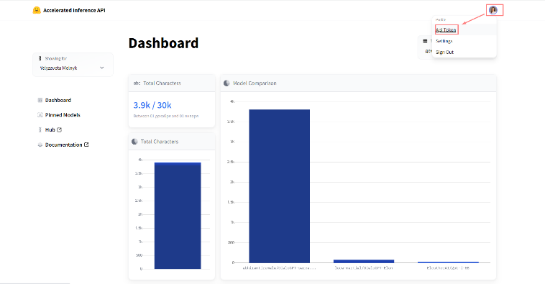


Рис. 4.56 Початкова сторінка Hugging Face вебсайту

Рис. 4.57 Інструментальна панель API прискореного висновку



*Примітка: З міркувань безпеки НЕ треба ДІЛИСЯ ЦІМ КЛЮЧОМ, це приватний ключ.*

**4.2.6.2 API прискореного висновку**

Приємно те, що можемо випробувати модель безпосередньо на веб-сайті за допомогою Hosted Inference API. Зробимо тест:

1. Напишемо що користувача звуть Томас і запитаємо як справи. Як бачимо, мдель відповіла, що добре.
2. Далі відповімо на її повідомлення “Добре”. Бачимо, що модель відповідає “Це чудово” (Рис. 4.59).

Оскільки не маємо підписки на Hugging Face, то маємо до 30 тис. введених символів на місяць. Тому зараз будемо використовувати їх з розумом, й зачекаємо фінальної стадії проекту для тестування моделі.

Тепер, коли є ключ API, можемо вибрати та спробувати модель в дії. Попередньо було вибрано модель GPT-J. Вона була випущена в репозиторії kingofolz/mesh-transformer-jax Беном Вангом і Араном Комацузакі. Це GPT-2-подібна модель причинної мови, навчена на наборі даних Pile.

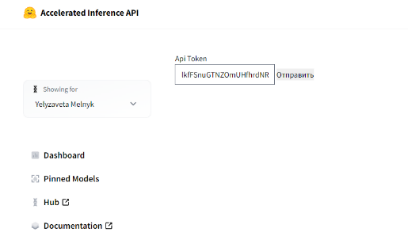


Рис. 4.58 Приватний API Токен

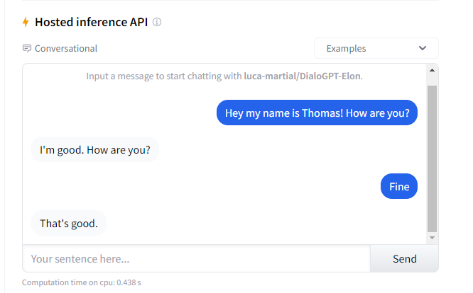


Рис. 4.59 Тест роботи моделі GPTJ онлайн

1. Тепер, коли вибрали модель, потрібно отримати URL-адресу API, для цього натиснемо Розгорнути/Прискорений висновок. Ця функція відкриває модальне вікно, де можемо скопіювати API\_URL (Рис. 4.60).

Тепер все готово щоб підключити API до Unity.

**4.2.6.3 Підключення HuggingFace API до Unity**

Тепер потрібно підключити API моделі Hugging Face до Unity, щоб мати можливість використовувати його.

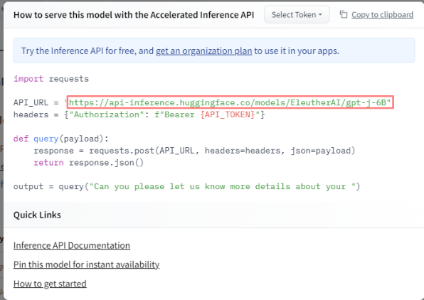
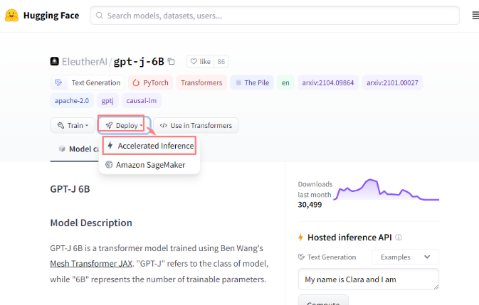


Рис. 4.60 Модальне вікно з API для інтеграції моделі

1. Створимо папку JSON у папці Scripts. Завантажимо туди JSON скрипти для того, щоб можна було відправляти запити й мати необхідні бібліотеки для скрипту.
2. Відкриємо Scenes/AI assistant.
3. Сцена виглядає так, є NPC, оточений різними об’єктами.
4. Створимо скрипт HuggingFaceAPI для виклику Hugging Interface, відкриємо файл й додамо опис функціоналу (Рис. 4.61).

Цей скрипт оброблятиме запит POST, щоб робить запитати API, далі чекає на обробку запиту. У випадку помилки повертає інформацію про помилку, у випадку правильної обробки – повертає відповідь NPC.

**Функція HFScore(): викликає API**

Щоб викликати API та обробити результат, будемо використовувати функцію Coroutine, оскільки цей тип функції може чекати виконання, адже після звернення потрібно зачекати, поки цей API поверне відповідь, перш ніж продовжити виконання.

* Спочатку формуємо JSON для запиту POST
* Потім робимо веб-запит і повертаємо відповідь.
* Відповідь являє собою строку тексту, яку ми виведемо юзеру, як відповідь на його текстове повідомлення.

Структура запиту:

[{"generated\_text":"Hi! I'm glad to hear that you like the site! If you have time to look, you will see that I have tried to stick to the old design as much as possible. A lot of the sites I have seen have changed to look"}]

Перед виведенням результату, спочатку потрібно прибрати непотрібні знаки й текст, тому напишемо функція для виділення у відповіді до запиту лише тексту.

1. У функції ProcessResult(), передамо параметер з відповіддю від GPTJ. Далі видалимо початкові символи, перед початком речення й зайві символи в кінці повідомлення. Тепер текст готовий до обробки.

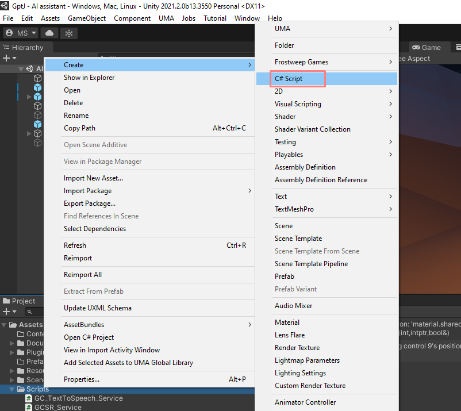


Рис. 4.61 Створення скрипта HuggingFaceAPI для використання GPT-J онлайн

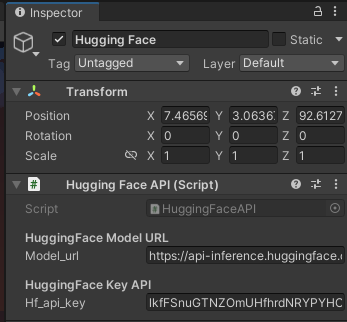


Рис. 4.62 Hugging Face API скрипт

**Заповнимо інспектор**

1. Створимо пустий об’єкт й назвемо його Hugging Face. Додамо до нього скрипт.
2. Останній крок, перш ніж працювати над сценою й перейти до наступного кроку, — це заповнити в інспекторі ключ API та URL-адресу моделі.
3. Натиснемо на об’єкт HuggingFaceAPI у сцені та в інспекторі оновимо Model\_url та Hf\_api\_key.

*Важливо: не ділилися з по сторонніми проектом, якщо ключ API визначено на сцені або в скрипті.*

**4.2.6.4 Створення контроллера для Аватару**

1. Створимо скрипт й назвемо його CharacterBehavior (Рис. 4.63). Цей клас використовується для керування поведінкою персонажа (відповідь на запитання) шляхом виклику екземпляра API HuggingFace.
2. Додамо скрипт на персонажа.
3. Створимо змінні для збереження інформації про інтерфейс користувача (Рис. 4.63). Приклад коду для створення змінних показано на Рис. 4.64.

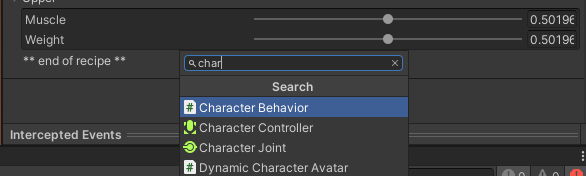


Рис. 4.63 Додання скрипта контролера й присвоєння посилань для полів

1. Створимо змінну для присвоєння початкового стану аніматора персонажу. Й при запуску скрипта присвоємо стандартну idle анімацію.

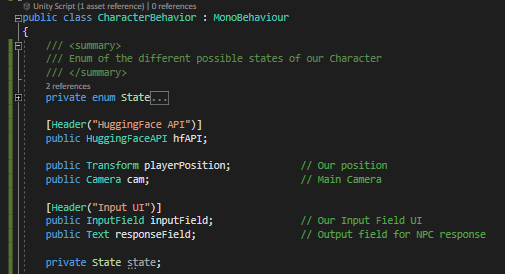


Рис. 4.64 Створення змінних

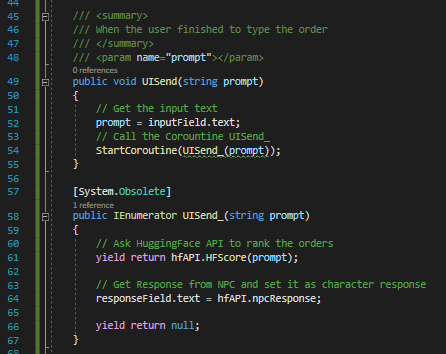


Рис. 4.65 Створення функцій для виклику Hugging Face API

1. Створимо функцію UISend(), що буде викликатися як тільки юзер припинить набирати повідомлення й натисне кнопку відправити. У цій функції відправимо повідомлення користувача на сервер, до GPTJ для обробки повідомлення (Рис. 4.65).
2. Як тільки обробка повідомлення закінчиться, отримаємо відповідь GPTJ й виведемо її користувачу. Це відбувається у функції UISend(), де параметром являється повідомлення користувача.

**4.2.6.5 Створення інтерфейсу користувача**

Створимо прототип для інтерфейсу користувача.

1. Створимо новий Canvas.
2. Додамо поле для вводу й текстове поле для виводу відповіді (Рис. 4.66).
3. Присвоємо значення полів для скрипта CharacterBehavior.
4. Збережемо проект. Тепер все готово для тестування.
5. Перейдемо у Play Mode й протестуємо як працює програма (Рис. 4.67).
6. Введемо привітання й натиснемо Enter, для відправлення повідомлення.
7. Бачимо, що отримали відповідь від GPTJ. Тож тепер час перейти до наступних кроків.

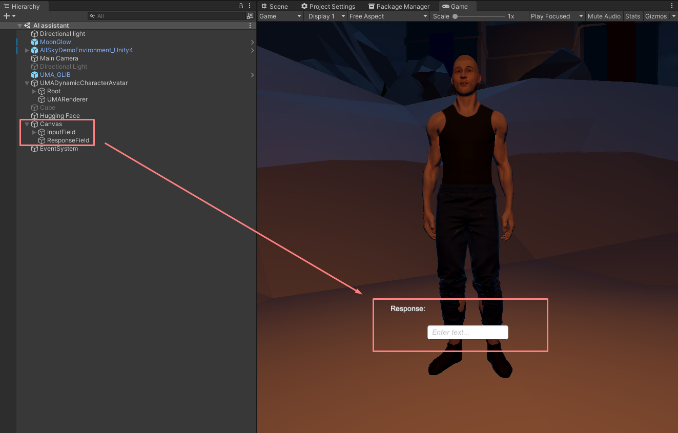
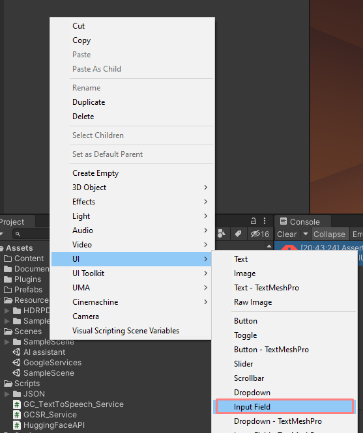


Рис. 4.66 Створення прототипу інтерфейсу користувача

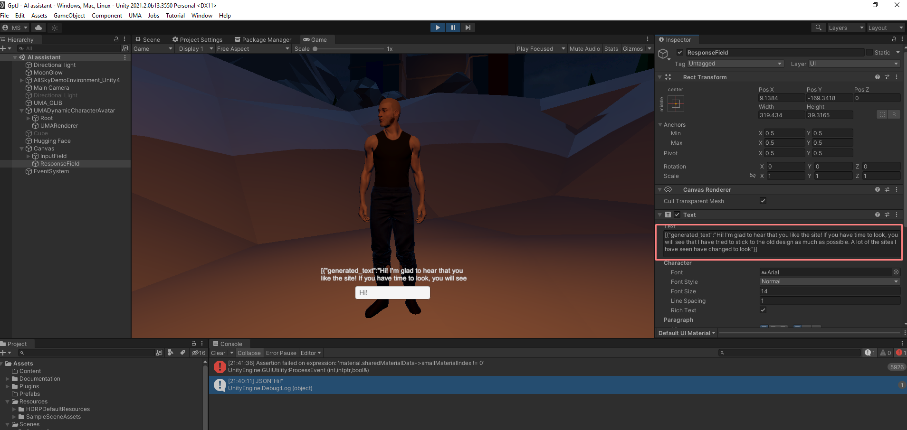


Рис. 4.67 Відповідь GPTJ на повідомлення

1. Додамо змінну textToSpeech, для того щоб користуватися сервісом для перетворення тексту в мовлення (Рис. 4.68).
2. При запуску скрипта, знайдемо об’єкт, що має компонент для перетворення тексту в мовлення й у функції, де ми виводимо відповідь від GPTJ – запустимо функцію для перетворення тексту в мовлення.



Рис. 4.68 Запуск функції перетворення тексту в мовлення

### **4.2.7 Створення й налаштування середовища для Аватару**

**4.2.7.1 Створення контролера для звуку**

Аудіо є досить цікавою концепцією, яку потрібно охопити не тільки в ігровому дизайні, але й при вивченні його природи загалом. Спосіб його сприйняття залежить від багатьох факторів, які потрібно враховувати, коли намагаєтеся контролювати або використовувати його.

Сприйняття аудіо, як правило, включає положення джерела та швидкість його руху, якщо воно взагалі рухається (ефект Доплера).

В Unity розташування джерела звуку важливо для характеристики його джерела. Наприклад, якщо потрібно переконатися, що gameObject, який відтворює звук водоспаду, збігається з фактичним gameObject водоспаду, і що сприйняття гравцем цього звуку є реальним. Збільшення гучності в міру наближення програвача, переконання, що панорамування звуку змінюється в міру зміни відносного положення водоспаду щодо програвача тощо.

Тож маємо справу з 2 основними компонентами, пов’язаними з аудіо в Unity, це:

* Слухач аудіо - це компонент, який автоматично підключається до основної камери щоразу, коли створюється сцена. Вона не має жодних властивостей, оскільки єдина її робота — виконувати роль точки сприйняття. Рекомендується залишити Audio Listener як є.
* Джерело звуку - це компонент, який фактично відповідає за відтворення звуку. У звичайній практиці розробки, як правило, добре зробити порожній gameObject, щоб виступати в якості джерела звуку, і зробити його дочірнім, коли маємо справу з великими складними структурами, щоб мати чітке уявлення про те, де знаходиться джерело звуку.

Компонент Audio Source має досить багато властивостей, з якими можемо повозитися. Це включає його висоту, панорамування, просторове змішування, і якщо відкриємо налаштування 3D-звуку, знайдемо варіанти додавання ефектів Доплера та зменшення гучності.

Однак найбільше цікавий слот AudioClip. Саме сюди йде звуковий ефект, який потрібно відтворити. Unity підтримує досить багато поширених звукових форматів, включаючи .mp3 та .ogg тощо.

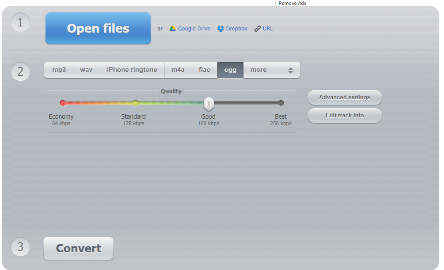
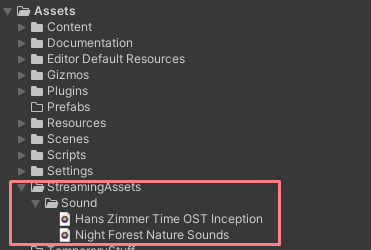


Рис. 4.69 Конвертація звуку в формат ogg та додання файлів в проект

1. Отже тепер знаючи основи й головну ідею завантажимо аудіо дорожки з ютубу у форматі mp3(Hans Zimmer [[21](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)] – Time й Звуки болота вночі [[22](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]) й потім конвертуємо звук у формат .ogg на сайті для конвертації аудіо онлайн [[20](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)].
2. Помістимо конвертовані звуки в папку Streaming Assets/ Sound, попередньо створивши папку для звуку (Рис. 4.69).
3. Створимо скрипт, що буде контролювати музику, що грає на фоні й назвемо його MusicControl (Рис. 4.70).

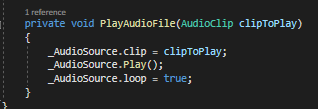
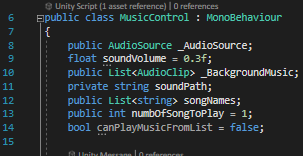


Рис. 4.70 Створення змінниз й функції для відтворення звуку

1. Створимо змінні для збереження інформації про джерело звуку, гучність звуку, розташування аудіофайлів, назви пісень, номер пісні що будемо відтворювати на фоні й змінну, щоб знати чи завантажені аудіо файли в пам'ять чи ні (Рис. 4.70).
2. На початку завантаження скрипта (тобто в функції Awake()) припишемо імена пісень у списку, завантажимо шлях до пісень й налаштуємо гучність відтворення звуку.
3. Створимо функцію LoadAudio з параметром audioName, що буде завантажувати аудіо файл, створюва новий кліп для цього файлу, заповнювати інформацію про вайл й додаватиновий аудіо кліп до списку музики для фону.
4. Винесемо функцію, щ завантажує кожен окремий звук, в окрему функцію GetAudioFromFile(). Це допоможе зачекати завантаження звуку і після завершення цієї дії приступити до створення кліпу, адже якщо файл великий і завантажуватиметься довго, міг бі бути шанс що створення кліпу почнеться раніше за закінчення його завантаження. Аби уникнути цієї ситуації використаємо карутини (IEnumerator) (Рис. 4.71).
5. Створимо функцію, що буде відтворювати звук й назвемо її PlayAudioFile(). Спочатку присвоїмо джерелу аудіо новий кліп, далі запустимо його відтворення й включимо функцію повторного відтворення аудіо файлу (Рис. 4.70).
6. Протестуємо як працює програма. Як бачимо після завантаження пісень аудіо кліп починає відтворюватися, а отже все правильно працює.

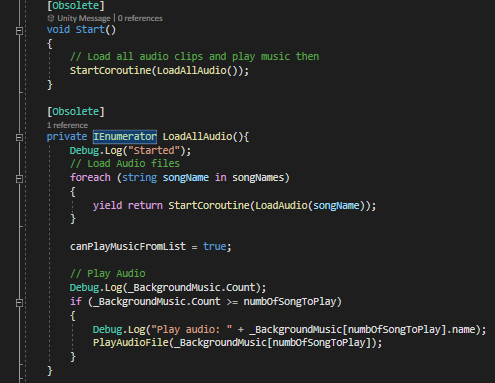
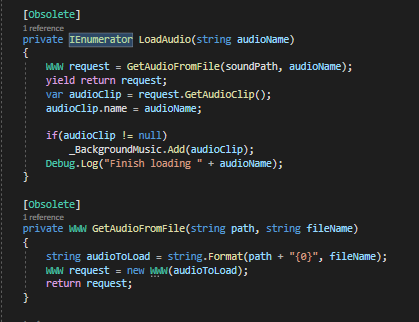


Рис. 4.71 Функції для завантаження аудіо звуку

**4.2.7.2 Створення й налаштування середовища й світла**

Щоб персонаж не стояв просто на боксі, потрібно додати середовище, де він буде знаходитися, а також створити правильне освітлення, позицію камери й налаштування музики для настрою. Останнє вже є, тож приступимо до налаштування сцени в цілому .

**Додання середовища для персонажа**

В ассеті AllSky маємо елементи для сцени, які можна використати.

1. Тому додамо готову демонстраційну сцену. Вона показує просте низько полігональне середовище, щоб продемонструвати налаштування освітлення та туману для неба.
2. Переглянемо сцену в цілому й знайдемо гарне місце для розташування персонажу.
3. Прибиремо бокс на якому стояв персонаж, й розмістимо його прямо в центрі сцени (Рис. 4.72).

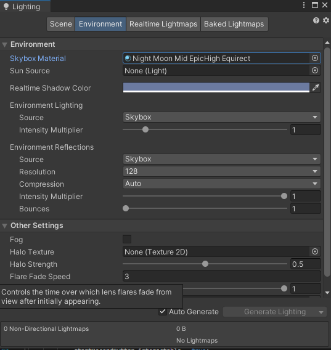
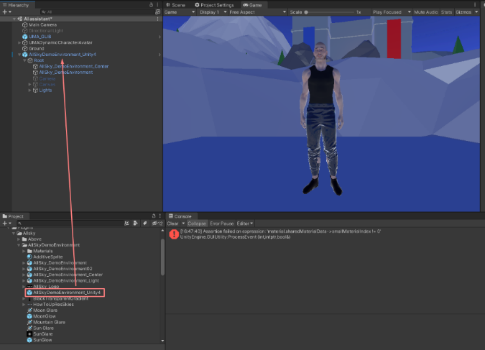


Рис. 4.72 Створення середовища й налаштування освітлення

**Додання й налаштування Skybox й освітлення**

Одне з головних рішень, яке необхідно прийняти щодо освітлення навколишнього середовища, — це система неба, яку збираємося використовувати у сцені. Щоб бути технічним, небо малюється камерою перед візуалізації кадру. Компонент неба знаходиться в компоненті візуального середовища HDRP, який знаходиться в томі, тому насправді обсяг є межею. Структура томів у HDRP — це потужна система для керування глобальними та локальними змінами налаштувань за замовчуванням, одним із яких є налаштування неба.

Саме тому для проекту використаємо одні з наявних в асеті хмар.

1. Переглянемо наявні в ассеті SkyBox-и й підберемо щось, що гарно виглядає.
2. Хмари вечірнього неба з половиною місяця виглядають чудово, тому оберемо їх для налаштування сцени. Сцена буде освітлена у контурі відтворення прямого освітлення з кольоровим простором лінійного освітлення – це значно покращить візуальний вигляд в цілому.
3. Щоб побачити налаштування освітлення, в проекті потрібено перейти в налаштування освітлення. (У розділі «Редагувати»->Налаштування проекту->Програвач) (Рис. 4.72).
4. Тут у вкладці середовище, додамо обраний SkyBox хмар (Night Mid Moon Hight Epic).
5. Для демонстраційної сцени можна збільшити кількість світла Pixel у налаштуваннях якості та відстань спрямованої світлової тіні. Також налаштуємо окремі параметри світла, аби покращити вигляд сцени (Рис. 4.72).
6. Як бачимо кінцевий результат виглядає набагато краще, ніж це було до того (Рис. 4.73).

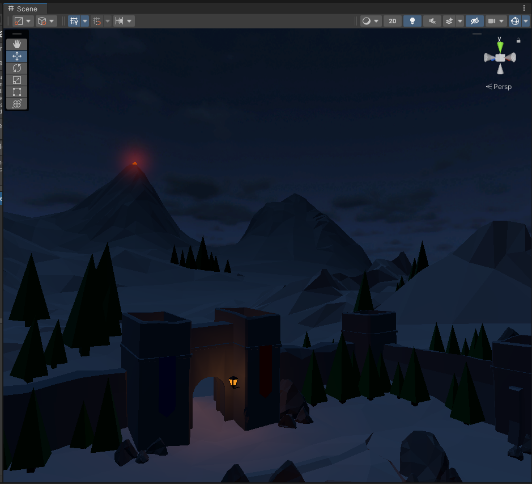


Рис. 4.73 Сцена з персонажем після налаштування

**Позиціонування персонажу і камери**

Тепер необхідно виправити розташування камери й вигляд персонажу.

1. Покрутимо камеру навколо осі й постараємося знайти гарний ракурс, де розташуємо персонажа.
2. Після знаходження ракурсу перемістимо персонажа це місце й налаштуємо кут зору камери так щоб було зручно з ним говорити.
3. Все виглядає чудово, проте не дуже світло (забагато тіней на персонажі). Додамо осередок світла й помістимо його прямо над нашим персонажем.
4. Пограємо з налаштуванням кольору освітлення й оберемо колір ближчий до помаранчево-коричневого. Постараємося знайти найкраще позиціонування світла, щоб персонаж виглядав, як в добрій ламповій атмосфері.

Чудово на цьому налаштування середовища закінчено, проте є ще пару фінальних штрихі (Рис. 4.73).

Об’єкти в інспекторі знаходяться в повному хаосі, тому створимо пусті об’єкти для збереження й групування об’єктів на сцені. Створимо об’єкт для інтерфейсів, освітлення, середовища й для збереження пустих об’єктів з скриптами. Згрупуємо об’єкти на сцені й розташуємо їх у ці нові групи (так щоб вони стали дочірніми об’єктами) (Рис. 4.74).

Тепер останній крок об’єднаємо 2 сцени в одну, це допоможе легше призначати посилання на об’єкти (Рис. 4.75). Після об’єднання сцен перевіримо чи всі посилання збереглися й присвоюємо значення, що були втрачені.

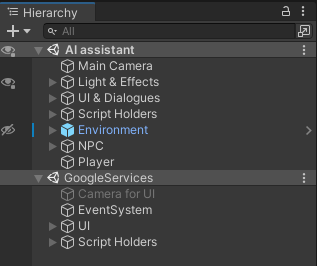


Рис. 4.74 Впорядкуємо структуру проекту

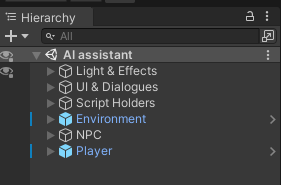


Рис. 4.75 Об’єднання сцен з групуванням об’єктів

**Налаштування Інтерфейсу користувача**

Останній крок – замінимо стрий UI, на більш новий й гарний інтерфейс, для цього використаємо Dark Theme UI ассет.

1. Змінимо розмір екрана на Full HD (1920x1080).
2. Знайдемо панель, що гарно виглядає й може підійти для інформації про GPT-J. Панель Content1 досить гарно вписується проте треба внести де-які зміни.
3. Розмістимо панель на екрані й перейменуємо в NPC info. Скриємо елементи, що не потрібні й залишемо активними лише Image, User Title, Rank Text й Response Text (копія Points Test зі дещо зміненим налаштуванням).
4. Розмістимо панельтак щоб вона охоплювала майже всю ширину, залишимо отступи в 20 пікселів. По висоті з верху також 20 пікселів, а з низу підкорегуємо положення так щоб контент був на голову вище, ніж персонаж.
5. Змінимо розмір поля для виводу відовіді GPTJ й дозволимо відповіді розширюватися вниз.
6. Змістимо написи й іконку максимально в ліво, залишивши трохи відступ від рамки. В Text Rank напишемо назву AI (GPT-J), у User Title – напишемо ім’я героя Richard.
7. Тепер потрібно знайти меню, що може підйти для поля вводу. Меню з налаштуванням виглядає досить солідно, додавши декілька змін буде ідеально виглядати.
8. Додамо на сцену меню з налаштуванням. Приберемо з середини усі елементи й налаштуємо зовнішній вигляд.
9. Зменшимо інтенсивність кольору фону. Тепер розтягнемо форму так щоб прямокутниу був більш плоский. Розмістимо її з низу екрану по центру (Рис. 4.76).
10. Перемісимо попередню форму вводу сюди ж, щоб вона була дочірнім елементом фону. Скриємо колір фону для фрми вводу (Рис. 4.76).
11. Змінимо кольори вводу з чорних на білі. Зкорегуємо відображення тексту, його розмір й розташування у формі.
12. Додамо кнопку для запису аудіо файлів. Знайдемо нову іконку, що підходила би до дизайна. Замінимо стару іконку новою (Рис. 4.76).
13. Видалимо всі зайві об’єкти з сцени (старий інтерфейс).
14. Виправимо елементи, де загублено посилання на матеріал або шрифт.
15. Додамо до елементів меню скрипти, які біли на сарому меню й налаштуємо загублені посилання.

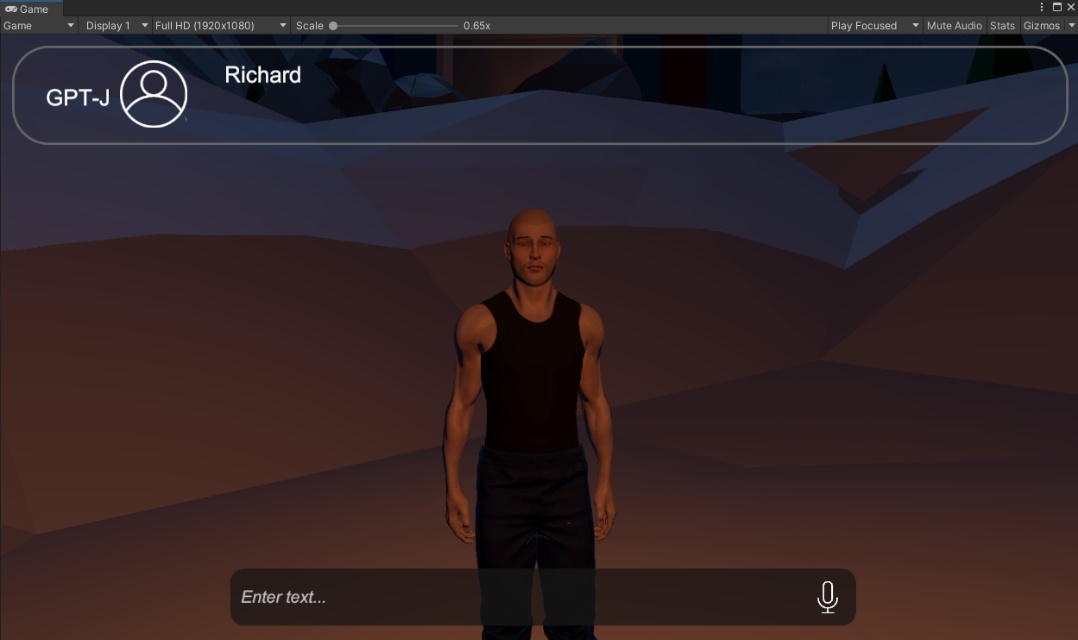


Рис. 4.76 Фінальний інтерфейс користувача

Тепер меню готове до користування, тож протестуємо як працює програма.

**Створення ефекту друкарської машинки**

1. Знайдемо підходящий шрифт й замінимо стандартний Areal на Montserrat (Рис. 4.77).
2. Також у полі для виведення слів користувача додаймо його ім’я (Richard).
3. Створимо скрипт UITextTypeWriter, що буде писати відповідь GPT-J у стилі друкарської машинки. Це допоможе краще розуміти текст й бачити візуально що каже герой (Рис. 4.77).
4. Додамо цей скрипт прямо під панеллю, для виводу тексту (ResponseField).
5. Створимо 2 змінні. Змінну для збереження поля для виводу тексту, й змінну для збереження тексту, що потрібно вивести.
6. На початку завантаження скрипта візьмемо компонет текст з об’єкта, де розташовано скрипт, збережемо текст цього компонета й зітремо його.

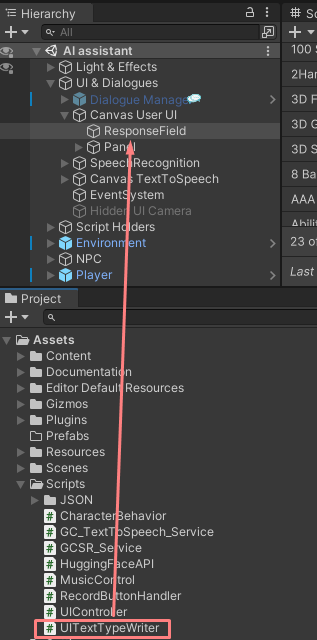
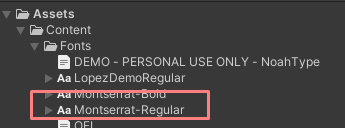
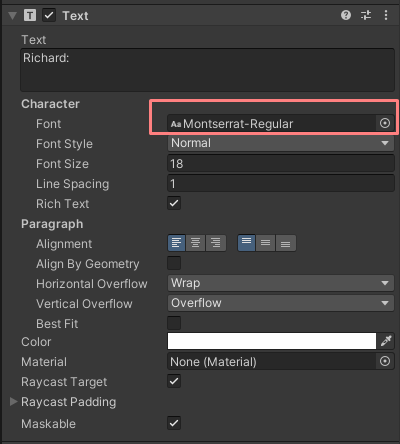


Рис. 4.77 Додання скрипту на сцену. Завантаження й заміна шрифтів на сцені

1. Створимо функцію PlayText(), що буде відображати текст символ за символом, тим самим імітуючи ефект друкарської машинки.
2. Створимо функцію DisplayText(), яку ми будемо викликати, щойно портібно відобразити текст що GPT-J говорить. Ця функція прописує ім’я нашого персонажа відразу (Richard), а потім поступово напирає повідомлення, що було отримано від GPT-J.

# **ВИСНОВКИ**

Голосові помічники — це програми на цифрових пристроях, які прослуховують словесні команди та відповідають на них. Користувач може сказати: «Яка погода?» а голосовий помічник відповість прогнозом погоди на цей день і місцеположення. Вони можуть сказати: «Розкажи мені історію», і помічник кинеться в казку. Користувач може навіть сказати: «Замовте мою улюблену піцу», і вечеря буде в дорозі!

Голосові помічники настільки прості у використанні, що багато людей забувають зупинитися і ДІВУЮТЬСЯ, як вони працюють. Як нас розуміють голосові помічники? Це магія? Складна система кодів? Справжня людина, яка слухає на іншому кінці? Відповідь менш складна, ніж ви думаєте.

Все починається з сигнального слова. Ви коли-небудь називали ім’я друга, щоб привернути його увагу? З тієї ж причини користувачі називають імена своїх голосових помічників. Вони можуть сказати: «Привіт, Сірі!» або просто «Алекса!» Яким би не було сигнальне слово, воно будить пристрій. Це сигналізує голосовому помічнику, що він повинен почати звертати увагу.

Після того як голосовий помічник почує своє сигнальне слово, він починає запис. Пристрій чекає паузи, щоб дізнатися, що ви завершили свій запит. Потім голосовий помічник надсилає ваш записаний запит через Інтернет до своєї бази даних.

Потрапивши в базу даних, ваш запит порівнюється з іншими запитами. Він розділений на окремі команди, які може зрозуміти ваш голосовий помічник. Потім база даних надсилає ці команди назад голосовому помічнику. Отримавши команди, голосовий помічник знає, що робити далі. Пристрій може поставити запитання, щоб переконатися, що він розуміє, що ви хочете. Якщо він вважає, що розуміє, голосовий помічник виконає поставлене вами завдання.

Якщо ви коли-небудь користувалися голосовим помічником, то знаєте, що все це відбувається дуже швидко! Якщо ви скажете: «Привіт, Сірі! Яка погода?» Siri повідомляє вам за секунди. Чим більше вказівок отримають пристрої, тим краще і швидше вони будуть виконувати наші запити.

Голосові помічники постійно вдосконалюються. Вони вчаться так само, як і люди. Ви коли-небудь просили голосового помічника про щось і отримували результати, на які ви не очікували? Кожного разу, коли це відбувається, голосовий помічник вчиться на своїй помилці. Якщо ви скажете: «Алекса, грай рок-музику», а вона грає кантрі, ви скажете Alexa зупинитися. Це вчить пристрій, що команда, яку він отримав зі своєї бази даних, була неправильною. Він передає це з базою даних і намагається зробити краще наступного разу.

Голосові помічники можуть робити практично все, що ви можете придумати. Вони відповідають на запитання, телефонують і навіть можуть керувати такими частинами вашого будинку, як світло та термостат! Це звучить як наукова фантастика? Подумати ще раз! Понад 35 мільйонів американців вже користуються голосовими помічниками, і їх кількість зростає! Які запити ви будете робити?

# **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу:  <https://www.knowledgeisle.com/wp-content/uploads/2019/12/2-Aur%C3%A9lien-G%C3%A9ron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O%E2%80%99Reilly-Media-2019.pdf>
2. Deep Learning with Python [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим  доступу:  <https://tanthiamhuat.files.wordpress.com/2018/03/deeplearningwithpython.pdf>
3. The Simulation Hypothesis: An MIT Computer Scientist Shows Why AI, Quantum Physics, and Eastern Mystics All Agree We Are in a Video Game [Електроний ресурс]: [Веб-сайт].- [Електронні  данi].-Режим доступу: <https://www.amazon.com/Simulation-Hypothesis-Computer-Scientist-Quantum/dp/B07RWQ4GKK/ref=zg_bs_3887_45?_encoding=UTF8&psc=1&refRID=N98JS73T5511A5PF378R>
4. Github repository [Електроний ресурс]: [Веб-сайт].- [Електронні данi].-Режим доступу: <https://github.com/kingoflolz/mesh-transformer-jax>
5. Builtin [Електроний ресурс]: [Веб-сайт].- [Електронні данi].-Режим доступу:  <https://builtin.com/artificial-intelligence>
6. Mobinius Blog [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://www.mobinius.com/blogs/advantages-of-ai-as-smart-assistant>
7. The Pile paper, Arxiv [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим  доступу: <https://arxiv.org/pdf/2101.00027.pdf>
8. The Pile [Електроний ресурс]:[Веб-сайт].- [Електронні  данi].-Режим доступу: <https://pile.eleuther.ai/>
9. Unity3D Machine Learning [Електроний ресурс]:[Веб-сайт].- [Електронні данi].- Режим доступу: <https://unity3d.college/2017/10/25/machine-learning-in-unity3d-setting-up-the-environment-tensorflow-for-agentml-on-windows-10/>
10. A Game Developer Learns Machine Learning - Getting Started [Електроний ресурс]:[Веб-сайт].- [Електронні данi].- Режим доступу: <https://mikecann.co.uk/posts/a-game-developer-learns-machine-learning-getting-started>
11. A tour of the C# language [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
12. The Pile: An 800GB Dataset of Diverse Text for Language Modeling [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://arxiv.org/pdf/2101.00027v1.pdf>
13. GPT-J-6B Notebook [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://github.com/mallorbc/gpt-j-6b>
14. Python Releases for Windows [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://www.python.org/downloads/windows/>
15. Anaconda Individual Edition [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу:<https://www.anaconda.com/products/individual#windows>
16. How to use Machine Learning AI in Unity! (ML-Agents) [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://www.youtube.com/watch?v=zPFU30tbyKs>
17. Unity ML-Agents Toolkit [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://github.com/Unity-Technologies/ml-agents>
18. ML-Agents: Hummingbirds [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://learn.unity.com/course/ml-agents-hummingbirds/?tab=live>
19. Accelerated Inference API Dashboard [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://api-inference.huggingface.co/dashboard/>
20. Online Audio Converter [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://online-audio-converter.com/>
21. Hans Zimmer - Time (OST "Inception") │ Fingerstyle guitar 2 Hour Loop [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://www.youtube.com/watch?v=z3BHrFvqZmE&t=12s>
22. Swamp Sounds at Night - Frogs, Crickets, Light Rain, Forest Nature Sounds | 3 Hours [Електроний ресурс]:[Веб-сайт].- [Електронні данi].-Режим доступу: <https://www.youtube.com/watch?v=ih4_1FyVjaY&t=2065s>