

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Додаток для перекладу аудіо та тексту на основі
штучного інтелекту»**

Виконав студент групи ІПЗм-24-1
ЄРЕМЕНКО Артем Ігорович

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЄРЕМЕНКО Артем Ігорович**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Додаток для перекладу аудіо та тексту на основі штучного інтелекту»**,

затверджена Наказом закладу вищої освіти від **«18» вересня 2025 р. № 491/с**

Вихідні дані для роботи: методи та алгоритми тренування штучного інтелекту для перекладу аудіо та тексту та впровадження штучного інтелекту у переклад аудіо та тексту.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	22.09.2025р.	22.09.2025р.
2	Чижмотря О.В.	22.09.2025р.	22.09.2025р.
3	Чижмотря О.В.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» складається з опису перекладача для аудіо та тексту на основі штучного інтелекту та дослідження методів та алгоритмів тренування штучного інтелекту. Пояснювальна записка складається з вступу, трьох розділів та списку літератури. Пояснювальна записка до кваліфікаційної магістерської роботи містить 57 сторінок, 28 ілюстрацій, 2 таблиці, 3 додатки та 40 посилань на зовнішні джерела.

Метою роботи є розробка додатку для перекладу аудіо та тексту на основі штучного інтелекту.

В роботі описано постановку завдання та вимоги системи, оглянуто існуюче програмне забезпечення за тематикою проєкту. Також описано вибір інструментальних засобів для реалізації додатку. Далі описано аналіз методів та алгоритмів тренування штучного інтелекту. Після чого описано створення та дослідження алгоритму навчання моделі та оглянуто його результати. До того ж описано процес проєктування додатку для перекладу аудіо та тексту на основі штучного інтелекту та оглянуто його інтерфейс.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, FLASK, JINJA2, PYTHON, MT5.

ABSTRACT

The master's thesis consists of a description of the translator for audio and text based on artificial intelligence and a study of methods and algorithms for training artificial intelligence. The explanatory note consists of an introduction, three sections and a list of references. The thesis note contains 57 pages, 28 illustrations, 2 tables, 3 appendix and 40 links to external sources.

The purpose of the work is to develop an application for translating audio and text based on artificial intelligence.

The work describes the statement of the task and system requirements, reviews existing software on the topic of the project. The selection of tools for implementing the application is also described. Next, an analysis of methods and algorithms for training artificial intelligence is described. After that, the creation and study of the model training algorithm is described and its results are reviewed. In addition, the process of designing an application for translating audio and text based on artificial intelligence is described and its interface is reviewed.

KEYWORDS: ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, FLASK, JINJA2, PYTHON, MT5.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ, АНАЛІЗ АНАЛОГІВ ТА АЛГОРИТМІВ РЕАЛІЗАЦІЇ ДОДАТКУ ПЕРЕКЛАДУ АУДІО ТА ТЕКСТУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ	9
1.1. Постановка задачі	9
1.2. Огляд та аналіз наявних аналогів	10
1.3. Обґрунтування вибору інструментальних засобів та архітектури перекладача на основі штучного інтелекту	14
1.4. Огляд методів та алгоритмів машинного навчання, опис моделі mT5	16
Висновки до розділу 1	21
РОЗДІЛ 2. АНАЛІЗ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТРЕНУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ	22
2.1. Аналіз задачі та методів її вирішення	22
2.2. Характеристика джерела даних та розробка алгоритму	25
2.3. Аналіз проведеного тренування моделі для перекладу тексту	36
Висновки до розділу 2	39
РОЗДІЛ 3. РОЗРОБКА ТА ОПИС ДОДАТКУ	40
3.1. Проєктування додатку	40
3.2. Огляд інтерфейсу.	42
Висновки до розділу 3	46
ВИСНОВКИ.....	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	48
ДОДАТКИ.....	52

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Batch – купа, пакет, вибірка даних.

NLP – (англ. natural language processing) – обробка природної мови.

NLG – (англ. natural language generation) – генерація природної мови.

Датасет – (англ. dataset) – набір даних.

ВСТУП

У наш час, коли кожна частина суспільства, швидко прямує до глобалізації усіх сфер її діяльності, проблема мовних бар'єрів зустрічається все частіше. На основі чого зростає потреба у нових інструментах швидкого та доступного перекладу, не тільки тексту а й аудіо. Завдяки розвитку та дослідженню технологій штучного інтелекту та їх інтеграції у системи перекладу, людство наближається до вирішення даної перешкоди. Перекладачі на основі штучного інтелекту надають можливість швидкого та точного перекладу мов у будь-якому форматі.

Додатки такого типу дозволяють обійти проблему мовного бар'єру у різних сферах життя: швидкий переклад під час зустрічі з іноземцями, також не слід забувати про звичайне спілкування людей у мережі інтернет. Наукова сфера також має багато користі, пришвидшення процесів передачі інформації між науковцями різних країн позитивно впливає на строки досліджень та їх точність.

Мета роботи полягає в тому, щоб розробити додаток для перекладу тексту та аудіо на основі штучного інтелекту.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Правильно розробити постановку завдання.
2. Оглянути та порівняти існуючі аналоги.
3. Обрати програмні інструменти для реалізації.
4. Дослідити наявні методи та алгоритми.
5. Розробити алгоритм машинного навчання.
6. Реалізувати додаток з використанням натренованої моделі MT5.

Об'єктом дослідження є переклад аудіо та тексту за допомогою штучного інтелекту.

Предметом дослідження є методи та алгоритми тренування штучного інтелекту для перекладу тексту.

Наукова новизна роботи полягає у використанні та дослідженні сучасних методів машинного навчання на прикладі моделі MT5, для подальшого використання у процесах перекладу тексту та аудіо.

Практичне значення полягає у наданні можливості швидкого контекстуального перекладу тексту та аудіо у вигляді зручного додатку.

Апробація. За тематикою кваліфікаційної роботи було опубліковано дві тези доповідей [1] [2].

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ, АНАЛІЗ АНАЛОГІВ ТА АЛГОРИТМІВ РЕАЛІЗАЦІЇ ДОДАТКУ ПЕРЕКЛАДУ АУДІО ТА ТЕКСТУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

1.1. Постановка задачі

Багатомовна комунікація у сучасному світі є невід’ємною частиною міжнародного бізнесу, освіти та навіть буденного спілкування. Даний процес породжує проблеми якісного та швидкого перекладу інформації на різні мови. Інформація у свою чергу може бути у форматі тексту та аудіо. Для кожної сторони є зацікавленість у передачі та отриманні інформації у найточнішому вигляді, відносно оригіналу. Таким чином постає питання у використанні інструментів для перекладу, з метою зберегти початковий сенс та суть повідомлення.

Існуючі додатки найчастіше фокусуються на одному типі перекладів, також доволі часто виникають помилки, втрати оригінального сенсу. З розвитком технологій штучного інтелекту та таких підходів, як обробка природної мови, точність перекладів значно зросла. На сьогоднішній день, існує низка додатків та вебсайтів, які надають можливості контекстного перекладу за допомогою штучного інтелекту, вони фактично замінили звичайні перекладачі, через більшу точність перекладів. Також не слід забувати, що використання перекладачів, з метою перекласти аудіо у текст іншої мови, допомагає людям з вадами слуху отримувати інформацію.

Отже, мета роботи полягає в тому, щоб розробити додаток для перекладу тексту та аудіо на основі штучного інтелекту.

Для досягнення даного результату мету можна розділити на наступні завдання:

1. Аналіз існуючих аналогів.
2. Вибір інструментальних засобів реалізації.
3. Дослідження наявних методів та алгоритмів.
4. Тренування власної моделі штучного інтелекту.

5. Реалізація додатку для перекладу аудіо та тексту.

Дане дослідження зосереджується на підтримці перекладу аудіо та тексту. Планується підтримка 2-х мов: англійська та українська. Головним методом для перекладу є «обробка природної мови», разом з технологією токенізації даних. Для кращого розуміння необхідно розібрати функціональні вимоги додатка.

Основні функціональні вимоги: переклад тексту з однієї мови на іншу, переклад тексту у вигляді файлу (блокнот та документ WORD), трансформація мовлення у текст з файлу (speech-to-text), переклад аудіо у вигляді файлу з однієї мови на іншу, запис аудіо з мікрофону та його переклад у текстовому форматі.

До нефункціональних вимог можна віднести: час розпізнавання та перекладу аудіо не повинен перевищувати 10 секунд, можливість додавання додаткових мов, додаток повинен мати зрозумілий інтерфейс, середній час навчання користувача роботі з додатком не повинен перевищувати 5 хвилин.

Очікуваним результатом є додаток, який буде містити, вище перераховані, методи перекладу, та мати зрозумілий інтерфейс. Швидкість перекладу, завдяки штучному інтелекту повинна бути швидкою, та підтримувати контекстуальний переклад.

1.2. Огляд та аналіз наявних аналогів

На даний момент існує велика кількість додатків та вебсайтів, що спеціалізуються на перекладі інформації за допомогою штучного інтелекту. Вони відрізняються функціональністю та якістю перекладу. Для кращого розуміння очікуваного результату, розглянемо деякі з них.

Першим перекладачем є DeepL Translator [3]. DeepL SE – це компанія, яка спеціалізується на обробці мови інструментами штучного інтелекту, який вони самостійно розробляють. Один з їх продуктів, є нейронний машинний перекладач, який був запущений у серпні 2017 р. Він використовує алгоритми на основі згорткових нейронних мереж для забезпечення високоякісного та контекстуального перекладу (рис. 1.1).

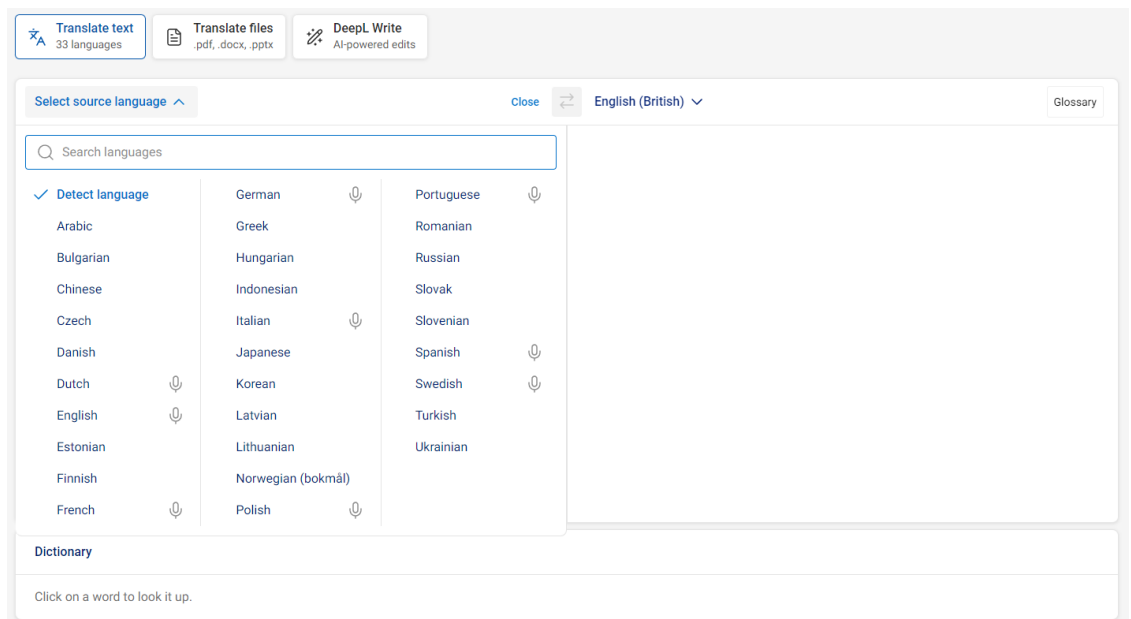


Рис. 1.1 – Інтерфейс перекладача DeepL

Оглянемо характеристики, якими володіє даний перекладач: на даний момент DeepL підтримує 33 мови, серед яких англійська, українська, японська, корейська та інші. Завдяки використанню передових нейронних мереж, переклади мають більш природне звучання у порівнянні зі звичайними перекладачами, також користувачі відмічають про більш точні переклади, враховуючи контекст інформації. Окрім текстового перекладу, сервіс дозволяє перекладати документи у форматах .docx, .pptx та PDF, розміром до 5МБ. Для звичайних користувачів, працює обмеження у 1500 символів за один запит, також існує платна підписка DeepL Pro, яка не тільки знімає ці обмеження, а й надає доступ до додаткового функціоналу. Сервіс також має мобільні додатки, які дозволяють перекладати текст, фото, файли та мовлення (аудіо), вони також можуть надавати інтелектуальні підказки для покращення якості письма на деяких мовах.

До переваг у даному програмному додатку можна віднести високу якість перекладу, зручність використання та конфіденційність перекладів.

Серед недоліків можна навести обмеження кількості мов та обмеження безкоштовної версії.

Перекладач DeepL є одним з найпопулярніших та найпотужніших перекладачів на основі інструментів штучного інтелекту на сьогоднішній день.

QuillBot – це онлайн платформа, що надає набір інструментів на основі штучного інтелекту для покращення письма, перекладу тексту, перефразування та перевірки граматики (рис. 1.2) [4].

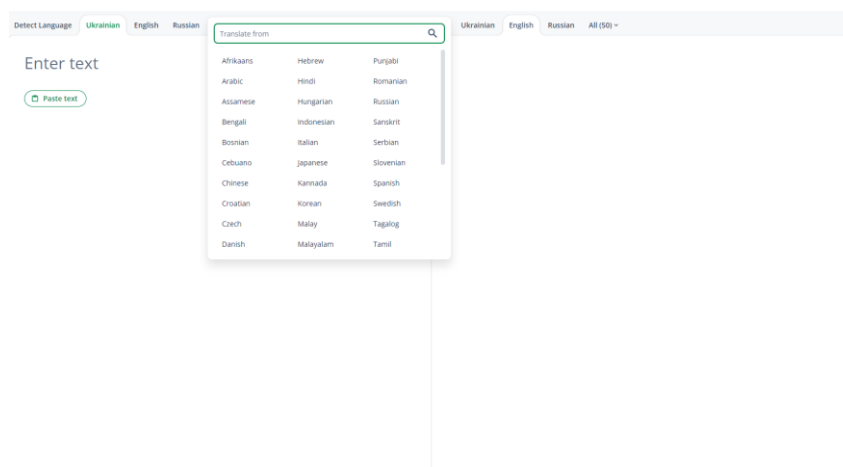


Рис. 1.2 – Інтерфейс перекладача QuillBot

Розглянемо основні характеристики даного сервісу, до яких можна віднести підтримку 50 мов, що є гарним результатом серед конкурентів, платформа має обмеження у безкоштовній версії, користувачі можуть перекладати до 5000 символів за раз та не мають доступу до усього функціоналу, великий перелік лінгвістичних інструментів для покращення навичок письма, інтеграція з великою кількістю інструментів, наприклад підключення до екранної клавіатури мобільного телефону, для виправлення помилок та написання тексту іншою мовою.

До переваг цього програмного застосунку належать: наявність великої кількості можливостей для інтеграції інструментів, відсутність реклами та зручність у використанні.

Застосунок також має деякі недоліки, а саме відсутність можливості перекладу документів та аудіо та обмеження безкоштовної версії.

QuillBot є вузько-спеціалізованою платформою, яка фокусується на покращенні навичок письма та можливості інтегруватися з більшою кількістю популярних інструментів.

OpenL – це програмне забезпечення для перекладу на основі штучного інтелекту, яке підтримує більше ніж 100 мов, включаючи англійську, китайську,

японську, арабську, французьку, німецьку, іврит та індонезійську. Воно використовує ШІ та машинне навчання для точних перекладів і надає допомогу в створенні контенту, виправленні граматики та вивченні мов (рис. 1.3) [5].

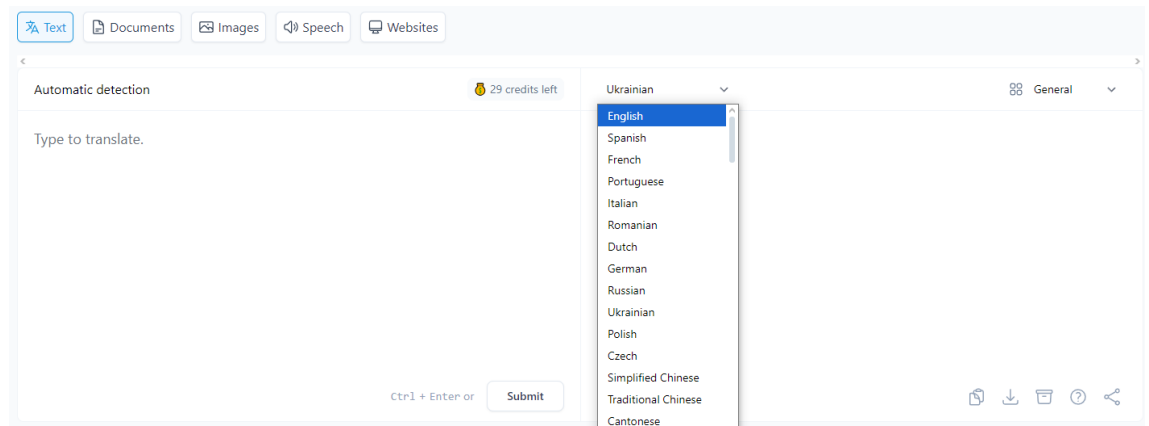


Рис. 1.3 – Інтерфейс перекладача OpenL

Ключовими особливостями перекладача OpenL є підтримка 100 мов, що надає популярності даній платформі, оскільки підтримуються навіть менш популярні мови. Сервіс дозволяє перекладати тексти, документи, зображення та аудіо файли. Окрім перекладу, портал має інструменти для покращення письма, серед яких перевірка граматики, вдосконалення стилю, інструменти для вивчення мов та підведення підсумків тексту. OpenL також має мобільний додаток для IOS, що дозволяє отримати доступ до усіх інструментів з телефону.

Серед переваг, які має даний додаток, можна виділити: високу точність перекладу, універсальність та наявність додаткових інструментів.

Але даний перекладач також містить наступні недоліки: портал має обмеження для безкоштовних користувачів, 30 звичайних перекладів на день, 10 перекладів аудіо, ліміт у 1500 символів, обмеження розміру файлу у 10МБ. Також наявність окремих платних версій для вебсайту та мобільного додатку.

OpenL є потужним інструментом для перекладу великої кількості мов та їх вивчення, про те суворі обмеження безкоштовної версії не дають повністю оцінити потенціал даного сервісу. Нажаль для комфортної роботи з великими текстами, збереженням історії перекладів та доступом до додаткових функцій необхідно купувати платну підписку.

Для кращого розуміння відмінностей аналогів, розглянемо їх у розрізі порівняльної таблиці (табл. 1.1).

Таблиця 1.1

Порівняльна таблиця аналогів

Характеристика	DeepL	QuillBot	OpenL
Підтримка мов	33 мови	50 мов	Понад 100 мов
Якість перекладу	Висока, природні переклади завдяки нейронним мережам	Добра, зручний для щоденного використання	Висока, адаптована до різних форматів
Формати для перекладу	Текст, документи (.docx, .pptx, PDF)	Лише текст	Текст, документи (PDF, Word), зображення, аудіо
Обмеження безкоштовної версії	1500 символів за запит	5000 символів за запит	30 перекладів на день, 1500 символів за запит
Додаткові функції	Інтеграція API, офлайн-режим (Pro)	ПЕРЕФРАЗУВАЛЬНИК, перевірка граматики, підсумовувач	Переклад зображень, перевірка граматики, вдосконалення стилю
Головна перевага	Якість перекладу та точність	Додаткові інструменти для покращення текстів	Універсальність та широкий набір функцій

1.3. Обґрунтування вибору інструментальних засобів та архітектури перекладача на основі штучного інтелекту

Існує велика кількість архітектур додатків, серед яких було обрано клієнт-серверну. Сервер оброблює запити користувачів та надсилає відповіді. Клієнти, у свою чергу, використовують застосунки, які дозволяють спілкуватися з сервером, у даній реалізації використовується браузер. Мережа забезпечує зв'язок, який відбувається у просторі інтернет мережі. База даних відповідає за збереження даних.

Дана модель визначає три рівні операцій: рівень представлення даних – інтерфейс за допомогою якого дані потрапляють до користувача; прикладний рівень – реалізує логіку застосунку, займається обробкою даних, тобто сервер; рівень управління даними – забезпечує зберігання та доступ до даних, реалізується за допомогою СКБД.

Обрану архітектуру необхідно реалізувати за допомогою поєднання інструментів розробки вебінтерфейсу та вебсерверу, тому було обране наступне об'єднання засобів.

Для розмітки вебсторінок застосовуються HTML5 [6], CSS [7], JavaScript [8]. Розробка серверної частини відбувається мовою програмування Python [9] із використанням фреймворку Flask [10]. У якості шаблонізатора, для наповнення сторінок даними, обрано Jinja2 [11]. Додатково залучені такі бібліотеки та інструменти: jQuery [12], Flask-SocketIO [13], Werkzeug [14], OS [15], transformers [16], sentencepiece [17], datasets [18], SpeechRecognition [19]. Для проведення дослідження та виконання завдання перекладу обрано претреновану модель штучного інтелекту mT5 [20].

Сервер є найголовнішою частиною будь якого вебдодатку, мова програмування Python є гнучкою мовою з можливістю реалізації алгоритмів різної складності, що дозволяє зручно розроблювати серверну частину.

Фреймворк Flask містить у собі інструменти для розробки вебдодатків. Він дозволяє розмічати посилання для серверу, оброблювати запити, які до нього надходять, працювати з файлами, що є важливим для системи перекладача та має великий набір інтеграцій з іншими бібліотеками.

Бібліотека Werkzeug, є набором інструментів для роботи, захисту та перевірки працездатності серверу. Вона дозволяє захищати паролі користувачів, за допомогою шифрування SHA256, захищати назви файлів, перед зберіганням на сервері та багато іншого.

Через те, що Flask не підтримує асинхронність, буде використана бібліотека-розширення Flask-SocketIO. Вона дозволяє підключити сторінку до

серверу у асинхронному режимі, це дозволить уникнути зависань інтерфейсу користувача під час обробки даних.

Для тренування та подальшого використання штучного інтелекту, знадобиться бібліотека Transformers. Вона дозволяє працювати з претренованими моделями для їх налаштування під необхідні задачі, у даному випадку це переклад тексту.

SentencePiece використовується для аналізу тексту та його підготовки до вводу у модель штучного інтелекту. В основному вона буде використана для токенизації даних. Токенізація – це розбивка тексту на окремі слова або токени, що дозволяє комп'ютеру краще зрозуміти текст, таким чином перекладач зможе відслідкувати контекст повідомлення та отримати усю необхідну інформацію.

Datasets це бібліотека, яка була створена для уніфікації процесу збору та використання тренувальних даних для моделей штучного інтелекту. Завдяки ній, ми зможемо зручно завантажити необхідні, для тренування, дані, та використати їх [21].

Обрана претренована модель штучного інтелекту – mT5, розроблена на основі моделі T5 [22], вона є мульти-лінгвістичною та працює на основі «уваги», що дозволяє звертати увагу на контекст повідомлення. Важливим фактором є text-to-text підхід, що зводить усі операції до трансформування вхідного тексту у вихідний [23].

Основним фокусом моделі є обробка природної мови (NLP)[24], що ідеально підходить до задач перекладання різних мов. Вона побудована на базі seq2seq [25] модулів, які розраховані на обробку послідовних даних, таких як текст. Також слід зазначити, що вона містить 2 основних блоки: енкодер (обробка вхідного тексту) та декодер (генерація вихідного тексту).

1.4. Огляд методів та алгоритмів машинного навчання, опис моделі mT5

Навчання - це процес, знайомий кожному. Люди навчаються щодня і демонструють чудові результати в цьому процесі. Вони спостерігають закономірності в мінливому середовищі навколо себе, будують певну модель

змін у цьому середовищі і приймають певні рішення. Навколишнє середовище певним чином реагує на прийняті рішення, і люди знову коригують свою модель світу.

Машинне навчання - це спрощена (адаптована до машини) версія процесу навчання, який відбувається з людиною. Зазвичай машинне навчання має певний набір прикладів, спостережень і реакцій на ці спостереження. Завдання полягає в тому, щоб побудувати моделі, які будуть ефективно описувати наявні дані і робити надійні прогнози [26].

Машинне навчання відрізняється від того, до якого ми звикли, тим, що ми намагаємося навчити комп'ютер вчитися.

Існує два типи навчання: індуктивне навчання та дедуктивне навчання.

Індуктивне навчання передбачає спостереження за світом і побудову моделей, які пояснюють причини різних явищ. Ці моделі потім багаторазово перевіряються, деякі з них «виживають», щоб бути використаними і вдосконаленими, тоді як інші повністю відкидаються. Дедуктивне навчання схоже на шкільну математику, де учням дають готові формули і вчать застосовувати їх на практиці [27].

Машинне навчання – це індуктивне навчання, тому що ми вчимо машину вчитися переважно на прикладах, спостерігаючи за великою кількістю реальних прикладів, будуючи на них моделі, тестуючи їх і застосовуючи до більшої кількості прикладів.

Мета машинного навчання полягає у передбаченні результатів на основі вхідних даних. Чим різноманітніші вхідні дані, тим легше машині знаходити закономірності і тим точніший результат.

Отже, якщо ми хочемо навчити машину, нам потрібні три речі: дані, ознаки та алгоритми.

Якщо ми хочемо виявити спам, нам потрібні приклади спам-листів; щоб спрогнозувати ціни на акції, нам потрібна історія цін; щоб дізнатися про інтереси користувачів, нам потрібні їхні вподобання або пости. Нам потрібно якомога більше даних. Десятки тисяч прикладів - це мінімум.

Процес збору даних може відбуватися по різному, хтось робить це вручну - процес повільніший, даних менше, але вони безпомилкові. Інші роблять це автоматично - просто згодовують машині все, що знаходять, і сподіваються на краще. Найрозумніші, як Google, використовують власних користувачів для безкоштовного маркування. Наприклад ReCaptcha, яка іноді просить «знайти всі дорожні знаки на картинці» - далі всі ці дані використовуються для машинного навчання.

Дуже важливим є наявність ознак (властивості, характеристики) – це можуть бути пробіг автомобіля, стать користувача, ціна акцій або навіть лічильник частоти появи слова в тексті.

Машина повинна знати, що саме шукати. Добре, коли дані просто викладені в таблицях – назви їхніх стовпців є ознаками. Але що, якщо у нас є сто гігабайтів фотографій котів? Коли є багато ознак, модель працює повільно та неефективно. Часто вибір правильних ознак займає більше часу, ніж будь-що інше в навчанні. Однак, є й зворотні ситуації, коли користувач вирішує вибрати лише "правильні" ознаки з його точки зору і вводить суб'єктивність у модель, в результаті чого вона починає брутально брехати.

Ту саму задачу майже завжди можна вирішити різними способами. Вибір методу визначає точність, швидкість роботи та розмір отриманої моделі. Але є один нюанс: якщо дані "сміття", то навіть найкращий алгоритм не допоможе.

Для подальшого розуміння необхідно розпізнати відмінності між штучним інтелектом, машинним навчанням та нейронними мережами.

Штучний інтелект – це назва всієї галузі, як біологія чи хімія. Машинне навчання є однією з галузей штучного інтелекту. Важливий, але не єдиний, що існує. Нейронні мережі можна вважати одним із видів машинного навчання. Вони популярні, але є й інші не менш хороші. Глибоке навчання – це архітектура нейронних мереж, один з підходів до їх побудови та навчання. На практиці мало хто розрізняє, де використовуються глибокі нейронні мережі, а де вони не дуже актуальні.

Фокусом цього дослідження є робота штучного інтелекту з текстом, тому необхідно розібрати обробку природної мови.

Природні мови – це ті мови, якими розмовляють люди. Обробка природної мови охоплює все, що потрібно комп'ютеру для розуміння природної мови. Вона є підгалуззю штучного інтелекту та лінгвістики, присвячену тому, щоб комп'ютери розуміли висловлювання або слова, написані людьми. Природна мова, також відома як звичайна мова, якою люди спілкуються усно або письмово для загального спілкування. Цей термін виник тому, що коли користувач хоче спілкуватися з комп'ютером, ми не можемо змусити користувачів вчити специфічну для машин мову. Мови можуть бути будь-якими, такими як хінді, французька, англійська, китайська тощо. Також слід зазначати, що мова є системою, набором правил або набором символів.

Основною задачею обробки природної мови є розуміння, де вхідними даними є природна мова, та відслідковування сенсу (контексту) повідомлення.

Багато алгоритмів обробки природної мови (NLP) базуються на машинному навчанні, переважно статистичному. Робота машинного навчання відрізняється від спроб обробки мови. Попередні реалізації завдань обробки мови зазвичай включали безпосереднє ручне кодування великих наборів правил. Замість цього, машинне навчання вимагає використання загального алгоритму навчання для автоматичного вивчення таких правил через аналіз великої кількості типових прикладів з реального світу. Багато різних класів алгоритмів машинного навчання були застосовані до завдань обробки природної мови (NLP). Ці алгоритми приймають на вхід великий набір характеристик, які генеруються вхідної інформації. Деякі раніше створені алгоритми, такі як дерева рішень, генерували системи жорстких правил типу "якщо-то", подібні до систем рукописних правил, які тоді були поширені. Однак, дослідження все більше зосереджуються на статистичних моделях, які приймають м'які, ймовірнісні рішення, базуючись на привласненні дійсних ваг кожному входу. Різні системи, засновані на алгоритмах машинного навчання, мають багато переваг над правилами, створеними вручну, серед яких зведення слів до більш поширених

варіантів, краща стійкість до незнайомих вхідних даних та покращення роботи при збільшенні тренувальних даних.

Моделі на основі поєднання обробки природної мови та машинного навчання окрім перекладу та перефразування можуть бути застосовані для оптичного розпізнавання тексту (коли необхідно розпізнати текст на картинці), розмітці частин мови у реченні, граматичного аналізу та відповіді на запитання.

Розберемо модель mT5, яка буде використана для перекладання.

mT5 – це багатомовна модель, яка створена на основі попередньої моделі T5 та успадковує усі її переваги, серед яких: універсальний текстовий формат та дизайн тексту. Дані для тренування були отримані завдяки публічному вебскануванню «Common Crawl».

T5 – це попередньо навчена мовна модель, основною відмінністю якої є використання єдиного формату «текст-у-текст» для всіх задач обробки природної мови на основі тексту. Цей підхід є природним для генеративних завдань (таких як машинний переклад або абстрактне узагальнення), де формат завдання вимагає від моделі генерувати текст на основі певних вхідних даних. Це більш незвично для завдань класифікації, де T5 навчається видавати буквальні мітки (наприклад, «позитивний» або «негативний» для аналізу настроїв) замість індексу класу. Основна перевага цього підходу полягає в тому, що він дозволяє використовувати точно таку ж навчальну мету (максимальна ймовірність з примусовим навчанням) для кожного завдання, що на практиці означає, що один набір гіперпараметрів може бути використаний для ефективного доопрацювання будь-якого наступного завдання.

Додатковою відмінною рисою T5 є її масштаб, з попередньо навченими моделями розмірами від 60 мільйонів до 11 мільярдів параметрів. Ці моделі були попередньо навчені на приблизно 1 трильйоні токенів даних.

Архітектура моделі та процедура навчання, яка була використана для mT5, дуже схожа на T5. Основним фактором у попередньому навчанні багатомовних моделей є те, як вибирати дані з кожної мови. Зрештою, цей вибір є грою з нульовою сумою. Якщо мови з низькими ресурсами вибираються занадто часто,

модель може перевчитися; якщо мови з високими ресурсами не тренуються достатньо, модель буде недовчена. Тому був застосований підхід з підвищенням ресурсів для менших мов.

Висновки до розділу 1

В даному розділі було проведено постановку задачі, у якій було виділено основний функціонал додатку для перекладу тексту та аудіо. Далі було оглянуто наявні аналоги та їх характеристики. Також було описано та обґрунтовано вибір інструментальних засобів для реалізації додатку та проведення тренування моделі. Останнім кроком був огляд та аналіз наявних методів та алгоритмів машинного навчання, разом з описом характеристик моделі mT5.

РОЗДІЛ 2. АНАЛІЗ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТРЕНУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ

2.1. Аналіз задачі та методів її вирішення

Існуючі системи машинного перекладу вже досягли значного успіху однак, вони не завжди забезпечують високу точність, особливо у випадку мов з обмеженими ресурсами або складних контекстів.

Метою цього розділу є дослідження та реалізація процесу навчання моделі штучного інтелекту для машинного перекладу на основі архітектури mT5 (Multilingual Text-to-Text Transfer Transformer), розробленої компанією Google. Основним завданням є адаптація цієї моделі для перекладу конкретної мовної пари (англійська–українська), що включає попередню обробку даних, навчання моделі, оцінку результатів та аналіз ефективності.

Завдання машинного перекладу належить до класу завдань обробки природної мови (NLP) і, зокрема, до завдань послідовного перетворення вхідного тексту з однієї мови на текст іншою мовою з максимальним збереженням значення.

Генерація природної мови (англ. Natural Language Generation, NLG) [28], є частиною процесу обробки природної мови, яка займається створенням та вивченням додатків, для відтворення інформації у вигляді зрозумілого для людей тексту. Даний процес приймає вхідні дані у різному вигляді (текст, фото, таблиця, тощо.), після підготовки даних та їх аналізу, готує відповідь у найдоречнішому вигляді. Серед реальних прикладів, де цей процес відбувається, можна навести, як відповіді чатботів, так і переклади текстів з одної мови на іншу. Перевагою даного підходу є збереження сенсу речень та структури тексту загалом.

Деякі вчені описують структуру системи, яка спирається на певні правила, чітко розділяючись на 3 рівні. Першим є планування документу, що визначає порядок у якому інформація буде написана [29]. Другим є мікропланування, де генеруються посилання на об'єкти або сутності, разом з вибором слів, які будуть

застосовуватися [30]. Також відбувається зіставлення схожих речень для покращення читабельності. Останній, третій рівень, реалізація, тут генерується фактичний текст, використовуючи реальні правила синтаксису та морфології.

Використання моделі mT5 передбачає доопрацювання великої попередньо навченої моделі на нових даних, пов'язаних з цільовими мовами. Завдання ускладнюється високими вимогами до обчислювальних ресурсів, необхідністю якісної анотації текстового корпусу та потребою у правильному форматуванні вхідних та вихідних даних. На результати може вплинути морфологічна складність української мови, відносно англійської.

В результаті проведення тренування очікується отримання навченої версії моделі mT5, здатної перекладати тексти між зазначеними мовами.

Обрана претренована модель – mT5 small, має менший розмір, ніж базова версія, але швидкість тренування вища. Архітектура моделі базується на seq2seq transformers [31] [32], які розраховані на обробку послідовних даних та звертання уваги на контекст. Також слід зазначити, що модель використовує text-to-text підхід, що зводить усі операції до трансформування вхідного тексту у вихідний.

mT5 була попередньо навчена та датасеті MC4, який представляє собою очищені результати сканування вебсторінок. Він містить понад 100 мов [33].

Під час предтренування використовувався метод «masked span corruption», подібний до того, який був використаний для тренування T5. Його суть полягає у тому, вхідні фрагменти тексту випадково пошкоджуються (видаляються або замінюються на спеціальні токени). Модель у свою чергу повинна відновити ці фрагменти, генеруючи послідовність, яка заповнює пропущені частини. Такий підхід дозволяє моделі навчитися контекстуальному розумінню мови без використання вручну анотованих даних.

У свою чергу тренування, яким ми будемо займатися називається: тонке налаштування. Його суть полягає у тому, що тепер ми подаємо текст, як генеративну задачу, таким чином модель вчиться перекладати на необхідні нам мови.

Під час перекладу текст у початковій мові подається на вхід разом із спеціальним префіксом і кодується векторним представленням за допомогою енкодера. Декодер генерує відповідь по одному токenu, використовуючи як закодовану інформацію, так і вже згенеровану частину перекладу.

Для тренування моделі буде використаний датасет OPUS-100 [34]. Він містить у собі пари речень, де кожна пара містить англійську та іншомовну версію. У нашому випадку, ми використаємо англійсько-українську версію.

Для проведення тренування буде використана мова Python та фреймворк для роботи з моделями Hugging Face Transformers, та додатково буде використано PyTorch, для налаштування моделі та можливості виконання процесу тренування на відеокарті, що для моделі такого типу є обов'язковим, оскільки інакше весь процес займе занадто багато часу. Для зручного використання датасету буде використано бібліотеку Datasets. Також слід зазначити, для вдалого виконання тренування на відеокарті потрібно встановити спеціальні драйвера та набір інструментів, в даному випадку це Cuda від Nvidia. Як альтернативу можна використати Google Colaboratory.

Першим кроком є розробка функцій для кодування даних, оскільки модель приймає данні у вигляді токенів, з цим допоможе AutoTokenizer, це частина моделі, яка зберігає у собі налаштування токенізації, які модель отримала при навчанні.

Далі потрібно розробити додаткові функції для тренування, а саме перетворення даних, збір їх у купу та перемішування мовних пар, в цьому допомагає NumPy.

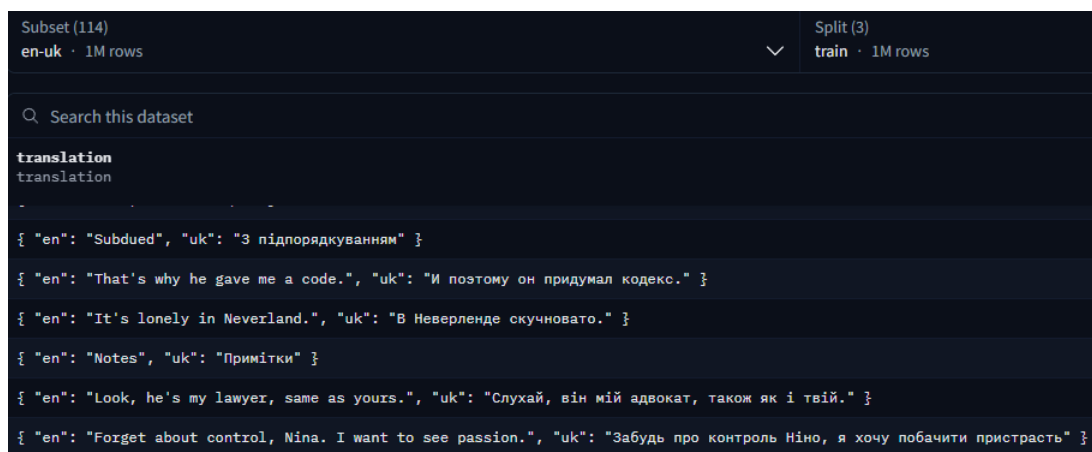
Після підготовки необхідного функціоналу, потрібно розробити алгоритм тренування, який підтримує контрольні точки, зберігає модель кожну 1000 кроків та підраховує втрати (losses).

Останнім кроком є будування графіку втрат, для візуального аналізу процесу тренування.

2.2. Характеристика джерела даних та розробка алгоритму

Обраний датасет – OPUS-100 [35]. Він побудований спеціально для навчання мовних моделей перекладу, де англійська мова використовується, як центральна (модель MT5 використовує англійську мову, як центральну). Датасет містить багато підмножин, які розділяють його на мовні пари. Такий тип представлення даних називається: паралельні пари.

Оскільки даний набір даних розташований на сайті `huggingface`, ми можемо використати бібліотеку `datasets` для того, щоб завантажити лише необхідну нам пару, у даному випадку це англійська-українська. Набір даних містить 1 мільйон рядків, у кожному вказана англійська версія тексту та українська. Дані представлені у форматі словників з двома ключами: «translation», кодом мови «en» або «uk» (рис. 2.1).



Subset (114)	Split (3)
en-uk · 1M rows	train · 1M rows
Search this dataset	
translation	
translation	
-	
{ "en": "Subdued", "uk": "З підпорядкуванням" }	
{ "en": "That's why he gave me a code.", "uk": "И поэтому он придумал кодео." }	
{ "en": "It's lonely in Neverland.", "uk": "В Неверленде скучновато." }	
{ "en": "Notes", "uk": "Примітки" }	
{ "en": "Look, he's my lawyer, same as yours.", "uk": "Слухай, він мій адвокат, також як і твій." }	
{ "en": "Forget about control, Nina. I want to see passion.", "uk": "Забудь про контроль Ніно, я хочу побачити пристрасть" }	

Рис. 2.1 – Структура даних

Також даний датасет має додаткові структури для валідації та тренування, кожен з них складається з 2 тисяч рядків (рис. 2.2).



Split (3)
train · 1M rows
Split (3)
train (1M rows)
validation (2k rows)
test (2k rows)

Рис. 2.2 – Додаткові структури

Однак, під час аналізу датасету, було помічено, що він має доволі велику кількість рядків, де українська частина була іншою мовою, частіше всього

російською. Пов'язано це в основному, с тим що датасет був створений на основі сканування вебсторінок, тому деякі повідомлення з форумів, містять російську мову (рис. 2.3)

```
{ "en": "How do you know that Amy's the head cheerleader?", "uk": "Откуда ты знаешь, что Эми - главная чирлидерша?" }  
{ "en": "Good job.", "uk": "Хорошая работа." }
```

Рис. 2.3 – Приклад неякісних даних

Для очистки набору даних було використано бібліотеку langdetect[36]. За допомогою цієї бібліотеки, ми можемо відфільтрувати данні, залишаючи рядки, де мова була відмічена, як українська. Після очистки залишилося близько 470000 рядків, цього більш ніж достатньо для тренування.

Серед переваг використання такого датасету є простота, оскільки можна використати офіційні інструменти для роботи з ним. Також цей набір даних охоплює велику кількість різножанрових текстів, що позитивно впливає на загальну можливість перекладу моделі текстів за різною тематикою.

Для тренування було використано 150000 рядків, оскільки тренування навіть на такій кількості даних займе дуже багато часу. Загальний розмір дорівнює 56 MB.

Спочатку імпортуємо усі необхідні бібліотеки та модулі, також вказуємо посилання на репозиторій моделі, шлях для зберігання та максимальну довжину послідовності токенів.

```
from datasets import load_dataset  
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer  
from transformers import get_linear_schedule_with_warmup  
import matplotlib.pyplot as plt  
import numpy as np  
import seaborn as sns  
import torch  
from torch import optim  
from torch.nn import functional as F  
from tqdm import tqdm_notebook  
  
model_repo = 'google/mt5-small'  
model_path = 'D:/curse/model/'  
max_seq_len = 40
```

Далі підключаємо автоматичний токенізатор та модель, переводимо режим обробки на відеокарту, та вказуємо максимальну довжину послідовності.

Токенізатор автоматично завантажить налаштування, які були збережені під час першого створення моделі, її розробниками.

```
tokenizer = AutoTokenizer.from_pretrained(model_repo,
cache_dir='D:/curse/model/cache/')
model = AutoModelForSeq2SeqLM.from_pretrained(model_repo,
cache_dir='D:/curse/model/cache/')
model = model.cuda()
model.config.max_length = 40
```

Далі підключаємо датасет та розбиваємо його на секції для тренування та тестування. Частина, яка буде використана для тренування, обмежуємо до 150 тис. рядків, цього вистачить для проведення навчання та дозволить скоротити часові затрати.

```
train_dataset = datasets.load_from_disk('dataset/mydata')
train_dataset = train_dataset.select(range(150000))
test_dataset = datasets.load_from_disk('dataset/mytestdata')
```

Наступним кроком буде додавання нових спеціальних токенів. Оскільки модель не має можливості зрозуміти, у яку мову ми намагаємося перевести текст, на потрібно додати спеціальні вказівки, вони будуть розміщатися та початку повідомлення, так як модель побудована з підходом до звертання уваги, дивлячись на токен, вона буде розуміти, на яку мову їй перекладати текст.

```
LANG_TOKEN_MAPPING = {
    'en': '<en>',
    'uk': '<uk>'
}

special_tokens_dict = {'additional_special_tokens':
list(LANG_TOKEN_MAPPING.values())}
tokenizer.add_special_tokens(special_tokens_dict)
model.resize_token_embeddings(len(tokenizer))
```

Далі опишемо функцію, яка буде приймати вхідний рядок, додавати до нього спеціальний токен, та перетворювати слова у токени. Важливим є параметр `truncation`, він вказує, що якщо довжина послідовності токенів, не дорівнює 40, то потрібно звести послідовність до цієї довжини, якщо рядок занадто довгий, то його обріже, якщо занадто короткий, то вставить пусті токени.

```
def encode_input_str(text, target_lang, tokenizer, seq_len,
                    lang_token_map=LANG_TOKEN_MAPPING):
    target_lang_token = lang_token_map[target_lang]

    input_ids = tokenizer.encode(
```

```

        text=target_lang_token + text,
        return_tensors='pt',
        padding='max_length',
        truncation=True,
        max_length=seq_len)
    return input_ids[0]

```

Також описуємо схожу функцію для вихідного тексту, але не додаємо спеціальний токен, оскільки він нам не потрібен, при отриманні відповіді від моделі.

```

def encode_target_str(text, tokenizer, seq_len,
                      lang_token_map=LANG_TOKEN_MAPPING):
    token_ids = tokenizer.encode(
        text=text,
        return_tensors='pt',
        padding='max_length',
        truncation=True,
        max_length=seq_len)

    return token_ids[0]

```

Далі описуємо важливу для тренування функцію, яка буде формувати данні для тренування. Вона приймає рядок з набору даних, випадково перемішує мови, та вказує, що використати для тренування, що як результат тренування. Обидва рядки пропускаються, через описані функції, які перетворюють їх на токени.

```

def format_translation_data(translations, lang_token_map,
                           tokenizer, seq_len=128):
    langs = list(lang_token_map.keys())
    input_lang, target_lang = np.random.choice(langs, size=2, replace=False)
    input_text = translations[input_lang]
    target_text = translations[target_lang]

    if input_text is None or target_text is None:
        return None
    input_token_ids = encode_input_str(
        input_text, target_lang, tokenizer, seq_len, lang_token_map)
    target_token_ids = encode_target_str(
        target_text, tokenizer, seq_len, lang_token_map)
    return input_token_ids, target_token_ids

```

Для пришвидшення тренування, данні у модель будуть загрузатися купами (batch). Тому описуємо функцію, яка буде підготовлювати купи даних, пропускаючи їх через описану вище функцію. Також це перше місце, де використовується обробка на відеокарті, підготовка даних, таким чином значно прискорюється.

```
def transform_batch(batch, lang_token_map, tokenizer):
    inputs = []
    targets = []
    for translation_set in batch['translation']:
        formatted_data = format_translation_data(translation_set,
lang_token_map, tokenizer, max_seq_len)
        if formatted_data is None:
            continue
        input_ids, target_ids = formatted_data
        inputs.append(input_ids.unsqueeze(0))
        targets.append(target_ids.unsqueeze(0))
    batch_input_ids = torch.cat(inputs).cuda()
    batch_target_ids = torch.cat(targets).cuda()

    return batch_input_ids, batch_target_ids
```

Далі використаємо особливість мови Python, опишемо генератор даних, якій, при звернені, буде видавати нову купу даних, таким чином дозволяючи проходити по датасету купами, які автоматично готуються до тренування моделі.

```
def get_data_generator(dataset, lang_token_map, tokenizer, batch_size=32):
    dataset = dataset.shuffle()
    for i in range(0, len(dataset), batch_size):
        raw_batch = dataset[i:i + batch_size]
        yield transform_batch(raw_batch, lang_token_map, tokenizer)
```

Також було проведено дослідження впливу значення змінної lr (швидкість навчання) на процес навчання. Оскільки ця змінна повинна окремо підбиратися для кожної моделі та її задач, зазвичай використовуються значення між $1e^{-4}$ до $7e^{-4}$. Значення $1e^{-4}$ видає занадто швидкий темп навчання, оптимізатор робить занадто великі кроки, це відображено швидким спадом втрат на графіку, що негативно впливає на засвоєння даних (рис. 2.4).

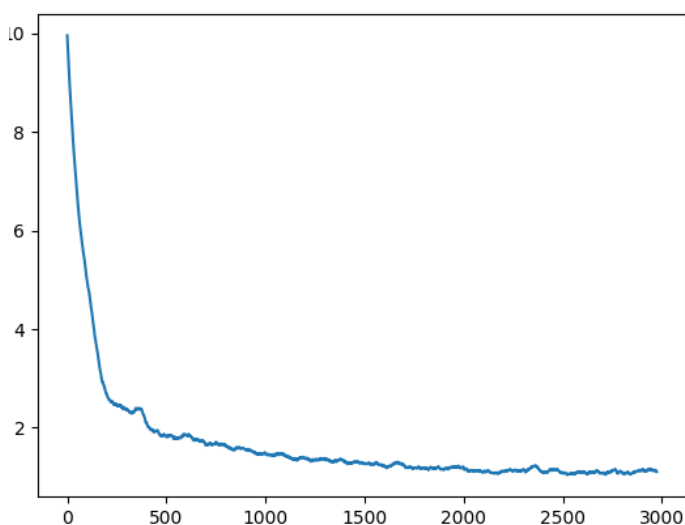


Рис. 2.4 – Тестування $lr=1e^{-4}$

Значення $3e^{-4}$ має кращу динаміку на графіку, але початковий спад та зміна швидкості навчання відбувається занадто швидко, це вказує занадто швидку конвергенцію (рис. 2.5).

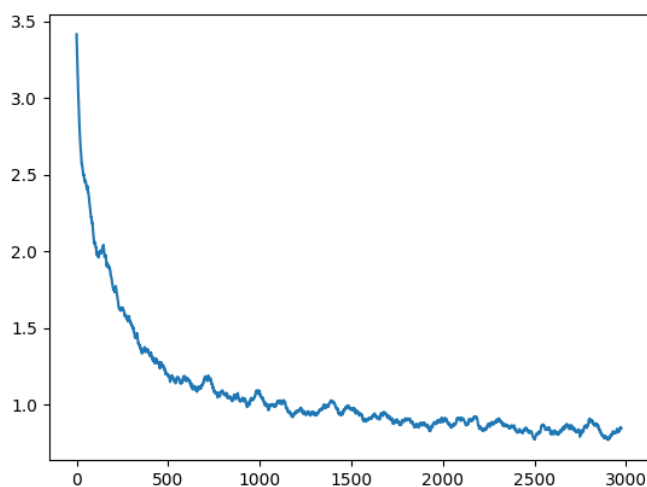


Рис. 2.5 – Тестування $lr=3e^{-4}$

Наступним, було досліджено значення $5e^{-4}$, графік має плавний спад та зміну швидкості навчання, конвергенція відбувається плавно та з прийнятною швидкістю (рис. 2.6).

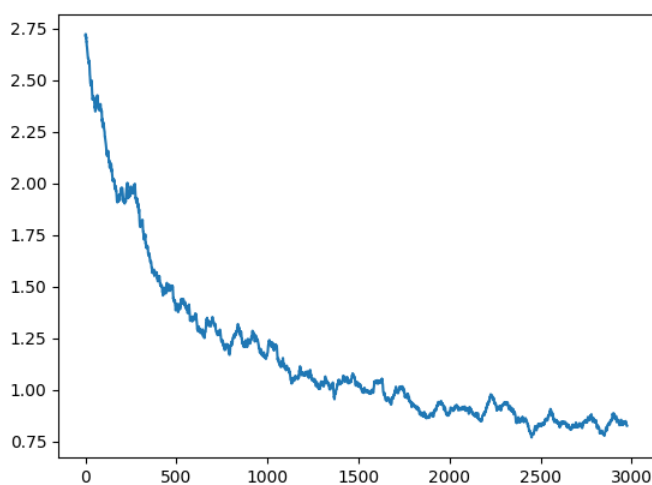


Рис. 2.6 – Тестування $lr=5e^{-4}$

У свою чергу, значення $7e^{-4}$, навпаки, показує занадто повільний спад втрат, що призводить до необхідності довшого навчання для досягнення бажаного результату тренування. Також графік є дуже нестабільним, це є результатом збіжності моделі до неоптимальних рішень (рис. 2.7).

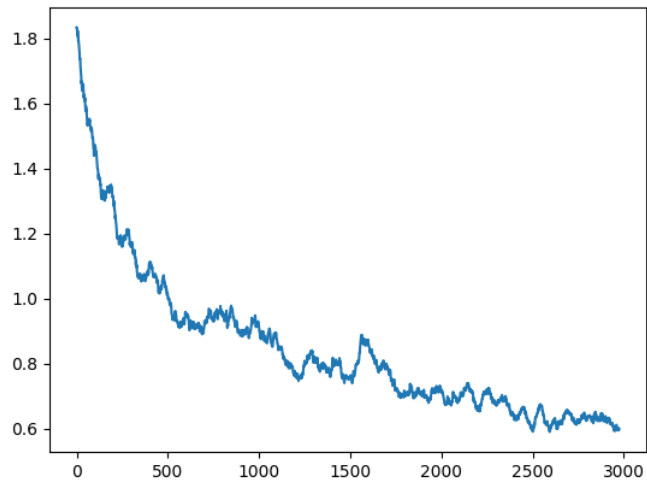


Рис. 2.7 – Тестування $lr=7e^{-4}$

Результати дослідження вказують на те, що найкращим значенням для цієї моделі є $5e^{-4}$.

Додатково, перед налаштуванням алгоритму, було проведено дослідження впливу розміру купи на процес тренування. Досліджувалися розміри 8, 16, та 32, на 10000 рядків.

Розмір купи у 8 рядків, надає гарні результати тренування, але займає багато часу (рис. 2.8).

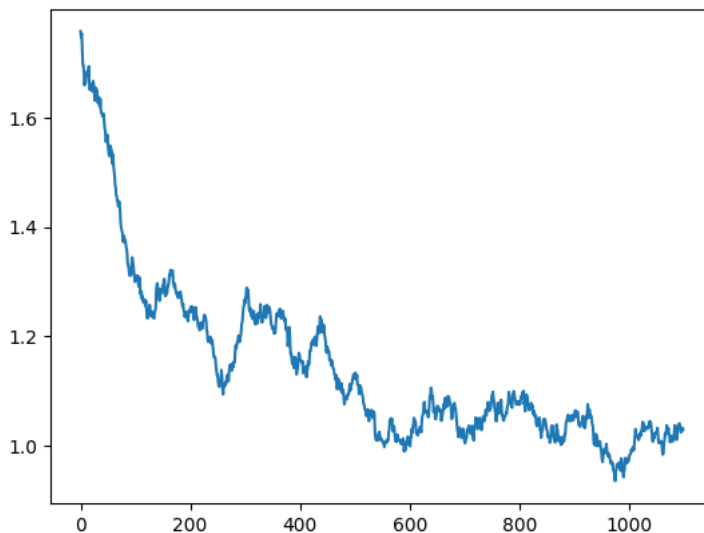


Рис. 2.8 – Тестування $batch=8$

Значення у 16, показує набагато швидші та стабільніші результати, але помилка зросла до 1,869, що є очікуваним наслідком збільшення розміру батча (рис. 2.8).

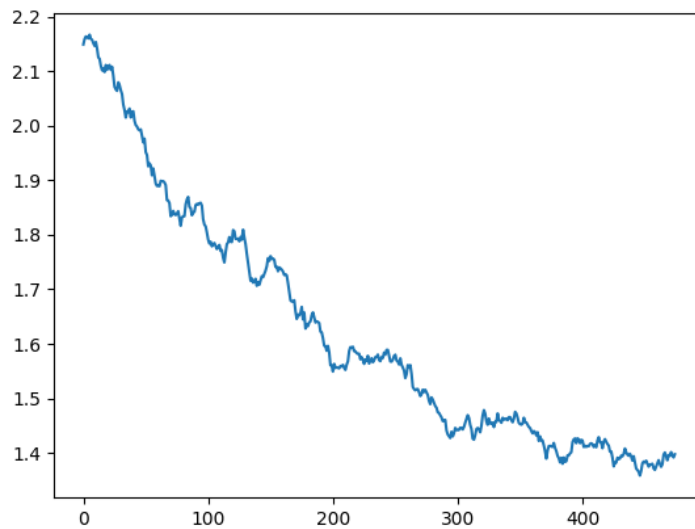


Рис. 2.8 – Тестування batch=16

Останнім перевірялося значення 32, його результати є повільнішими, ніж 16, але швидшими, ніж 8. Однак помилка є найбільшою з усіх протестованих значень (рис. 2.9).

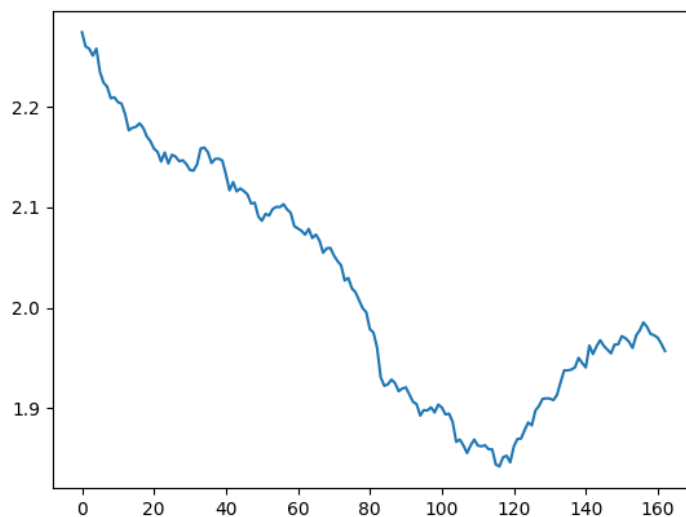


Рис. 2.9 – Тестування batch=32

На результати впливають не лише зміна параметрів, а й потужності комп'ютеру, на спеціально підготовлених для тренування ШІ машинах, значення купи у 32 рядки, може видавати найшвидші результати, а погана якість тренування може бути компенсована більшим набором даних, та збільшеною кількістю епох.

Для кращого розуміння результатів дослідження, було побудовано таблицю (табл 2.1).

Результати дослідження впливу розміру купи

Розмір купи	Втрата при тестуванні	Затрачений час (сек)
8	1,172	236
16	1,869	157
32	2,148	223

Обраним розміром купи для подальшого навчання моделі є розмір у 16 рядків, він надає найкращі результати у розрізі якості та часу. Понижена точність перекладу скомпенсована більшим розміром набору даних.

Далі задаємо усі необхідні параметри, кількість епох (5), розмір купи (16), частоту виводу результатів у консоль, швидкість навчання, частоту зберігання проміжкових результатів, розігрів моделі [37], також підключаємо оптимізатор типу Adam [38], та допоміжний планувальник.

```
n_epochs = 5
batch_size = 16
print_freq = 100
lr = 5e-4
n_batches = int(np.ceil(len(train_dataset) / batch_size))
total_steps = n_epochs * n_batches
checkpoint_freq = 1000
n_warmup_stes = int(total_steps * 0.01)
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
scheduler = get_linear_schedule_with_warmup(optimizer, n_warmup_stes,
total_steps)
```

Додатково опишемо функцію для оцінки моделі, вона буде викликатися кожен контрольну точку, та перевіряти результати навчання.

```
def eval_model(model, gdataset, max_iters=8):
    test_generator = get_data_generator(gdataset, LANG_TOKEN_MAPPING,
                                       tokenizer, batch_size)

    eval_losses = []
    for i, (input_batch, label_batch) in enumerate(test_generator):
        if i >= max_iters:
            break

        model_out = model.forward(
            input_ids=input_batch,
            labels=label_batch)
        eval_losses.append(model_out.loss.item())

    return np.mean(eval_losses)
```

Нарешті описуємо головний цикл тренування, він буде робити ітерації у датасеті, отримувати відповідні купи даних для тренування, після чого дані будуть подаватися у модель, оптимізатор та планувальник контролюють процес навчання. Також при досягненні контрольних точок, оцінюємо та зберігаємо модель.

```
losses = []
for epoch_idx in range(n_epochs):
    data_generator = get_data_generator(train_dataset, LANG_TOKEN_MAPPING,
                                       tokenizer, batch_size)

    for batch_idx, (input_batch, label_batch) in
        tqdm_notebook(enumerate(data_generator), total=n_batches):
        optimizer.zero_grad()
        model_out = model.forward(input_ids=input_batch, labels=label_batch)
        loss = model_out.loss
        losses.append(loss.item())
        loss.backward()
        optimizer.step()
        scheduler.step()

        if (batch_idx + 1) % print_freq == 0:
            avg_loss = np.mean(losses[-print_freq:])
            print('Epoch: {} | Step: {} | Avg. loss: {:.3f} | lr:
                {}'.format(epoch_idx + 1, batch_idx + 1, avg_loss,
                    scheduler.get_lr()[0]))

            if (batch_idx + 1) % checkpoint_freq == 0:
                test_loss = eval_model(model, test_dataset)
                print('Saving model with test loss of {:.3f}'.format(test_loss))
                model.save_pretrained(model_path)
                tokenizer.save_pretrained(model_path)

    test_loss = eval_model(model, test_dataset)
    print('Test loss of {:.3f}'.format(test_loss))
```

Завдяки тому, що втрати зберігалися при кожному кроці тренування, ми можемо відобразити їх у вигляді графіку, для наглядного аналізу результатів тренування.

```
window_size = 50
smoothed_losses = []
for i in range(len(losses)-window_size):
    smoothed_losses.append(np.mean(losses[i:i+window_size]))

plt.plot(smoothed_losses[100:])
plt.show()
```

Ще слід розібрати алгоритм використання моделі для перекладу тексту. Після завантаження моделі, використовуючи посилання на локальний файли, нам

все ще потрібно описати функцію для переведення тексту у токени, з додаванням спеціального токена мови.

```
model_path = './model/'
max_seq_len = 40

LANG_TOKEN_MAPPING = {
    'en': '<en>',
    'uk': '<uk>'
}

tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForSeq2SeqLM.from_pretrained(model_path)
model = model.cuda()

model.config.max_length = 40
def encode_input_str(text, target_lang, tokenizer, seq_len,
                    lang_token_map=LANG_TOKEN_MAPPING):
    target_lang_token = lang_token_map[target_lang]

    input_ids = tokenizer.encode(
        text=target_lang_token + text,
        return_tensors='pt',
        padding='max_length',
        truncation=True,
        max_length=seq_len)

    return input_ids[0]
```

Далі ми можемо використати будь яке речення, для перекладу, використовуючи описану функцію. Але у цей час для отримання відповіді, ми використовуємо функцію `model.generate()`, яка приймає у себе рядок токенів, точність перекладу та кількість варіантів перекладу. Як було описано вище, модель є генеративною, тобто результатом перекладу є згенерований рядок, на основі вхідних даних. Саме цьому процесу генерації ми навчали модель.

```
test_sentence = "Привіт, моє ім'я Артем і я люблю програмування."
print('Raw input text:', test_sentence)

input_ids = encode_input_str(
    text=test_sentence,
    target_lang='en',
    tokenizer=tokenizer,
    seq_len=model.config.max_length,
    lang_token_map=LANG_TOKEN_MAPPING)
input_ids = input_ids.unsqueeze(0).cuda()

print('Truncated input text:', tokenizer.convert_tokens_to_string(
    tokenizer.convert_ids_to_tokens(input_ids[0])))

output_tokens = model.generate(input_ids, num_beams=20, num_return_sequences=3)
for token_set in output_tokens:
    print(tokenizer.decode(token_set, skip_special_tokens=True))
```

Таким чином виконуються переклади з будь якої мови, яку ми використовували при тренування, лише необхідно змінювати налаштування токенизатора, який буде додавати відповідний токен цільової мови перекладу.

2.3. Аналіз проведеного тренування моделі для перекладу тексту

Оглянемо результати тренування. Спочатку модель видає доволі великі втрати, при високій швидкості навчання, що є доволі очікуваним, за швидкість навчання відповідає оптимізатор, з використанням якого кінцевий результат покращується.

У кінці навчання, результат значно змінюється, втрати набагато нижче, швидкість навчання повільніша, що вказує на потенційне перенавчання, при експериментах перед фінальним тренуванням, було помічено, що при 8 епохах, модель перенавчається, тому кількість була знижена до 5 епох.

Також оглянемо візуальне представлення втрат відносно до кроків у вигляді графіку (рис. 2.10).

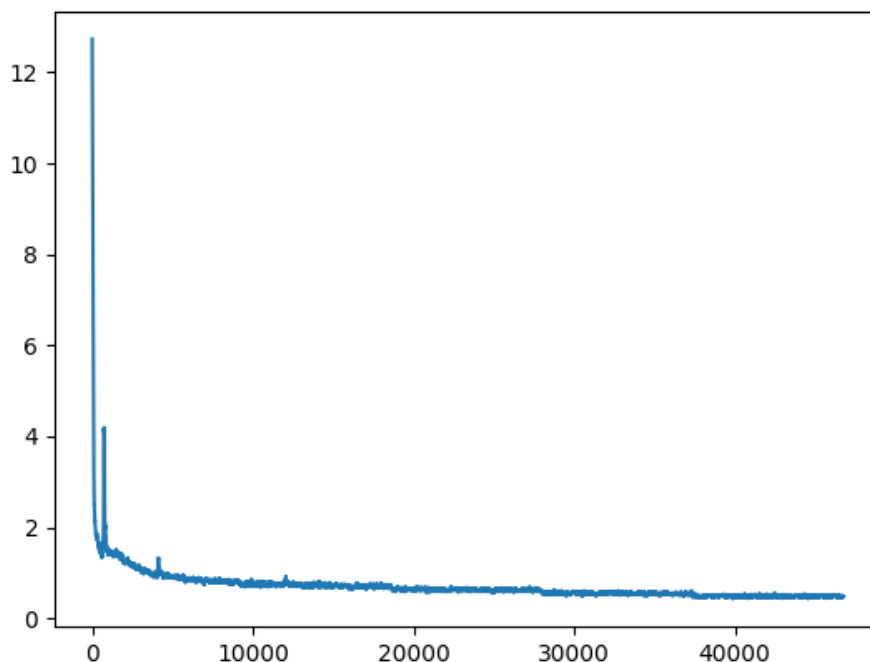


Рис. 2.10 – Графік втрат

На графіку можна побачити, що спочатку втрати доволі швидко зменшувалися, при досягненні кроку у 10000, графік показує плато, що вказує на успішне навчання, але втрати все ще є занадто високими, тому було навчання було продовжено, для усунення помилок та кращого запам'ятовування датасету [39].

Спробуємо перекласти декілька речень, спочатку перекладемо речення «Це тестове повідомлення.» на англійську мову (рис. 2.11).

```
Raw input text: Це тестове повідомлення.  
Truncated input text: <en> Це тестове повідомлення.</s>  
This is a test message.  
This is a test notification.  
This is the test message.
```

Рис. 2.11 – Переклад з української на англійську

На результатах можна побачити, що в усіх варіантах перекладу, сенс повідомлення був збережений, але контекст трішки відрізняється, це відбувається через спеціальні налаштування, які дозволяють побачити декілька варіантів перекладу.

Також пробуємо перекласти дане речення, з англійської на українську (рис. 2.12).

```
Raw input text: This is a test message.  
Truncated input text: <uk> This is a test message.</s>  
Це тестове повідомлення.  
Це тестне повідомлення. Це тестне повідомлення.  
Це тестне повідомлення.
```

Рис. 2.12 – Переклад з англійської на українську

У даному випадку лише перший переклад є коректним, інші містять помилки. Оскільки під час звичайної роботи ми будемо бачити лише перший варіант перекладу (найкоректніший), то це не є проблемою, але це вказує на потенційні помилки у наборі даних, або його недостатній розмір.

Спробуємо перекласти інше речення на українську (Рис. 2.13).

```
Raw input text: Do you like to play computer games?  
Truncated input text: <uk> Do you like to play computer games?</s>  
Вам подобається грати у комп'ютерні ігри?  
Вам подобається грати в комп'ютерні ігри?  
Для комп'ютерних ігор тобі подобається грати у комп'ютерні ігри?
```

Рис. 2.13 – Переклад іншого речення на українську

Знову непогані результати перекладу. Останній варіант перекладу, занадто відійшов від основного контексту, тому видав нісенітницю.

У результаті тренування, ми отримали модель, яка здатна добре перекладати з української на англійську, а ось навпаки переклад вийшов не такий точний, частково пов'язано це з архітектурою моделі MT5, оскільки центральною мовою є англійська, модель не дуже добре перекладає з англійської на українську. Також артефакти при перекладі можуть бути пов'язані з вибором маленької версії моделі, вона містить менше токенів, тому не завжди зв'язки створюються правильно. Одним з варіантів буде перехід до базової моделі, як результат збільшиться розмір та час навчання, але повинні покращитись результати.

На переклад з англійської на українську, ще впливає складність української мови з морфологічної точки зору, формування правильних частин мови та речень є складнішою у цьому напрямку.

Графік втрат свідчить про успішне навчання моделі на тренувальних даних, але результат перекладу залежить від напрямку перекладу, модель краще перекладає з української на англійську. Для того щоб покращити результати доцільно: використати більш об'ємну версію моделі, збільшити розмір тренувальних даних. Також дана модель була збережена у форматі SafeTensors, що дозволяє довчити дану модель, на більш спеціалізованому україномовному наборі даних. Додатково рекомендується дослідити інтеграцію з методом параметр-ефективного донавчання LoRA. Це може покращити час навчання при обмежених ресурсах та позитивно вплинути на якість перекладу.

Висновки до розділу 2

В даному розділі було проведено аналіз задачі та описано методи необхідні для її вирішення. Далі було описано огляд та очистку тренувального набору даних, після чого було розроблено алгоритм для проведення машинного навчання. Під час розробки алгоритму було проведено дослідження впливу його параметрів на результати навчання. Також було проаналізовано результати тренування моделі mT5.

РОЗДІЛ 3. РОЗРОБКА ТА ОПИС ДОДАТКУ

3.1. Проектування додатку

При використанні додатку для перекладу тексту та аудіо за допомогою штучного інтелекту, користувачі мають доступ до наступних функцій:

1. Переклад тексту, який вводиться у відповідне поле, на іншу мову.
2. Переклад тексту з файлу (підтримуються формати txt, doc, docx).
3. Переклад аудіо з файлу (підтримуються формати wav та mp3).
4. Переклад аудіо з мікрофону (запис відбувається у інтерфейсі додатку, після чого надсилається до серверу для обробки).
5. Транскрипція аудіо з файлу.
6. Завантаження файлу з результатами перекладу.

Візуалізуємо можливості користувача за допомогою діаграми варіантів використання (рис. 3.1).

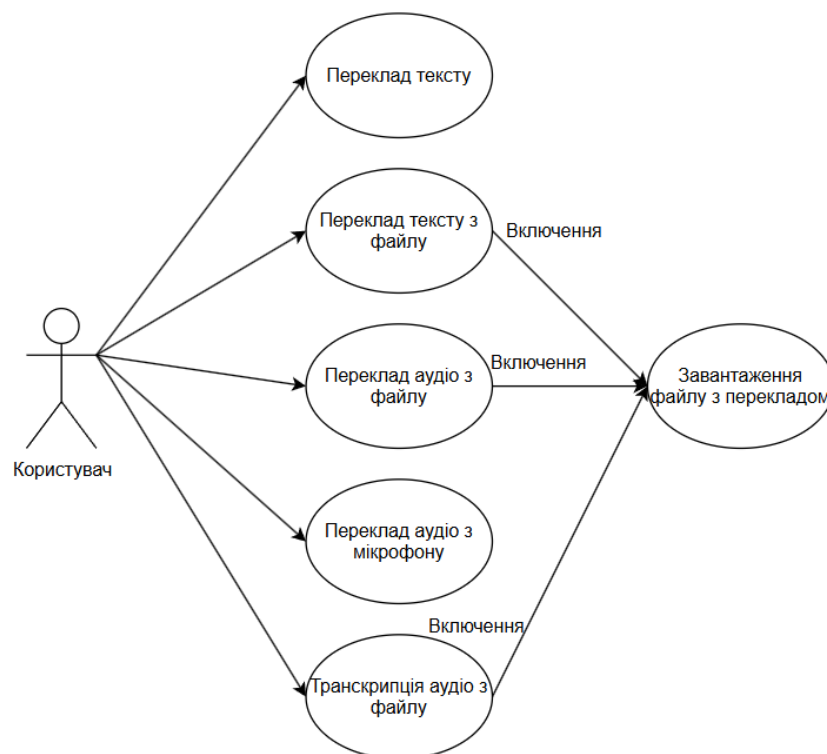


Рис. 3.1 – Діаграма варіантів використання

Для розуміння взаємодії компонентів у додатку для перекладу тексту та аудіо, було розроблено діаграму компонентів, яка відображає у собі залежності та зв'язки між різними частинами системи (рис. 3.2).

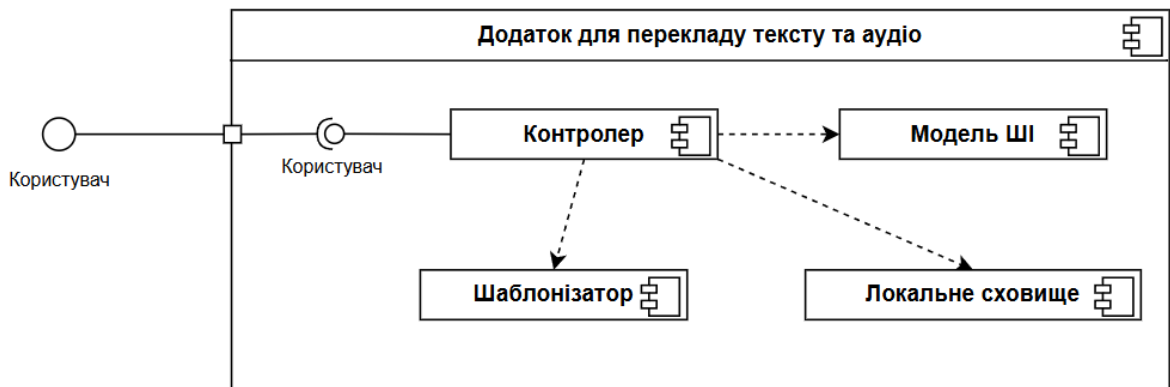


Рис. 3.2 – Діаграма компонентів

Користувач є основним джерелом даних, він використовує додаток за допомогою інтерфейс «Користувач», який передає данні на обробку контролеру. Контролер є основним компонентом, який вирішує, що робити з цими даними. Коли потрібно зберегти або створити файл, контролер звертається до локального сховища. Якщо контролеру потрібно перекласти текст, він звертається до моделі штучного інтелекту, яка виконує переклад. Шаблонізатор, у свою чергу, відповідає за відображення сторінок та наповнення їх даними.

Також було створено діаграму розгортання, вона візуалізує фізичні елементи системи, які задіяні під час роботи додатку (рис. 3.3).

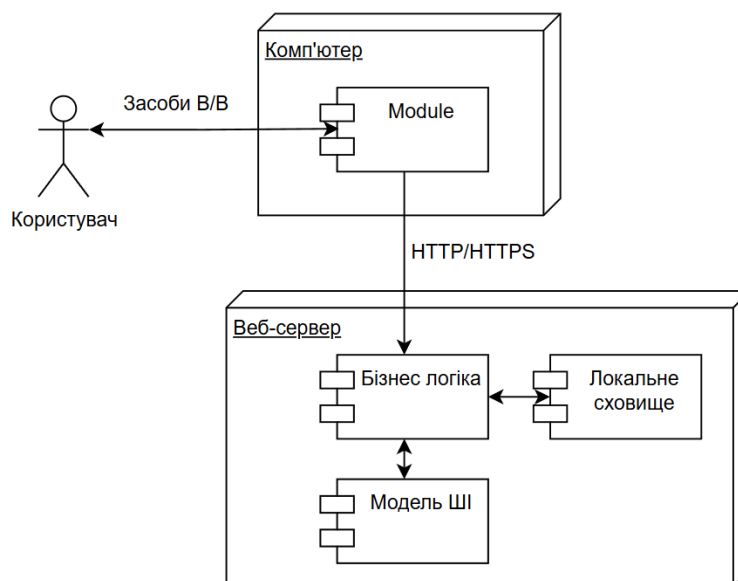


Рис. 3.3 – Діаграма розгортання

3.2. Огляд інтерфейсу

При відкритті головної сторінки додатку, користувач бачить форму для перекладу тексту, який можна ввести у відповідне поле, також список з вибором, на яку мову потрібно виконати переклад. Зверху сторінки знаходиться меню, для зручного переходу до інших варіантів перекладу (рис. 3.4).

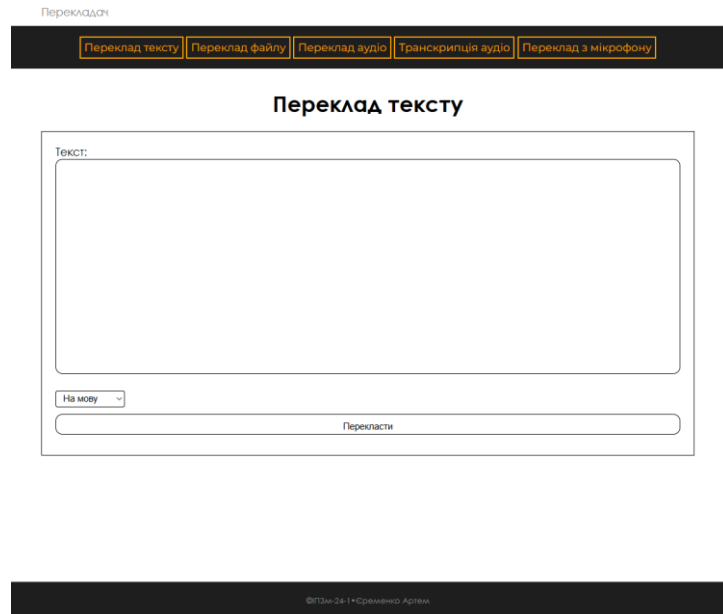


Рис. 3.4 – Головна сторінка

Після введення тексту, та вибору мови, відображається сторінка з результатами перекладу. Усі дані відображаються асинхронно, що дозволяє уникнути зависань сторінок під час перекладу великих обсягів тексту, під час обробки даних користувачу відображається анімований елемент, що сигналізує про коректну роботу додатка. (рис. 3.5).

Результат перекладу

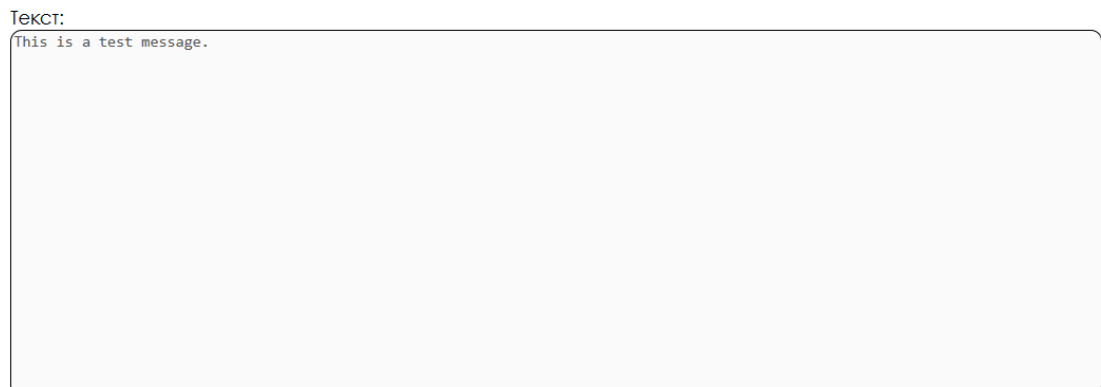


Рис. 3.5 – Сторінка з результатом перекладу

При переході на сторінку «Переклад файлу», користувач потрапляє на спеціалізований інтерфейс, призначений для роботи з документами. На цій сторінці передбачено зручний та інтуїтивно зрозумілий механізм завантаження файлу. При натисканні кнопки обрати файл, відкривається стандартне системне вікно вибору файлів. (рис. 3.6).

Перекладач

Переклад тексту | **Переклад файлу** | Переклад аудіо | Транскрипція аудіо | Переклад з мікрофону

Переклад файлу

Файл:

Choose File No file chosen

На мову ▾

Завантажити

Рис. 3.6 – Сторінка перекладу файлу

Після надсилання форми та обробки файлу, користувачу надається посилання для завантаження вихідного файлу, який буде у тому ж форматі, що й надісланий користувачем документ. Посилання оформлене у вигляді великої кнопки з текстом «Завантажити файл», після натискання відкривається у нова вкладка, яка автоматично закривається після початку завантаження (рис. 3.7).

Перекладач

Переклад тексту | Переклад файлу | Переклад аудіо | Транскрипція аудіо | Переклад з мікрофону

Завантаження файлу

Завантажити файл

Рис. 3.7 – Сторінка завантаження файлу

При переході на сторінку «Переклад аудіо», відкривається форма, у яку можна завантажити аудіо-файл, обрати мову з якої буде виконаний переклад, та на яку буде перекладено. Вибір початкової мови пов'язан з потребою налаштування алгоритму транскрипції (рис. 3.8).

Переклад аудіо

Рис. 3.8 – Сторінка перекладу аудіо

Після відправки форми, користувач також отримує посилання на завантаження файлу з перекладом.

При відкритті сторінки «Транскрипція аудіо», користувач бачить форму для надсилення файлу та вибору мови аудіо. Реалізовано підтримку форматів .wav та .mp3, при перегляді системного меню для відсилення файлу, усі інші формати є прихованими, це дозволяє уникнути помилок при обробці на стороні серверу та спрощує пошук для користувачів (рис. 3.9).

Транскрипція аудіо

Рис. 3.9 – Транскрипція аудіо

Після надсилення файлу, користувач отримує посилання для завантаження файлу.

При переході на сторінку «Переклад з мікрофону», користувач бачить вибір початкової та цільової мови, а також кнопку «Записати». Через те, що запис відбувається за допомогою модулю браузеру та JavaScript, під назвою

MediaRecorder [40], ми маємо можливість відразу надіслати файл до серверу, а при отриманні відповіді вивести його на сторінку (рис. 3.10).

The screenshot shows the desktop version of the 'Переклад з мікрофону' (Microphone Translation) interface. At the top, there is a dark navigation bar with five buttons: 'Переклад тексту' (Text Translation), 'Переклад файлу' (File Translation), 'Переклад аудіо' (Audio Translation), 'Транскрипція аудіо' (Audio Transcription), and 'Переклад з мікрофону' (Microphone Translation). Below the navigation bar, the title 'Переклад з мікрофону' is centered. The main content area contains two dropdown menus for language selection, labeled 'Українська' (Ukrainian) and 'Англійська' (English). Below these is a red-outlined button labeled 'Записати' (Record). At the bottom, there is a large text area with the placeholder text 'This is a test message.'

Рис. 3.10 – Переклад аудіо з мікрофону

Для зручного користування додатком з мобільного телефону, інтерфейс був адаптований під менші розміри екрану. Меню навігації реалізовано у вигляді вертикального списку, що дозволяє зручно обирати потрібний функціонал. Також усі форми та вміст сторінок автоматично змінюють свою ширину, відносно до розмірів екрану користувача (рис. 3.11).

The screenshot shows the mobile version of the 'Переклад з мікрофону' (Microphone Translation) interface. At the top, there is a dark navigation bar with five buttons: 'Переклад тексту' (Text Translation), 'Переклад файлу' (File Translation), 'Переклад аудіо' (Audio Translation), 'Транскрипція аудіо' (Audio Transcription), and 'Переклад з мікрофону' (Microphone Translation). Below the navigation bar, the title 'Переклад з мікрофону' is centered. The main content area contains two dropdown menus for language selection, labeled 'Українська' (Ukrainian) and 'Англійська' (English). Below these is a red-outlined button labeled 'Записати' (Record). At the bottom, there is a large text area with the placeholder text 'This is a test message.'

Рис. 3.11 – Вигляд інтерфейсу на мобільному пристрої

Висновки до розділу 3

В даному розділі було описано процес проектування додатку для перекладу тексту та аудіо на основі штучного інтелекту. Під час проектування було розроблено діаграми варіантів використання, компонентів та розгортання, що допомогло візуалізувати його будову. Також було оглянуто інтерфейс додатку, разом з його функціональними можливостями.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи освітнього ступеня «магістр», було розроблено додаток для перекладу тексту та аудіо на основі штучного інтелекту. Під час розробки, було виконано дослідження алгоритмів та методів машинного навчання на прикладі моделі mT5. У ході цього дослідження було розроблено та оптимізовано алгоритм тренування та виявлено, що правильна оптимізація гіперпараметрів значно підвищує продуктивність алгоритму. Зокрема налаштування змінної швидкості навчання, значення у $5e^{-4}$ забезпечує швидшу конвергенцію та зменшує ризик перетренування, дозволяючи моделі досягати вищої точності. Іншою важливою оптимізацією є генерація даних у вигляді куп. Використання розміру у 16 рядків, не тільки оптимізує використання обчислювальних ресурсів, але й покращує узагальнення моделі за рахунок стохастичного градієнтного спуску, що зменшує шум та прискорює ітерації тренування.

Отримані результати мають практичне значення, для розвитку штучного інтелекту в багатомовних контекстах, наприклад у машинному перекладі або чат-ботах. Додаток для перекладу тексту та аудіо, використовує натреновану модель, що демонструє практичне застосування, навіть для штучного інтелекту натренованого з обмеженими ресурсами.

Для подальших досліджень рекомендується вивчення інтеграції моделі mT5 з LoRa, що повинно покращити результати у сценаріях з обмеженими ресурсами та розширити функціонал.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Єременко А. І., Вакалюк Т. А. Обробка природної мови. Сучасні інформаційні системи та технології: матеріали VII Всеукр. наук.-практ. інтернет-конф. за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (29 листопада 2024 р., м. Херсон, м. Хмельницький) / за ред. А. А. Григорової. – Херсон: Книжкове видавництво ФОП Вишемирський В. С., 2024. С. 27-29, <http://www.tsatu.edu.ua/vmf/wp-content/uploads/sites/17/cict-2024-1.pdf>
2. Єременко А. І., Вакалюк Т. А. Огляд стратегій машинного навчання. Збірник матеріалів науково-практичної конференції «штучний інтелект і безпека» (21 листопада 2024 р., м. Київ) / за ред. ІПМЕ ім. Г.Є.Пухова НАН України, 2024. С. 55.
3. Онлайн перекладач DeepL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.deepl.com/en/translator>
4. Онлайн перекладач QuillBot [Електронний ресурс] – Режим доступу до ресурсу: <https://quillbot.com/translate>
5. Онлайн перекладач OpenL [Електронний ресурс] – Режим доступу до ресурсу: <https://openl.io>
6. Документація HTML [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
7. Документація CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
8. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
9. Документація Python [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3.11/>
10. Документація Flask [Електронний ресурс] – Режим доступу до ресурсу: <https://flask.palletsprojects.com/en/2.0.x/>
11. Документація Jinja2 [Електронний ресурс] – Режим доступу до ресурсу: <https://jinja.palletsprojects.com/en/3.0.x/>

- 12.Документація jQuery [Електронний ресурс] – Режим доступу до ресурсу:
<https://api.jquery.com>
- 13.Документація Flask-SocketIO [Електронний ресурс] – Режим доступу до ресурсу: <https://flask-socketio.readthedocs.io/en/latest>
- 14.Документація Werkzeug [Електронний ресурс] – Режим доступу до ресурсу: <https://werkzeug.palletsprojects.com/en/3.0.x/>
- 15.Документація Python OS [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/library/os.html>
- 16.Документація Transformers [Електронний ресурс] – Режим доступу до ресурсу: <https://huggingface.co/docs/transformers/index>
- 17.Документація SentencePiece [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/google/sentencepiece>
- 18.Документація Datasets [Електронний ресурс] – Режим доступу до ресурсу: <https://huggingface.co/docs/datasets/index>
- 19.Документація SpeechRecongition [Електронний ресурс] – Режим доступу до ресурсу: <https://pypi.org/project/SpeechRecognition>
- 20.Документація mT5 [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/transformers/v4.8.2/model_doc/mt5.html
- 21.Datasets: A Community Library for Natural Language Processing / Q. Lhoest et al. Association for Computational Linguistics. 2021. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 175–184, <https://doi.org/10.18653/v1/2021.emnlp-demo.21>.
- 22.mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer / L. X. Linting et al. Association for Computational Linguistics. 2021. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 483–498, <https://doi.org/10.18653/v1/2021.naacl-main.41>.
- 23.Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / C. Raffel et al. Exploring the Limits of Transfer Learning with a

- Unified Text-to-Text Transformer. 2020. Vol. 21 : Journal of Machine Learning Research. pp. 1–67, <https://doi.org/10.48550/arXiv.1910.10683>.
- 24.Онищенко К. Г., Данієль Я., Каменєв Р. Аналіз Методів Обробки Природної Мови. Друкарня Мадрид. Харків, 2020. Інформаційні системи та технології : матеріали 9-ї Міжнар. наук.-техн. конф.. С. 186–190, <https://openarchive.nure.ua/handle/document/16164>.
- 25.Yin X., Wan X. How Do Seq2Seq Models Perform on End-to-End Data-to-Text Generation?. Association for Computational Linguistics. 2022. Vol. 1 : Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. pp. 7701–7710, <https://doi.org/10.18653/v1/2022.acl-long.531>.
- 26.Context-aware seq2seq translation model for sequential recommendation / K. Sun et al. Information Sciences. 2021. Vol. 581. pp. 60–72, <https://doi.org/10.1016/j.ins.2021.09.001>.
- 27.Samreen Naeem, Aqib Ali, Sania Anam, Muhammad Munawar Ahmed. (2023). An Unsupervised Machine Learning Algorithms: Comprehensive Review. International Journal of Computing and Digital Systems. Vol. 1. pp. 911-921, <https://doi.org/10.12785/ijcds/130172>.
- 28.A Survey of Natural Language Generation / C. Dong et al. Association for Computing Machinery. 2022. Vol. 55 : ACM Computing Surveys. pp. 1–38, <https://doi.org/10.1145/3554727>.
- 29.Cripwell L., Legrand J., Gardent C. Document-Level Planning for Text Simplification. ACL. 2023. Vol. 1 : 17th Conference of the European Chapter of the Association for Computational Linguistics. pp. 993–1006, <https://doi.org/10.18653/v1/2023.eacl-main.70>.
- 30.Puduppully R. S. Data-to-text Generation with Neural Planning. The University of Edinburgh. 2022, <http://dx.doi.org/10.7488/era/2123>.
- 31.Patwardhan N., Marrone S., Sansone C. Transformers in the Real World: A Survey on NLP Applications. MDPI. 2024. Vol. 14 : Information. pp. 242, <https://doi.org/10.3390/info14040242>.

32. Goyal A., Bengio Y. Inductive biases for deep learning of higher-level cognition. The Royal Society Publishing. 2022. Vol. 478 : Proceedings of the Royal Society A. pp. 1–35, <https://doi.org/10.1098/rspa.2021.0068>.
33. Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation / B. Zhang et al. Association for Computational Linguistics. 2020. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1628–1639, <https://doi.org/10.18653/v1/2020.acl-main.148>.
34. Набір даних OPUS-100 [Електронний ресурс] – Режим доступу до ресурсу: <https://huggingface.co/datasets/Helsinki-NLP/opus-100>
35. Махаммедов Ж. Ж., Кирієнко О. В., Ткаченко В. О. Оцінка трансформерних моделей mT5 для українсько-англійського перекладу. Наукові записки НаУКМА. Комп'ютерні науки. 2025. Т. 8 : Нейронні мережі та машинне навчання. С. 97–101, <https://doi.org/10.18523/2617-3808.2025.8.97-101>.
36. Документація langdetect [Електронний ресурс] – Режим доступу до ресурсу: <https://pypi.org/project/langdetect>
37. Ma J., Yarats D. On the Adequacy of Untuned Warmup for Adaptive Optimization. AAAI Press. 2021. Vol. 35, iss. 10 : Technical Tracks. pp. 8828–8836, <https://doi.org/10.1609/aaai.v35i10.17069>.
38. Understanding the Adam Optimization Algorithm in Machine Learning / O. Hospodarsky et al. 2nd International Workshop on Computer Information Technologies in Industry 4.0. 2024. Vol. 3742. pp. 235-248, <https://ceur-ws.org/Vol-3742/paper17.pdf>.
39. Language models scale reliably with over-training and on downstream tasks / S. Y. Gadre et al. International Conference on Learning Representations. 2024. Vol. 2403.08540. pp. 1–22, <https://doi.org/10.48550/arXiv.2403.08540>.
40. Документація MediaRecorder [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder>

ДОДАТКИ

Код алгоритму очистки набору набору

```
from datasets import load_dataset
from langdetect import detect, lang_detect_exception

dataset = load_dataset('Helsinki-NLP/opus-100', 'en-uk')
train_dataset = dataset['train']

def lambdaHandler (i):
    try:
        if detect(i['translation']['uk']) == 'uk':
            return True
    except lang_detect_exception.LangDetectException:
        return False

new_dataset = train_dataset.filter(lambda i: lambdaHandler(i))
new_dataset.save_to_disk('dataset/mydata')

test_dataset = dataset['test']
new_test_dataset = test_dataset.filter(lambda i: lambdaHandler(i))
new_test_dataset.save_to_disk('dataset/mytestdata')
```

Повний код алгоритму тренування

```

from datasets import load_dataset
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
from transformers import get_linear_schedule_with_warmup
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import torch
from torch import optim
from torch.nn import functional as F
from tqdm import tqdm_notebook
import datasets

model_repo = 'google/mt5-small'
model_path = 'D:/curse/model/'
max_seq_len = 40

tokenizer = AutoTokenizer.from_pretrained(model_repo,
cache_dir='D:/curse/model/cache/')
model = AutoModelForSeq2SeqLM.from_pretrained(model_repo,
cache_dir='D:/curse/model/cache/')
model = model.cuda()
model.config.max_length = 40
dataset = load_dataset('Helsinki-NLP/opus-100', 'en-uk')
train_dataset = datasets.load_from_disk('dataset/mydata')
train_dataset = train_dataset.select(range(150000))
test_dataset = datasets.load_from_disk('dataset/mytestdata')

LANG_TOKEN_MAPPING = {
    'en': '<en>',
    'uk': '<uk>'
}

special_tokens_dict = {'additional_special_tokens':
list(LANG_TOKEN_MAPPING.values())}
tokenizer.add_special_tokens(special_tokens_dict)
model.resize_token_embeddings(len(tokenizer))

def encode_input_str(text, target_lang, tokenizer, seq_len,
                    lang_token_map=LANG_TOKEN_MAPPING):
    target_lang_token = lang_token_map[target_lang]

    input_ids = tokenizer.encode(
        text=target_lang_token + text,
        return_tensors='pt',
        padding='max_length',
        truncation=True,
        max_length=seq_len)

    return input_ids[0]

def encode_target_str(text, tokenizer, seq_len,
                    lang_token_map=LANG_TOKEN_MAPPING):
    token_ids = tokenizer.encode(
        text=text,
        return_tensors='pt',
        padding='max_length',
        truncation=True,
        max_length=seq_len)

```

```

    return token_ids[0]

def format_translation_data(translations, lang_token_map,
                           tokenizer, seq_len=128):
    langs = list(lang_token_map.keys())
    input_lang, target_lang = np.random.choice(langs, size=2, replace=False)
    input_text = translations[input_lang]
    target_text = translations[target_lang]

    if input_text is None or target_text is None:
        return None

    input_token_ids = encode_input_str(
        input_text, target_lang, tokenizer, seq_len, lang_token_map)

    target_token_ids = encode_target_str(
        target_text, tokenizer, seq_len, lang_token_map)

    return input_token_ids, target_token_ids

def transform_batch(batch, lang_token_map, tokenizer):
    inputs = []
    targets = []
    for translation_set in batch['translation']:
        formatted_data = format_translation_data(translation_set,
lang_token_map, tokenizer, max_seq_len)
        if formatted_data is None:
            continue
        input_ids, target_ids = formatted_data
        inputs.append(input_ids.unsqueeze(0))
        targets.append(target_ids.unsqueeze(0))
    batch_input_ids = torch.cat(inputs).cuda()
    batch_target_ids = torch.cat(targets).cuda()

    return batch_input_ids, batch_target_ids

def get_data_generator(dataset, lang_token_map, tokenizer, batch_size=32):
    dataset = dataset.shuffle()
    for i in range(0, len(dataset), batch_size):
        raw_batch = dataset[i:i + batch_size]
        yield transform_batch(raw_batch, lang_token_map, tokenizer)

n_epochs = 5
batch_size = 16
print_freq = 100
lr = 5e-4
n_batches = int(np.ceil(len(train_dataset) / batch_size))
total_steps = n_epochs * n_batches
checkpoint_freq = 1000
n_warmup_stes = int(total_steps * 0.01)
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
scheduler = get_linear_schedule_with_warmup(optimizer, n_warmup_stes,
total_steps)

def eval_model(model, gdataset, max_iters=8):
    test_generator = get_data_generator(gdataset, LANG_TOKEN_MAPPING,
tokenizer, batch_size)
    eval_losses = []

```

```

    for i, (input_batch, label_batch) in enumerate(test_generator):
        if i >= max_iters:
            break

        model_out = model.forward(
            input_ids=input_batch,
            labels=label_batch)
        eval_losses.append(model_out.loss.item())

    return np.mean(eval_losses)

losses = []
for epoch_idx in range(n_epochs):
    data_generator = get_data_generator(train_dataset, LANG_TOKEN_MAPPING,
                                        tokenizer, batch_size)

    for batch_idx, (input_batch, label_batch) in
        tqdm_notebook(enumerate(data_generator), total=n_batches):
        optimizer.zero_grad()
        model_out = model.forward(input_ids=input_batch, labels=label_batch)
        loss = model_out.loss
        losses.append(loss.item())
        loss.backward()
        optimizer.step()
        scheduler.step()

        if (batch_idx + 1) % print_freq == 0:
            avg_loss = np.mean(losses[-print_freq:])
            print('Epoch: {} | Step: {} | Avg. loss: {:.3f} | lr:
{}').format(epoch_idx + 1, batch_idx + 1, avg_loss,
scheduler.get_lr()[0]))

        if (batch_idx + 1) % checkpoint_freq == 0:
            test_loss = eval_model(model, test_dataset)
            print('Saving model with test loss of {:.3f}'.format(test_loss))
            model.save_pretrained(model_path)
            tokenizer.save_pretrained(model_path)

    test_loss = eval_model(model, test_dataset)
    print('Test loss of {:.3f}'.format(test_loss))

window_size = 50
smoothed_losses = []
for i in range(len(losses)-window_size):
    smoothed_losses.append(np.mean(losses[i:i+window_size]))

plt.plot(smoothed_losses[100:])
plt.show()

```

Код обробки сторінки перекладу тексту

```
@app.route('/', methods=['GET', 'POST'])
def translate_text_route():
    if request.method == 'GET':
        return render_template("index.html")
    else:
        texttotr = request.form['texttotr']
        lang = request.form['lang']
        return render_template("translated_text.html", text=texttotr, lang=lang,
                                sync_mode=socket_.async_mode)

@socket_.on('text_translate', namespace='/translate_text')
def translate_text(data):
    try:
        texttotr = data['texttotr']
        lang = data['lang']
        texttotr = re.split(r'([.!?])', texttotr)
        text = []
        for i in range(0, len(texttotr)-1, 2):
            text.append(texttotr[i] + texttotr[i+1])
        print(text)
        result = translate(text, lang)
        text = ' '.join(result)
        emit('task_done', {'data': text})
        disconnect()
    except Exception as ex:
        print(ex)
```