МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПЕРЕКЛАДУ ВІДЕО З АНГЛІЙСЬКОЇ НА УКРАЇНСЬКУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРМ – 601.21810313

Виконав студент 6-го курсу, групи 601
_____О. А. Костін
«___» _____ 2024 р.

Керівник: канд. фіз.-мат. наук, доцент
_____І. В. Кулаковська
«___» _____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___ р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Костіну Олександру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи магістра «Інтелектуальна система перекладу відео з англійської на українську».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз.-мат. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «01» лютого 2024 р. № 20

2. Строк подання студентом роботи __ _____ 202_ р.

3. Вхідні (початкові) дані до роботи: відео англійською мовою.

Очікуваний результат роботи: інтелектуальна система, яка здатна перекласти відео з англійської мови на українську; перекладене відео українською мовою.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

– створити систему перекладу відео з англійської мови на українську;

– в системі повинна бути можливість завантажити відео з різних джерел;

– система повинна мати якість звуку без явних дефектів, і можливість ясно чути синтезовану доріжку українською;

– система повинна мати переклад, який буде в змозі передати основний сенс.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Охорона праці та безпека у надзвичайних ситуаціях».

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	Григор'єва Л. І., д-р біол. наук, проф.	
Методична частина	Кулаковська І. В., канд. фіз.-мат. наук, доцент	

Керівник роботи канд. фіз.-мат. наук, доцент, Кулаковська І. В.
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Костін О. А._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__» _____ 202_ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи магістра

Тема: Інтелектуальна система перекладу відео з англійської на українську.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРМ. Подання заяви на затвердження теми КРМ	01.09.2023	25.09.2023	Виконано
2	Отримання завдання на виконання КРМ	26.09.2023	01.11.2023	Виконано
3	Складання календарного плану на період виконання КРМ	02.11.2023	10.11.2023	Виконано
4	Огляд літератури за темою дослідження	11.11.2023	26.11.2023	Виконано
5	Проходження передатестаційної практики, збір та аналіз матеріалів до КРМ	27.11.2023	24.12.2023	Виконано
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	25.12.2023	12.01.2024	Виконано
7	Опис фахової частини КРМ, зокрема дослідження публікацій щодо перекладу відео, огляд існуючих моделей для вирішення поставленої задачі, реалізація обраних технологій з аналізом отриманих результатів	13.01.2024	25.01.2024	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2024	02.02.2024	Виконано
9	Перший попередній захист КРМ на засіданні комісії кафедри	29.01.2024	29.01.2024	Виконано
10	Корегування роботи за результатами попереднього захисту	30.01.2024	05.02.2024	Виконано
11	Доробка та остаточне оформлення КРМ	06.02.2024	11.02.2024	Виконано
12	Другий попередній захист КРМ на засіданні комісії кафедри	12.02.2024	12.02.2024	Виконано
13	Подання КРМ, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.02.2024	20.02.2024	Виконано
11	Захист КРМ перед екзаменаційною комісією (ЕК)	27.02.2024	27.02.2024	Виконано

Розробив студент Костін О. А. _____
(прізвище та ініціали) (*підпис*)

Керівник роботи канд. фіз.-мат. наук, доцент, Кулаковська І. В. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202_ р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра
студента групи 601 ЧНУ ім. Петра Могили

Костіна Олександра Анатолійовича

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПЕРЕКЛАДУ ВІДЕО З АНГЛІЙСЬКОЇ НА УКРАЇНСЬКУ”**

Актуальність даного дослідження полягає у необхідності створення системи перекладу відео з англійської на українську і базується на зростаючій потребі української аудиторії в доступі до більшої кількості відео, зокрема англійською мовою.

Об’єктом дослідження є процес інтелектуалізації перекладу відео контенту.

Предметом дослідження є методи та алгоритми машинного перекладу, їх застосування до відео контенту, а також інтерфейс та взаємодія користувача з системою.

Метою дослідження є розробка та впровадження інтелектуальної системи перекладу відео з англійської на українську мову.

В результаті виконання роботи було реалізовано інтелектуальну систему перекладу відео з англійської на українську мову. Розроблена система здатна перекладати відео з англійської мови, завантажувати відео з різних джерел, забезпечувати якісний звук без очевидних дефектів та зрозумілу синтезовану аудіо доріжку українською мовою, передавати основне значення перекладу, а також має інтерфейс користувача для завантаження та отримання перекладеного відео.

Робота складається з фахової частини, спеціальної частини з охорони праці та методичної частини. Кожен розділ фахової частини відповідно присвячений: аналізу та огляду існуючих технологій автоматичного перекладу відео, методам та інформаційним технологіям автоматичного перекладу відео, розробці інтелектуальної системи перекладу та програмній реалізації та тестуванню створеної системи для перекладу відео. Загальний обсяг роботи – 130 сторінок. Кваліфікаційна робота магістра містить два додатки, 38 рисунків, 5 таблиць і посилання на 74 літературні джерела.

Ключові слова: переклад відео, автоматичний переклад, розпізнавання мовлення, текст у мовлення, розділення аудіо, мовлення у текст, синтез мовлення.

ABSTRACT

to the master's qualification work by the student of the group 601 of Petro Mohyla Black Sea National University

Kostin Oleksandr

“INTELLIGENT VIDEO TRANSLATION SYSTEM FROM ENGLISH TO UKRAINIAN”

The relevance of this study lies in the need to create a video translation system from English into Ukrainian and is based on the growing need of the Ukrainian audience to access more videos, in particular in English.

The object of the study is the process of intelligent translation of video content.

The subject of the study is machine translation methods and algorithms, their application to video content, as well as the user interface and interaction with the system.

The purpose of the study is to develop and implement an intelligent video translation system from English into Ukrainian.

As a result of the work, an intelligent video translation system from English to Ukrainian was realized. The developed system is able to translate videos from English, download videos from different sources, provide high-quality sound without obvious defects and a clear synthesized audio track in Ukrainian, convey the main meaning of the translation, and has a user interface for uploading and downloading translated videos.

The work consists of a thesis, a special part on labor protection, and a methodological part. Each section of the thesis is devoted to the analysis and review of existing technologies for automatic video translation, methods and information technologies for automatic video translation, development of an intelligent translation system, software implementation and testing of the created system for video translation.

The overall scope of the work is 130 pages. The master's thesis contains 2 applications, 38 figures, 5 tables and references to 74 literary sources.

Keywords: video translation, automatic translation, speech recognition, text-to-speech, audio separation, speech-to-text, speech synthesis.

ЗМІСТ

ВСТУП

Сучасний стан проблеми інтелектуального перекладу відео з англійської на українську має важливе значення в умовах збільшення можливостей інформаційних технологій, активного росту кількості контенту англійською та потребі в контенті українською. Тема цієї роботи стає дедалі актуальнішою в контексті зростаючої популярності відео контенту та необхідності розширення доступності інформації.

Однією з ключових проблем є недостатня кількість контенту українською мовою порівняно з англійською. Багато корисної та цікавої інформації доступно лише англійською, що ускладнює доступ до неї для людей, які не володіють цією мовою на високому рівні. Створення інтелектуальної системи перекладу відео дозволить українцям отримувати доступ до цієї інформації швидко та зручно, в перекладі на рідну мову.

Така система має потенціал сприяти збільшенню кількості доступного українського контенту, а також покращити освітні можливості та розвиток галузей як культури, науки та бізнесу в Україні. Існуючих рішень вкрай мало, і вони не можуть цілком покрити великі потреби людей в якісній інформації.

Дослідження було представлено на конференції «Могилянські читання 2023» 10 листопада 2023 року, з темою «Інтелектуальна система перекладу відео з англійської на українську».

Актуальність системи перекладу відео з англійської на українську базується на зростаючій потребі в доступі до більшої кількості відео, зокрема англійською мовою.

Метою даної роботи є розробка та впровадження інтелектуальної системи перекладу відео з англійської на українську мову.

Завдання, які мають бути вирішеними:

– створити систему, яка зможе перекласти відео з англійської мови на українську;

– в системі повинна бути можливість завантажити відео з різних джерел;

– система повинна мати якість звуку без явних дефектів, і можливість ясно чути синтезовану доріжку українською;

– система повинна мати переклад, який буде в змозі передати основний сенс;

– повинен бути реалізований інтерфейс для користувачів, що дозволяє завантажувати та отримувати перекладене відео.

Об'єктом дослідження є процес інтелектуалізації перекладу відео контенту.

Предметом дослідження є методи та алгоритми машинного перекладу, їх застосування до відео контенту, а також інтерфейс та взаємодія користувача з системою.

Методи що мають бути використані – мова програмування Python для розробки основного застосунку, моделі штучного інтелекту для вирішення задач переведення мовлення у текст, тексту в мовлення, а також перекладу тексту та розподілення звуків.

1 АНАЛІЗ ТА ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ АВТОМАТИЧНОГО ПЕРЕКЛАДУ ВІДЕО

1.1 Роль автоматичного перекладу в забезпеченні доступу до інформації

Згідно зі статистикою використання мов контенту для вебсайтів на 16 вересня 2023 року [1], кількість сайтів англійською складає 53.3% від усього інтернету, а українською – лише 0.6%. Щодо відео контенту, згідно зі статистки на основі даних відео хостингу YouTube на 19 вересня 2021 року [2], відео контенту англійською 66%, щодо української невідомо, проте не більше 2%. Автоматичний переклад і озвучення відео з англійської на українську допоможе українцям отримати доступ до майже в сто разів більшої кількості інформації.

У контексті війни, велика частина українського населення відмовляється від використання російського контенту. Згідно з аналітичним звітом у лютому 2023, проведеним Київським міжнародним інститутом соціології, 39% почали споживати більше українського контенту на YouTube, 31% відмовилися від російського, 6% припинили споживати, 27% змін не спостерігали [3]. Зважаючи на обмежену кількість доступного контенту українською мовою, існує практика переорієнтування на західні джерела, де обсяг інформації значно більший. Це не лише сприяє розширенню інформаційного спектру, але й сприяє глибшому розумінню культурних особливостей інших країн.

Однак існує проблема, пов'язана з неоднаковою мовною підготовкою українців, багато з яких не володіють англійською мовою на високому рівні. Згідно з дослідженням, яке було проведено Київським Міжнародним Інститутом Соціології [4], 49% людей не має жодних навичок з англійської, тобто не зможе зрозуміти навіть короткий текст. Це може створювати певні труднощі у здобутті необхідної інформації. В цьому контексті автоматичний переклад, заснований на штучному інтелекті, стає цінним інструментом, який забезпечує можливість доступу до інформації без значних зусиль з боку користувача. Він дозволяє не

тільки отримувати інформацію, але і глибше розуміти її, опираючись на автоматизований переклад.

Додатково, автоматичний переклад і озвучення відео з англійської мови на українську мову є лише початковим кроком. Ця технологія може бути розширена для перекладу контенту з інших мов, таких як французька, німецька і багато інших, що дозволить українській аудиторії максимально розширити свої можливості сприймати та розуміти світову інформацію та культуру. Наприклад, додавши іспанську та португальську, кількість доступного контенту на YouTube збільшиться на 15% та 7% відповідно [2].

Автоматичне озвучення відео виявляється суттєвою інноваційною технологією, що має потенціал вплинути на розвиток кіноіндустрії. Зокрема, у випадку успішної розробки автоматичного дубляжу, зі збереженням емоцій та індивідуальних голосів акторів, ця технологія спроможна забезпечити доступність контенту для аудиторії, що не володіє мовою оригіналу. Подібно до цього, дана інновація дає можливість трансформувати аудіовізуальні твори, створені для однієї мови, в інші, надаючи глобальній аудиторії можливість сприймати та розуміти ці твори, незалежно від їхньої мови.

Важливою перевагою автоматичного озвучення є розширення аудиторії для фільмів, серіалів та інших форм відеоконтенту. Ця технологія дозволяє особам різних культур та мовних груп насолоджуватися творами, походження яких різниться, і розуміти повідомлення, що були б недоступними для них без процесу автоматичного перекладу. Розширений доступ до культурного спадку сприяє культурному обміну та співробітництву між різними національними та мовними спільнотами [5].

Технологія автоматичного перекладу налічує широкий спектр можливих застосувань, відзначаючись своєю важливістю у різних галузях діяльності та сферах життя. Нижче наведено деякі з основних областей її застосування.

1. **Кіноіндустрія**. У сфері кіномистецтва автоматичне озвучення має потенціал забезпечити фільмам та серіалам можливість прозвучати українською

мовою, і це може бути досягнуто без значних витрат часу та ресурсів. Багато творів кінопромисловості не мають професійного дубляжу, і впровадження автоматичного перекладу дозволить українській аудиторії отримати доступ до цих творів.

2. **Освіта.** Технологія перекладу може бути використана для полегшення навчання шляхом надання студентам можливості доступу до відеоуроків та навчальних матеріалів на іноземних мовах. Це сприяє підвищенню рівня розуміння предмета та забезпечує учням більший доступ до глобальної освіти.

3. **Туризм**. Технологія автоматичного перекладу може бути надзвичайно корисною для туристів, які подорожують до іноземних країн. Вона надає змогу перекладати інструкції, описи місцевих пам'яток, ресторанів та музеїв, що сприяє зручності та комфорту під час подорожей.

4. **Міжнародні компанії**. Для міжнародних компаній, які співпрацюють з різними мовними спільнотами, автоматичний переклад може бути інструментом для надання навчальних та інформаційних матеріалів своїм співробітникам та клієнтам з різних країн. Це сприяє полегшенню комунікації та сприяє глобальній бізнес-взаємодії.

5. **Творці контенту**. Автори відеороликів та інших видів контенту можуть використовувати цю технологію для автоматичного перекладу та озвучення свого матеріалу, надаючи можливість аудиторії з різних країн отримувати доступ до їх творчості.

У цілому, автоматичне озвучення та переклад відео мають потенціал впливати на різні сфери життя та галузі індустрії, сприяючи збільшенню доступності і розширенню глобальної комунікації та розуміння між різними культурами та мовними спільнотами [5].

1.2 Існуючі технології автоматичного перекладу

Існує декілька технологій автоматичного перекладу відео. Розглянемо їх далі.

1. Elevenlabs.

Пропонується сервіс повного автоматичного дублювання, який можна легко створити за хвилини [6]. Цей сервіс здатний зберігати емоції та оригінальні голосові характеристики, автоматично перекладаючи весь контент [7]. Він підтримує роботу з багатьма мовами, включно з українською.

Для початку роботи потрібно перейти на їх сторінку «Dubbing» і натиснути кнопку « + Create new dub» (див. рис. 1.1).

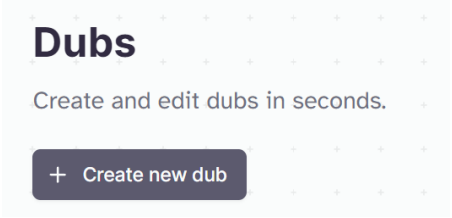

Рисунок 1.1 – Кнопка створення нового дубляжу

Далі треба заповнити поля з назвою, вказати потрібні мови та завантажити відео (див. рис. 1.2). Далі треба буде зачекати (див. рис. 1.3).

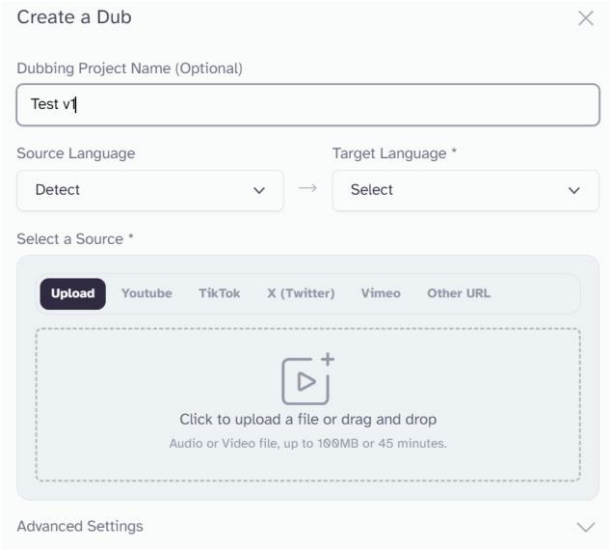

Рисунок 1.2 – Заповнення полів

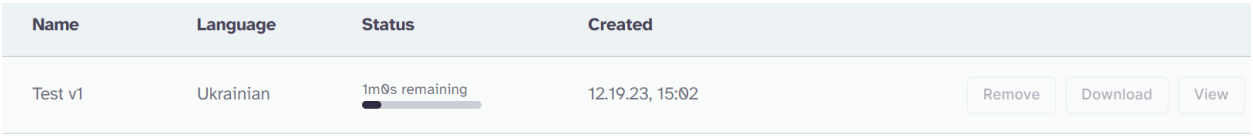

Рисунок 1.3 – Процес дубляжу

Після завершення процесу дубляжу можна переглянути та завантажити результат (див. рис. 1.4).

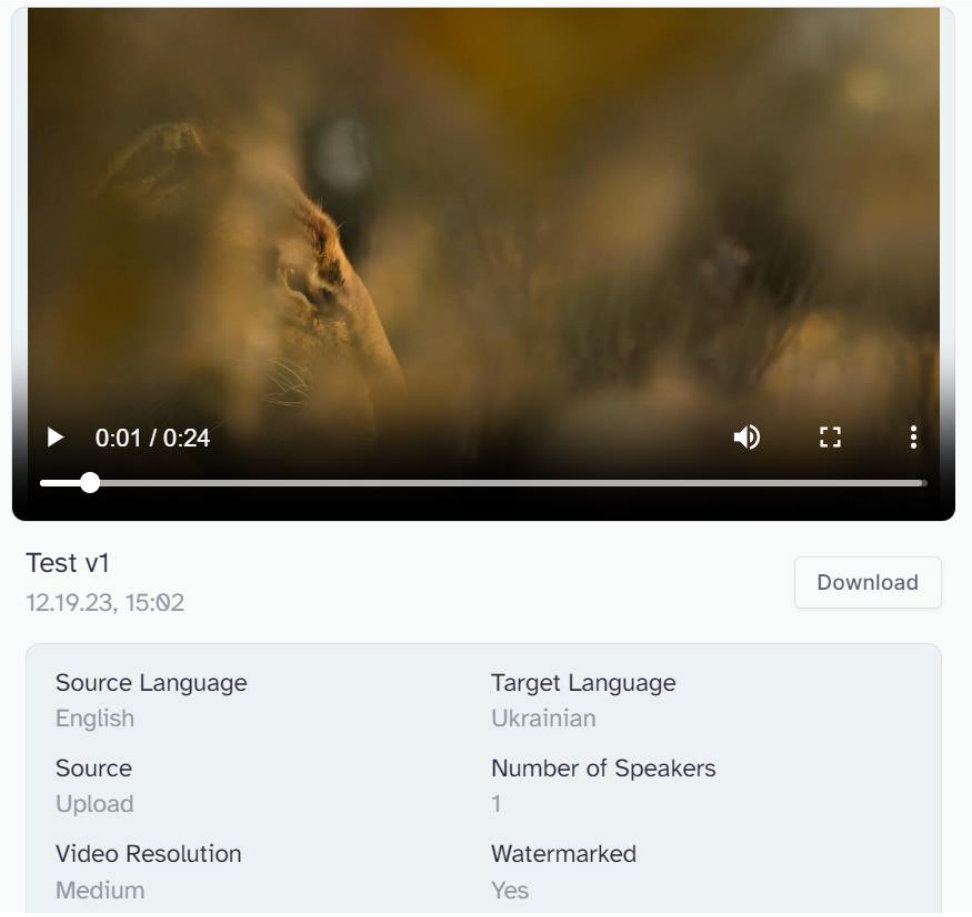

Рисунок 1.4 – Результат

Проте, станом на 18.12.2023, при спробі використання сервісу виявлено певні недоліки:

– синтезовані голоси мають дефекти, навіть у порівнянні із сервісом синтезу мови від тієї ж компанії, де такі дефекти не спостерігаються. Головною причиною цього є те, що сервіс прискорює або сповільнює голоси з метою

відповідності їхньому використанню у процесі дубляжу. Також якісь інші невідомі причини;

– неможливість корекції українського перекладу та виявлення неточностей в перекладі. Незважаючи на високу якість перекладу, присутність багатьох неперекладених фраз також зменшує якість сервісу;

– сервіс є платним, при цьому вартість перекладу відео тривалістю 30 хвилин може становити 20$.

Узагальнюючи, слід відзначити, що сервіс проявляє перспективи, однак йому необхідно надати додатковий час для вдосконалення та досягнення ідеальності в своїй роботі.

2. Вільний відеоперекладач.

Це бот у телеграм. В його описі написано: «Безкоштовний переклад відео YouTube з будь-якої мови на українську з якісною озвучкою» [8].

Для роботи, потрібно пройти перевірку на робота (див. рис. 1.5), і після успішного підтвердження надіслати посилання на потрібне відео.

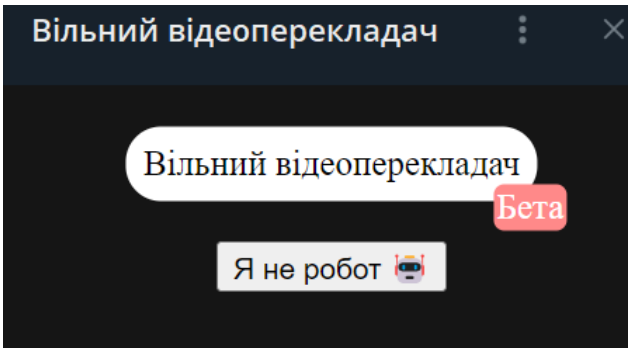

Рисунок 1.5 – Перевірка на робота

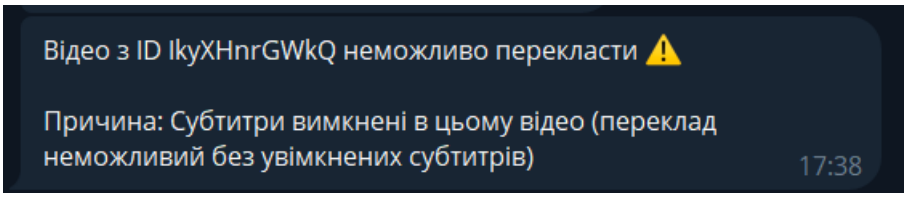

Рисунок 1.6 – Неможливість перекласти відео без субтитрів

Мінуси наступні:

– можна перекласти тільки відео з ютуб;

– перекласти можна лише відео, у яких є субтитри (див. рис. 1.6);

– за оригінальним звуком іноді не чутно українського.

3. Meta SeamlessM4T.

SeamlessM4T розроблено для забезпечення високоякісного перекладу, що дозволяє людям з різних мовних спільнот легко спілкуватися за допомогою мови та тексту [9]. Ця уніфікована модель дає змогу виконувати різні завдання, як-от переклад з мови на мову (S2ST), з мови на текст (S2TT), з тексту на мову (T2ST) тощо, не покладаючись на кілька окремих моделей [10].

Проте, під час проведення реальних тестів виявлено, що якість перекладу на українську мову є досить низькою, і часто це призводить до повного змінювання первинного змісту. Розробники визнають ці обмеження і вказують, що такі ситуації можливі. Деякі епізоди показують задовільні результати, проте загальна якість синтезу мовлення українською мовою також залишає бажати кращого. Спостерігається низька якість аудіо, а також пропуски у відтворенні деяких частин фраз.

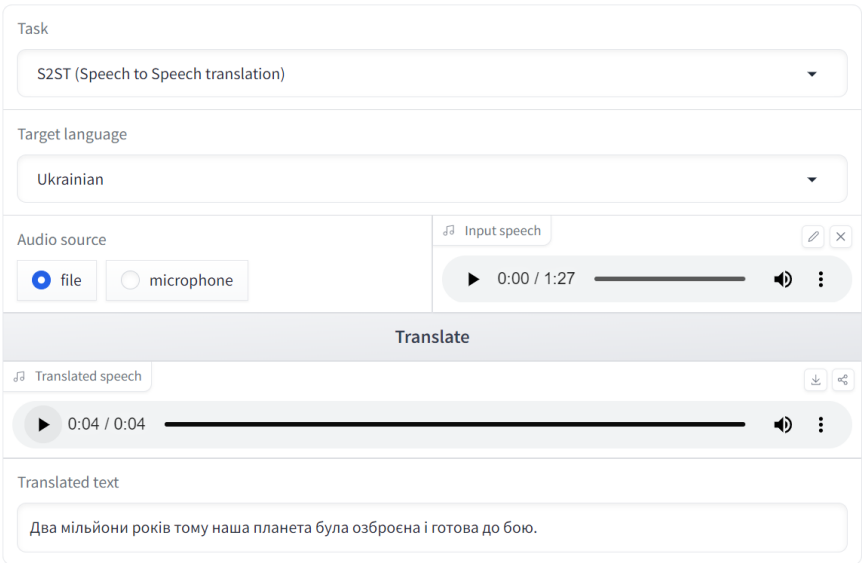

Рисунок 1.7 – Використання SeamlessM4T v1

У SeamlessM4T фразу «Two million years ago, and our **planet is a very different place**.», що означає «Два мільйони років тому наша **планета була зовсім інша**.», було перекладено як «Два мільйони років тому наша **планета була озброєна і готова до бою**». До того ж, у синтезованому аудіо промовляється «Два мільйони років тому наша **країна була озброєна і готова до бою**».

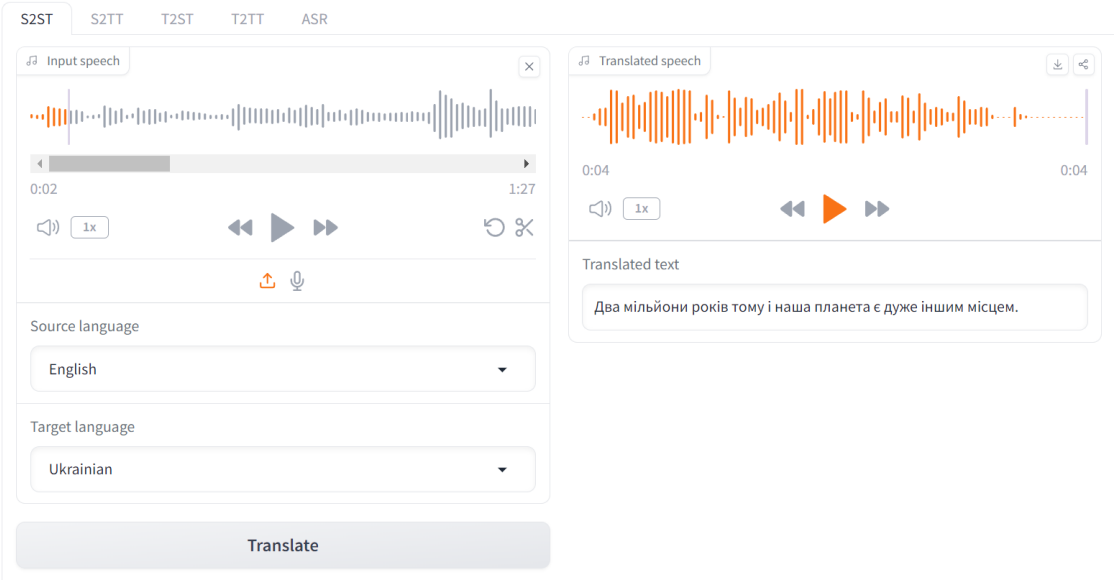

Рисунок 1.8 – Використання SeamlessM4T v2

У нової версії SeamlessM4T large v2 ситуація краще. Тут немає втрати сенсу, проте переклад не відчувається природним.

В цей раз фразу «Two million years ago, and our **planet is a very different place**.», що означає «Два мільйони років тому наша **планета була зовсім інша**.», було перекладено як «Два мільйони років і наша **планета є дуже іншим місцем**». Також синтезоване аудіо говорить цю ж фразу.

Загалом, всі ці сервіси пропонують можливості автоматичного перекладу та дублювання відео, але мають свої переваги та недоліки. Elevenlabs надає швидкий і простий спосіб створення дублювання, але має проблеми з якістю синтезованих голосів та перекладом українською мовою. Вільний відеоперекладач у телеграм спеціалізується на перекладі відео з YouTube на українську мову, але потребує

наявності субтитрів у вихідному відео. SeamlessM4T пропонує уніфіковану модель для різних завдань перекладу, але має проблеми з якістю перекладу на українську мову та синтезом мовлення. У новій версії є певні покращення, але проблеми залишаються. Узагальнюючи, всі ці сервіси мають потенціал, але потребують подальших покращень для досягнення ідеальності в роботі.

1.3 Вимоги до програмного забезпечення та постановка задачі

За результатом огляду сервісів, було створено таблицю 1.1.

Таблиця 1.1 – Порівняння існуючих сервісів перекладу відео

Особливість \ Сервіс	ElevenLabs	Вільний Відеоперекладач	Meta SeamlessM4T
Зручність використання	Простий у використанні	Простий у використанні	Потребує базових навичок програмування.
Початкова мова	Багато мов	Багато мов	Багато мов
Кінцева мова	Підтримує багато мов, включно з українською.	Тільки українська	Підтримує багато мов, включно з українською.
Завантаження відео	Можна завантажити будь-яке відео	Тільки відео з YouTube з субтитрами.	Тільки аудіо, але будь-яке
Якість перекладу	Висока	Висока, залежить від субтитрів у відео.	Середня
Недоліки перекладу	Неможливість виправити переклад.	Неможливість виправити переклад.	Неприродній переклад, але можна виправляти.
Якість синтезу мовлення	Висока	Середня	Низька
Дефекти аудіо	Іноді є деформація звуку	Іноді надто гучне оригінальне аудіо	Пропуск частин слів
Вартість	Близько $20 за 30-хв відео.	Безкоштовно	Безкоштовно

Основною метою є розробка та впровадження інтелектуальної системи озвучення відео, здатної перекласти англійське аудіо та озвучити його українською. Ця система покликана усунути обмеження існуючих технологій, забезпечуючи високоякісний контекстно-орієнтований переклад та природне звучання.

Сучасні технології автоматичного перекладу та дубляжу, такі як ElevenLabs, Вільний відеоперекладач та Meta SeamlessM4T, мають низку недоліків з точки зору точності перекладу, якості звуку та зручності для користувача, особливо в контексті української мови. Ці недоліки підкреслюють потребу у вдосконаленій системі, яка не лише точно перекладає, але й зберігає високу якість отриманого аудіо.

Система буде зосереджена на перекладі та дубляжі відеоконтенту з англійської на українську мову. Вона включатиме наступні вимоги до створюваної системи.

1. Можливість перекладати відео з різних джерел.

Наявність системи, що дозволяє системі ефективно опрацьовувати відеоматеріали різного формату та з різних джерел, забезпечуючи можливість перекласти будь-що.

2. Можливість перекладати відео як із субтитрами, так і без субтитрів.

Здатність перекладати відео не тільки із вбудованими субтитрами, а й мати можливість зробити субтитри автоматично, виходячи з того що є.

3. Можливість самостійно коригувати переклад відео.

Функціонал, що надає користувачеві можливість вручну виправляти або покращувати автоматичний переклад відео, забезпечуючи можливість створення максимально правильного перекладу.

4. Мати якість звуку без явних дефектів.

Забезпечення високоякісного синтезу аудіо для перекладу відео, виключаючи артефакти та дефекти, які можуть вплинути на якість звуку.

5. Мати можливість ясно чути синтезовану доріжку українською.

Здатність системи накладати синтезовану доріжку так, щоб її було добре чутно, і щоб оригінальна доріжка не заважала сприйняттю відео.

6. Мати переклад який буде в змозі передати основний сенс.

Перекладі відео повинен передавати принаймні основну суть і не спотворювати оригінальну думку. В ідеалі, щоб це звучало природньо і точно.

7. Мати можливість перекладу безкоштовно.

Забезпечення можливості користувачам здійснювати переклад відео без вартісних обмежень або обов'язкових платежів, що сприяє доступності та широкому розповсюдженню системи і доступного контенту.

Основними викликами є переведення аудіо в текст і подальший переклад. Також синтез повинен бути достатньо якісним, щоб у користувача не виникало проблем із прослуховуванням.

У проекті буде використано поєднання існуючих технологій на основі машинного навчання, для розпізнавання і синтезу мови і для перекладу.

Очікується, що успішна розробка цієї системи значно покращить досвід україномовної аудиторії у безперешкодному доступі до більшої кількості інформації і англомовного контенту. Потенційно систему можна застосовувати в освітньому контенті, розважальних медіа та послугах перекладу відеоматеріалів.

Висновки до розділу 1

У першому розділі було проведено аналіз існуючих технологій автоматичного перекладу відео та їхніх застосувань. Зазначено, що доступ до інформації для української аудиторії обмежений через нерівномірне розповсюдження контенту українською мовою порівняно з англійською. Автоматичний переклад та дублювання відео з англійської на українську може значно поширити доступ до інформації та сприяти глибшому розумінню культурних особливостей західних союзників України. Також відзначено проблему низького рівня англійської мови у багатьох українців та важкість отримання інформації через це. Автоматичний переклад на основі штучного інтелекту

визначено як цінний інструмент, який сприяє доступності інформації без значних зусиль з боку користувача.

У дослідженні також було вказано на можливість розширення технологій автоматичного перекладу на інші мови, що дозволить українській аудиторії максимально збільшити спроможність сприймати та розуміти світову інформацію та культуру. Відзначено, що ця технологія має потенціал впливати на розвиток кінематографу, особливо в контексті забезпечення доступу до контенту для аудиторії, яка не розмовляє у вихідній мові. Зазначено, що автоматичний переклад має широкий спектр можливих застосувань. Зроблено висновок, що ці технології сприяють збільшенню доступності та розширенню глобального спілкування та взаєморозуміння між різними культурними та мовними спільнотами.

2 МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ПЕРЕКЛАДУ ВІДЕО

2.1 Загальний алгоритм вирішення поставленої задачі

Відповідно до визначених вимог, був розроблений алгоритм програмного забезпечення. У реалізації можна виділити сім основних етапів обробки контенту:

1) відокремлення відео та аудіо. Для подальшої обробки відео та аудіо, вони відокремлюються. Ця операція виконується без перекодування, щоб ефективно маніпулювати аудіо та відео доріжками. Аналогічно в кінці все об’єднується;

2) розпізнавання мовлення. Цей етап передбачає розпізнавання мовлення в оригінальному відеоматеріалі. Цей процес передбачає аналіз аудіодоріжки з метою ідентифікації та транскрибування мовлення;

3) створення субтитрів та синхронізація. Субтитри створюються з розпізнаного в попередньому пункті мовлення. Для точної синхронізації субтитрів використовується спеціальний інструмент для корекції часових міток субтитрів, також слід провести нормалізацію отриманого результату;

4) переклад тексту. Наступним кроком є переклад отриманого тексту з оригінальної мови. Цей процес здійснюється за допомогою готових систем машинного перекладу, які використовують складні мовні моделі для забезпечення точності перекладу;

5) генерація мовлення. Останній етап включає в себе генерацію мовлення українською мовою, на основі перекладених субтитрів. Тут текстовий переклад перетворюється назад у мовленнєву форму, використовуючи технології синтезу мовлення;

6) приглушення оригінального звуку. Для підвищення якості фінального аудіо в контенті приглушується оригінальний звук за допомогою відповідної бібліотеки обробки аудіо, та накладається синтезований український;

7) об'єднання всього разом. Далі все об’єднується у фінальне відео.

Цей процес зображено на рис. 2.1 у більш детальній блок-схемі.

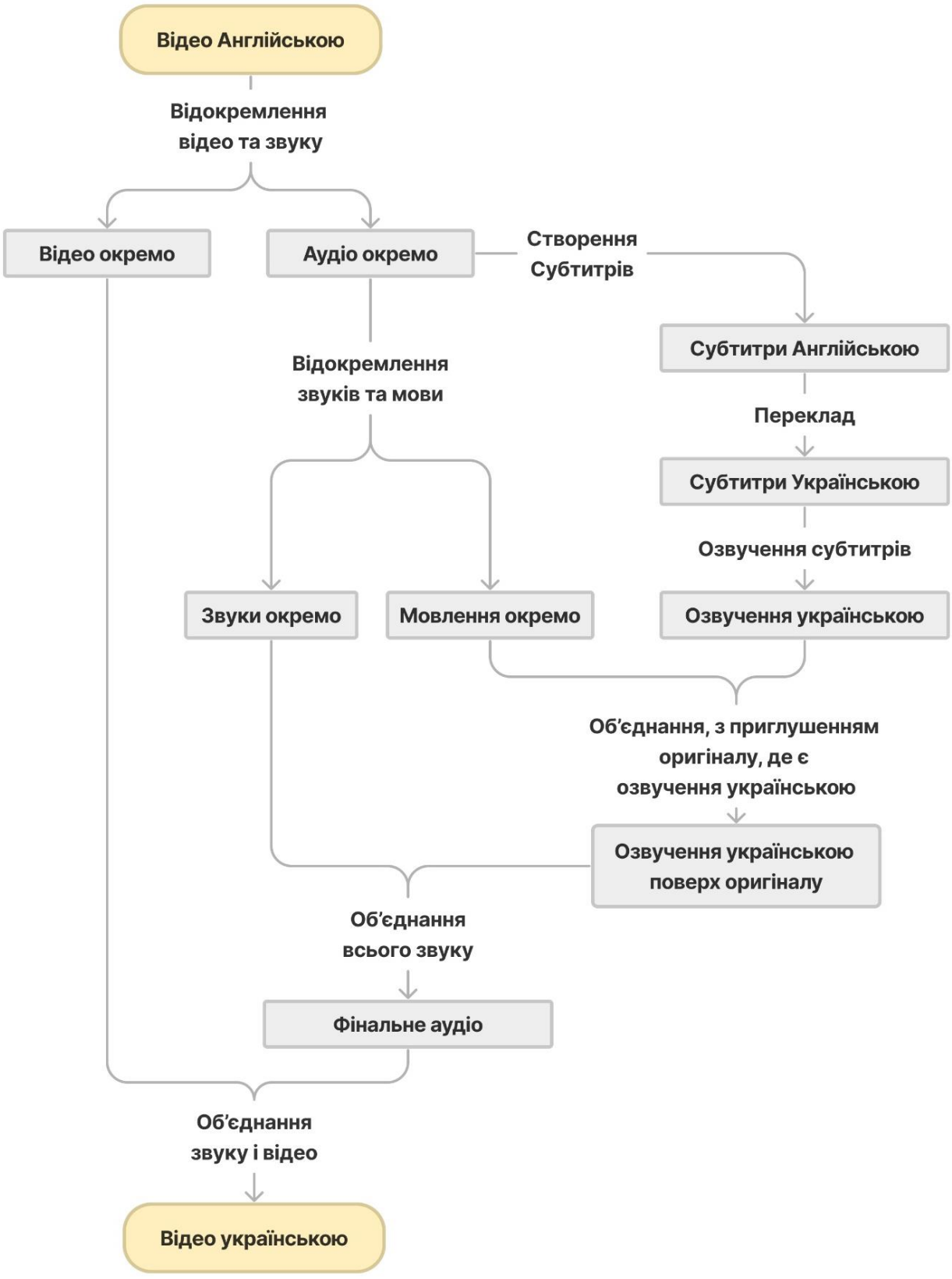

Рисунок 2.1 – Принцип роботи системи автоматичного перекладу відео

2.2 Автоматичне розпізнавання мовлення з аудіо в текст

Розпізнавання мовлення – це міждисциплінарна підгалузь інформатики та комп'ютерної лінгвістики, яка розробляє методології та технології, що дозволяють розпізнавати і перекладати розмовну мову в текст за допомогою комп'ютера. Вона також відома як автоматичне розпізнавання мови (ASR), комп'ютерне розпізнавання мови або перетворення мови в текст (STT).

Для оцінки ефективності засобу розпізнавання мовлення буде використаний показник частоти помилок у словах (WER), який є загальним показником ефективності системи автоматичного розпізнавання мови.

Загальна складність вимірювання ефективності полягає в тому, що розпізнана послідовність слів може мати іншу довжину, ніж еталонна послідовність слів (імовірно, правильна). WER походить від відстані Левенштейна, але працює на рівні слів, а не фонем. WER є цінним інструментом для порівняння різних систем, а також для оцінки покращень в межах однієї системи. Однак цей вид вимірювання не дає детальної інформації про природу помилок перекладу, тому необхідна подальша робота, щоб визначити основне джерело (джерела) помилок і зосередити будь-які дослідницькі зусилля.

Ця проблема вирішується шляхом попереднього вирівнювання розпізнаної послідовності слів з еталонною (вимовленою) послідовністю слів за допомогою динамічного вирівнювання рядків. Вивчення цього питання розглядається за допомогою теорії, яка називається степеневим законом, що встановлює кореляцію між розгубленістю та частотою помилок у словах [11].

Існує відкрита таблиця лідерів для моделей розпізнавання мовлення (Open ASR Leaderboard), яка оцінює моделі розпізнавання мови на Hugging Face Hub [12]. Моделі розташовані в рейтингу на основі їхнього середнього WER, від найнижчого до найвищого. Таблиця лідерів фокусується на розпізнаванні англійської мови, що і потрібно для вирішення поставленої задачі. У додатку А наведена ця порівняльна таблиця станом на 27.01.2024.

Лідером цієї таблиці є модель від Nvidia під назвою Parakeet RNNT 1.1B. Проте вона транскрибує мовлення малими літерами англійського алфавіту, що не підходить для поставленої задачі, бо це зменшить подальшу якість перекладу. Слід зазначити, що кращий показник WER не завжди показує краще розуміння результату людьми [13].

Наступною якісною моделлю, яка також вміє розпізнавати пунктуацію є модель Whisper large v3 від OpenAI. Це дуже корисно для створення автоматичного озвучення, оскільки чим точніше буде розпізнаний текст, тим краще буде переклад і фінальний результат.

Whisper – це відкрита, попередньо навчена модель для автоматичного розпізнавання мови та перекладу. Модель large v3 навчена на 1 млн. годин розмічених даних. Загалом, моделі Whisper демонструють високу здатність до узагальнення для багатьох наборів даних і галузей без необхідності тонкого налаштування [14]. Моделі навчалися як на англомовних, так і на багатомовних даних. Whisper – це ncoder-decoder модель на основі трансформатора, яку також називають sequence-to-sequence моделлю.

Моделі Whisper поставляються в п'яти конфігураціях з різними розмірами моделей. Найменші чотири навчаються або на англомовних, або на багатомовних даних. Найбільші є лише багатомовними. Для виконання завдання буде використана модель large-v3, яка найкраще справляється з розпізнаванням англійської та інших мов.

2.3 Методи розділення мовлення та інших звуків в аудіо

Розділення звуків – це завдання розділення аудіосигналів на окремі компоненти. Традиційно, існує два варіанти розподілення звуку:

1) Cinematic Sound Demixing Track (CDX) – поділ кінематографічної звукової доріжки на діалоги, звукові ефекти та решту [15];

2) Music Demixing Track (MDX) – розділення музичного аудіосигналу на окремі доріжки для вокалу, бас-гітари, барабанів та іншого [16].

У випадку вирішення нашої задачі, необхідно розділити звук на діалоги та решту. Існує широкий спектр архітектур з глибоким навчанням для розділення джерел звуку. Більшість з них використовують архітектуру кодер-декодер, натхненну архітектурою U-net [17]. Кодер витягує репрезентативні карти ознак різних просторових вимірів, а декодер реконструює семантичну маску з повною роздільною здатністю. Для деміксингу музики використовуються різні підходи, що базуються як на формах хвиль [18, 19], так і на спектрограмних доменах [20]. Найпопулярнішими моделями на сьогоднішній день є Spleeter [21], Unmix, Demucs (версії 2, 3 і 4) [22, 20], MDX-Net [23] і Ultimate Vocal Remover (UVR) [24].

UVR – це бібліотека з набором попередньо навчених моделей для виділення вокалу, яка включає попередньо навчені моделі MDX-Net і перенавчені моделі Demucs, а також власні моделі архітектури для виділення вокалу. Під час роботи буде використана ця бібліотека на Python. Всі моделі там мають гарну якість і можуть працювати з довгими аудіо. Був проведений тест основних моделей на однаковому аудіо довжиною 30с. Для роботи обрана модель UVR_MDXNET_9482, так як вона виявилась найшвидшою (див. табл. 2.1). В даній роботі якість не грає визначну роль, адже отримана голосова доріжка не буде видалятись повністю, а лише ставати тихіше в певних моментах.

Таблиця 2.1 – Порівняння швидкості моделей, які використовують в UVR

Модель	**Час**
UVR_MDXNET_9482	13,13 sec
UVR_MDXNET_1_9703	14,01 sec
UVR_MDXNET_2_9682	14,41 sec
UVR_MDXNET_KARA	14,72 sec
UVR_MDXNET_3_9662	14,86 sec
UVR_MDXNET_KARA_2	17,5 sec
UVR-MDX-NET-Inst_Main	18,03 sec
Kim_Vocal_2	21,67 sec
UVR-MDX-NET-Inst_3	21,81 sec

Закінчення таблиці 2.1

UVR_MDXNET_Main	21,9 sec
UVR-MDX-NET-Inst_1	22,31 sec
Kim_Vocal_1	22,57 sec
UVR-MDX-NET-Inst_HQ_3	22,84 sec
UVR-MDX-NET-Voc_FT	22,84 sec
Kim_Inst	22,86 sec
UVR-MDX-NET-Inst_2	22,92 sec
UVR-MDX-NET-Inst_HQ_1	23,36 sec
UVR-MDX-NET-Inst_HQ_2	23,52 sec

2.4 Переклад субтитрів

Під час роботи будуть перекладатись саме субтитри, бо в них, окрім тексту, наявні часові мітки. Їх можна використати для подальшої розстановки згенерованого аудіо. Розглянемо декілька високоякісних перекладачів.

Google Translate - багатомовний сервіс нейронного машинного перекладу, розроблений компанією Google для перекладу тексту, документів і вебсайтів з однієї мови на іншу. Він пропонує інтерфейс вебсайту, мобільний додаток для Android та iOS, а також API, який допомагає розробникам створювати розширення для браузерів і програмні додатки [25]. Станом на 2022 рік Google Translate підтримує 133 мови на різних рівнях [26]. Станом на квітень 2016 року він налічував понад 500 мільйонів користувачів [27], щодня перекладаючи понад 100 мільярдів слів.

ChatGPT (Chat Generative Pre-trained Transformer) – це чат-бот, розроблений OpenAI і запущений 30 листопада 2022 року. Заснований на великій мовній моделі, він дозволяє користувачам вдосконалювати та спрямовувати розмову до бажаної довжини, формату, стилю, рівня деталізації та мови. Послідовні підказки та відповіді, відомі як інженерія підказок, розглядаються на кожному етапі розмови як контекст [28]. Хоча основною функцією чат-бота є імітація людського

співрозмовника, ChatGPT є універсальним. Серед незліченних прикладів, він може створювати код, узагальнювати тексти, писати вірші та інше. І головне, він може перекладати тексти [29].

DeepL Translate – це онлайн-сервіс для машинного перекладу, розроблений німецькою компанією DeepL [30]. Він використовує метод машинного перекладу на основі нейронних мереж, який дозволяє генерувати більш природний і точний переклад, ніж традиційні методи. DeepL Translate підтримує 29 мов, включно з англійською та українською. Він також підтримує переклад між мовами, які важко перекладати, такими як японська та китайська. DeepL Translate був запущений у 2017 році і швидко став одним із найпопулярніших сервісів машинного перекладу. Він отримав високі оцінки від критиків і користувачів, які відзначають його точність, природність і легкість у використанні.

Давайте спробуємо перекласти певний фрагмент тексту за допомогою цих перекладачів. За основу взятий текст з трейлеру «Світ хижаків» від Netflix.

Оригінальний текст: «Predators across the planet battle hunger in the most hostile environment. All striving to find a way in their new world. In Serengeti, a brotherhood of cheetahs try to cling on to their mighty alliance. In South America, puma numbers are booming. But more cats means more competition. In subarctic Canada, the largest bears on earth dominate».

ChatGPT 4: «Хижаки по всій планеті борються з голодом у найворожіших умовах. Всі намагаються знайти свій шлях у світі, який змінюється. У Серенгеті братство гепардів намагається зберегти свій могутній союз. У Південній Америці число пум зростає. Але більше котів означає більше конкуренції. У субарктичній Канаді домінують найбільші ведмеді на землі».

Deepl Translate: «Хижаки по всій планеті борються з голодом у найбільш ворожому середовищі. Усі вони намагаються знайти свій шлях у новому світі. У Серенгеті братство гепардів намагається зберегти свій могутній альянс. У Південній Америці чисельність пум стрімко зростає. Але більше котів означає

більшу конкуренцію. У субарктичній Канаді домінують найбільші ведмеді на землі».

Google Translate: «Хижаки по всій планеті борються з голодом у найворожішому середовищі. Усі прагнуть знайти шлях у свій новий світ. У Серенгеті братство гепардів намагається вчепитися за свій могутній союз. У Південній Америці чисельність пум процвітає. Але більше котів означає більше конкуренції. У субарктичній Канаді домінують найбільші ведмеді на землі».

Оригінальний переклад: «Хижаки по всій планеті борються з голодом у найбільш ворожому середовищі. Усі прагнуть вижити у своєму новому світі. У Серенгеті братство гепардів намагається триматися за свій могутній союз. У Південній Америці кількість пум різко зростає. Але більше котячих — більше конкуренції. У субарктичній Канаді домінують найбільші ведмеді на землі».

Всі перекладачі в середньому справились добре. Виникає необхідність у порівнянні їхніх тарифів та можливостей безкоштовного користування.

Сервіс Deepl Translate пропонує підписку 5 євро та оплату в розмірі 20 євро за кожний мільйон символів [31]. Крім того, вони надають можливість безкоштовного користування в режимі "free", обмежену обсягом 500 тисяч символів щомісячно. Проте варто відзначити, що оформлення підписки на їхній сервіс в Україні неможливе.

У свою чергу, Google Translate API пропонує схожі цінові умови. Вони встановлюють ціну у 20 доларів США за кожний мільйон символів та надають можливість безкоштовного користування обсягом 500 тисяч символів [32]. Головною перевагою є те, що їхні послуги доступні для користувачів в Україні.

Ціноутворення GPT-4 відрізняється від звичайного підходу. Ціни встановлюються в ньому в токенах, де 1000 токенів відповідають приблизно 5500 символам англійською [33] та 1600 символам українською мовою. Додаткову інформацію можна знайти на їхньому вебсайті [34]. У цій системі вартість введення складає $0.01 за 1K токенів, а виведення – $0.03 за 1K токенів. Підраховуючи, отримаємо: 1000000/55000.01 = 1.81 та 1000000/16000.03 = 18.75.

Загальна сума складає приблизно 20.56 доларів. Таким чином, переклад тексту з англійської на українську мову обійдеться приблизно в 20.56 доларів. Також вони надають 5$ на перші три місяці користування. Сервіс доступний в Україні.

Таблиця 2.2 – Порівняльна таблиця якісних перекладачів

Перекладач	Ціна за 1М символів	Безкоштовні опції	Доступність в Україні
ChatGPT 4	20,56$ за 1 млн. символів	5$ на перші 3 місяці	Так
Deepl Translate	5€ +20€ за 1 млн. символів	500 тис. символів у місяць	API недоступне
Google Translate	20$ за 1 млн. символів	500 тис. символів у місяць	Так

Таким чином, щоб забезпечити українців перекладом з англійської на українську, кращім рішенням буде використовувати Google Translate, так як вони доступні в Україні і кожного місяця надають велику кількість символів для перекладу.

2.5 Методи синтезу мовлення українською

Синтез мовлення – це штучне відтворення людської мови. Комп'ютерна система, що використовується для цієї мети, називається синтезатором мовлення і може бути реалізована у вигляді програмного або апаратного забезпечення. Система перетворення тексту в мовлення (Text-to-speech, TTS) перетворює звичайний мовний текст у мовлення; інші системи перетворюють символічні лінгвістичні представлення, такі як фонетичні транскрипції, у мовлення[35]. Зворотним процесом є розпізнавання мовлення.

На сьогоднішній день існує достатньо синтезаторів мовлення українською мовою, серед яких можна виділити кілька найякісніших рішень. Серед них Elevenlabs [6], TTS від OpenAI [33], Microsoft Azure [36], Google WaveNet [37] і Ukrainian TTS [38], побудований на базі ESPnet. Варто відзначити, що Ukrainian

TTS є єдиним відкритим та безкоштовним синтезатором мовлення серед перелічених. Більше того, він демонструє досить стабільну та високоякісну роботу, що робить його гарним вибором для вирішення поставленої задачі.

ESPnet (End-to-End Speech Processing Toolkit) представляє собою інструментарій для комплексної обробки мовлення, призначений для вирішення завдань розпізнавання та синтезу мовлення. Цей інструмент включає в себе ряд ключових компонентів, таких як обробка акустичних сигналів, фонемні та мовні моделі, а також інтеграцію з сучасними методами машинного навчання. Його основною перевагою є можливість розв'язувати складні завдання обробки мовлення від початку до кінця, що дозволяє дослідникам і інженерам ефективно працювати над різноманітними аспектами цієї галузі [35].

Однією з важливих характеристик ESPnet є його відкритий код [39], що сприяє активному розвитку та спільнотовій співпраці. Це дозволяє користувачам розширювати функціональність інструменту, а також впроваджувати власні інновації у сфері обробки мовлення.

Ukrainian-TTS – це потужний інструмент для озвучування тексту українською мовою, який використовує передові технології та алгоритми для створення натурального та виразного мовлення. Основою Ukrainian-TTS є описана вище платформа з відкритим вихідним кодом для наскрізної обробки мовлення під назвою ESPNET [35], що дозволяє ефективно перетворювати текст на мовлення з високою точністю та природністю. Система підтримує кілька голосів, що забезпечує різноманітність тембрів та стилів мовлення, а також дозволяє користувачам вибирати найбільш підходящий голос для конкретних завдань.

Однією з ключових особливостей Ukrainian-TTS є її здатність автоматично встановлювати наголоси в тексті, використовуючи комбінацію словникових даних та моделей машинного навчання [38]. Це значно покращує якість та природність мовлення, роблячи його більш зрозумілим та приємним для слуху.

2.6 Обґрунтування та вибір технологій розробки ПЗ

Мова програмування.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з суворою динамічною типізацією [40], розроблена Гвідо ван Россумом у 1990 році. Її властивості, такі як високорівневі структури даних, динамічна семантика та динамічне зв'язування, роблять її привабливою для швидкого розроблення програм і поєднання наявних компонентів. Його філософія дизайну наголошує на читабельності коду з використанням значних відступів [41].

Однією з головних переваг Python є його модульність та повторне використання коду завдяки підтримці модулів та пакетів модулів. Це дозволяє розробникам легко організувати свій код у логічні модулі та забезпечує гнучкість у використанні.

Зазвичай Python використовується для широкого спектру завдань, від розробки вебзастосунків та аналізу даних до наукових досліджень та розв'язання складних математичних задач. Він підтримує кілька парадигм програмування, таких як об'єктно-орієнтована, процедурна, аспектно-орієнтована та функціональна, що дозволяє розробникам вибирати підхід, який найкраще відповідає їхнім потребам та задачам.

За останні роки Python став дуже популярним в області штучного інтелекту (ШІ), завдяки великій кількості бібліотек та інструментів, які сприяють розробці і реалізації моделей ШІ. Його зручність у використанні та розширення робить його вибором номер один для багатьох дослідників та розробників у цій сфері.

Робота з моделями штучного інтелекту стає набагато простішою та швидшою завдяки використанню Python. Бібліотеки, такі як TensorFlow, PyTorch, та Scikit-learn, дозволяють розробникам швидко створювати, навчати та реалізовувати різноманітні моделі і нейронні мережі для вирішення різноманітних завдань у сфері штучного інтелекту. Такий підхід сприяє прискоренню розробки та впровадженню інтелектуальних систем у практичні додатки, що в свою чергу сприяє розвитку цієї

галузі та її застосуванню в різних сферах життя. І оскільки в проєкті буде робота з великою кількістю різноманітних готових моделей, використання Python зробить цей процес зручним і легким.

Операційні системи.

При розробці системи важливо враховувати факт, що користувачі можуть мати різні пристрої з різними ОС. Тому використання вебзастосунків є найбільш оптимальним рішенням. Вебзастосунки дозволяють запускати програми через веббраузер на будь-якій операційній системі, що має підтримку браузера. Вебпрограми доставляються у всесвітній павутині користувачам, які мають активне підключення до мережі [42].

Python, як мова програмування, забезпечує зручний спосіб розробки вебзастосунків. Одним з інструментів, які допомагають створювати вебзастосунки з використанням Python, є Gradio.

Gradio – це бібліотека для швидкої розробки інтерактивних інтерфейсів для моделей машинного навчання та обробки даних з допомогою Python [43]. Вона дозволяє розробникам створювати вебінтерфейси для моделей машинного навчання безпосередньо з коду Python за допомогою кількох рядків коду. Gradio надає можливість відтворювати, тестувати та налаштовувати моделі машинного навчання в реальному часі простим інтерактивним способом через вебінтерфейс.

Використання Gradio дозволяє створювати вебзастосунки, які можуть бути запущені на будь-якій операційній системі з веббраузером, що робить їх доступними для широкого кола користувачів. Такий підхід до розробки системи дозволяє забезпечити її доступність та зручність використання для користувачів незалежно від операційної системи, що вони використовують.

Середовище розробки.

При розробці програмного забезпечення одним з важливих кроків є вибір середовища розробки, що забезпечить зручність, продуктивність та ефективність у процесі написання коду. Одним із найпопулярніших та універсальних середовищ є Visual Studio Code.

Visual Studio Code – це безкоштовний, легкий та потужний текстовий редактор, розроблений компанією Microsoft [44]. Він підтримує багато мов програмування та різноманітні технології, що робить його популярним серед розробників різних спеціалізацій. Переваги VS Code наступні:

– *безкоштовність та відкритість*. VS Code доступний безкоштовно для всіх користувачів, що робить його доступним для широкого кола розробників. Крім того, він має відкрите джерело, що дозволяє спільноті вносити власні покращення та розширення;

– *розширюваність*. VS Code має широкий вибір розширень, що дозволяє кожному користувачеві налаштувати редактор під свої потреби. Ці розширення включають підтримку різних мов програмування, інструменти для роботи з версійним контролем, вбудовані відладчики та багато іншого;

– *крос-платформенність*. VS Code підтримується на всіх основних операційних системах: Windows, macOS та Linux, що дозволяє розробникам працювати на своєму улюбленому середовищі;

– *швидкість та продуктивність*. Редактор працює швидко та ефективно, навіть з великими проектами, завдяки своїй легкості та оптимізаціям.

Використання Visual Studio Code для розробки на мові програмування Python також має свої переваги. VS Code надає вбудовану підтримку для Python, що включає автодоповнення, перевірку синтаксису, інтегровані інструменти для відладки та багато іншого. Також, VS Code має велику та активну спільноту користувачів Python, яка постійно розробляє нові розширення та доповнення для полегшення роботи з мовою.

Ще редактор легко інтегрується з іншими інструментами для розробки на Python, такими як інтерпретатори, середовища віртуальних середовищ та бібліотеки для аналізу даних. Також, VS Code має розширення для популярних фреймворків Python, таких як Django, Flask та інші, що спрощує розробку вебдодатків та інших проєктів.

Загалом, Visual Studio Code є потужним та зручним інструментом для розробки на Python, який надає розробникам засоби для продуктивної та ефективної роботи над проектами будь-якої складності.

Система контролю версій.

GitHub є однією з найпопулярніших платформ для спільної роботи над програмними проектами та зберігання їх версій у вебсервісі. Вона базується на системі контролю версій Git, яка є розподіленою і дозволяє розробникам відслідковувати зміни у файлах та працювати над проектами разом з колегами, незалежно від їх географічного розташування [45]. Git став популярним інструментом в розробці програмного забезпечення завдяки своїй швидкості, простоті та надійності.

GitHub, як сервіс онлайн-хостингу репозиторіїв, не лише надає функціональність Git, але й доповнює її різноманітними інструментами та сервісами для зручного управління проектами. Окрім базової можливості зберігання та відстеження версій коду, вона надає контроль доступу, систему багтрекінгу, функціонал керування завданнями та можливість створення вікі для кожного проекту.

Багтрекінг на GitHub дозволяє розробникам ефективно відслідковувати помилки, пропозиції та інші аспекти розвитку проекту, що робить процес розробки більш організованим та структурованим. Керування завданнями спрощує процес розподілу роботи між учасниками проекту, дозволяючи призначати, відстежувати та вирішувати завдання.

Таким чином, GitHub не лише надає інструменти для зберігання та відстеження версій коду, але і створює сприятливі умови для виправлення помилок.

Розробка дизайну.

Figma представляє собою інноваційну програму, яка займає універсальну позицію у світі дизайну та розробки інтерфейсів. У порівнянні з іншими спеціалізованими програмними рішеннями, які зазвичай зорієнтовані на конкретні завдання, Figma пропонує більш широкий функціонал, який може задовольнити

різноманітні потреби користувачів. Це забезпечує використання програми не лише для створення інтерфейсів користувача, але і для розробки прототипів, редагування векторних графіків, спільної роботи над проектами та інших завдань у сфері дизайну та розробки.

Однією з основних переваг Figma є його доступність та легкість в освоєнні. Доступ до програми можна отримати через веббраузер без необхідності встановлення спеціального програмного забезпечення [46], що робить її зручним інструментом для роботи навіть для тих користувачів, які не мають досвіду у програмуванні чи дизайні. Багатофункціональність та можливість спільної роботи над проектами у реальному часі роблять Figma привабливим варіантом для командних проектів, де залучені різні спеціалісти з різних областей.

Загалом, Figma відзначається своєю універсальністю, доступністю та легкістю в освоєнні, що робить його потужним інструментом для дизайну та розробки інтерфейсів користувача для різних типів проектів. Його можливості дозволяють не лише створювати естетичні та функціональні інтерфейси, але й ефективно співпрацювати над проектами та досягати високих результатів у короткі терміни.

Висновки до розділу 2

У другому розділі спочатку було створено загальну концепцію майбутнього застосунку і вирішено які задачі вирішувати.

Далі було розглянуто завдання автоматичного розпізнавання мовлення (ASR), яке використовується для перетворення мовлення на текст. В даному контексті було важливо порівняння значень показників помилок слів (WER) для оцінки продуктивності систем ASR. Кращим рішенням виявилось використання моделі Whisper large v3, розробленої OpenAI, яка володіє високою якістю розпізнавання та вміє розпізнавати пунктуацію.

Далі, у розділі розглянуто завдання розділення звуків у аудіо на окремі компоненти, зокрема на діалоги та інші звуки. Описано різні архітектури глибокого

навчання для вирішення цієї задачі та вказано на існування бібліотеки UVR для вокального виділення. Для розподілення мовлення та інших звуків було обрано модель UVR_MDXNET_9482, так як провівши порівняння з іншими моделями було виявлено, що ця найшвидша.

Далі, у розділі розглянуто завдання перекладу субтитрів та представлено порівняльний аналіз трьох перекладачів: Google Translate, ChatGPT 4 та DeepL Translate. Порівняно оригінальний текст та результати перекладу, а також вказано на особливості кожного з перекладачів, їх ціни та доступність в Україні. Порівнюючи ці переклади, було відзначено, що Google Translate є найкращим варіантом для перекладу з англійської на українську, оскільки він доступний в Україні та надає достатньо символів для перекладу щомісяця.

Потім розглянуто синтез мовлення українською мовою та вказано на існування декількох якісних рішень, зокрема Elevenlabs, TTS від OpenAI, Microsoft Azure, Google WaveNet та українського TTS на основі ESPnet. Надано інформацію про особливості ESPnet та українського TTS. Зроблено акцент на тому, що український TTS є відкритим та безкоштовним рішенням, і також має стабільну та високоякісну роботу, що і було вирішальним фактор у виборі цього рішення.

У заключному розділі розглянуто вибір технологій для розробки програмного забезпечення. Зокрема, розглянуті такі аспекти як вибір мови програмування, операційної системи, середовища розробки, системи контролю версій та інструменту для розробки дизайну.

Python був вибраний як основна мова програмування через свою простоту використання, багатофункціональність та підтримку багатьох парадигм програмування. Особливу увагу було приділено роботі зі штучним інтелектом, де Python виявив себе найкращим вибором завдяки великій кількості бібліотек та інструментів для розробки та реалізації моделей.

У розробці системи було обрано використання вебзастосунків через їхню універсальність та можливість запуску на будь-якій операційній системі, що підтримує браузер. Для цього використовується Python разом з бібліотекою Gradio,

яка дозволяє швидко розробляти вебінтерфейси для моделей машинного навчання та обробки даних.

У якості середовища розробки обрано Visual Studio Code через його безкоштовність, розширюваність, крос-платформенність та швидкість. VS Code також має вбудовану підтримку Python, що дозволяє зручно працювати з цією мовою.

Система контролю версій обрана GitHub, яка надає зручність у спільній роботі над проектом та забезпечує надійне зберігання версій програмного забезпечення.

У розробці дизайну використовується Figma, що надає широкий функціонал та можливість спільної роботи над проектом в реальному часі.

Вибір цих технологій забезпечує ефективність, продуктивність та зручність у процесі розробки програмного забезпечення, а також гарантує доступність та високу якість системи для користувачів.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРЕКЛАДУ

3.1 Дизайн програмного забезпечення для перекладу відео

Розробка будь-якого продукту зазвичай розпочинається з етапу створення макету, дизайну який визначає структуру та візуальний вигляд майбутнього інтерфейсу, а також його функціональність.

В області дизайну користувацьких інтерфейсів (UX-дизайн) існує декілька видів макетів: вайрфрейм, мокап та прототип. Кожен з цих видів макетів відображає перспективу майбутнього продукту з різних аспектів: структурного розташування елементів, візуального оформлення та функціональності відповідно. Розробка макетів зазвичай відбувається послідовно, починаючи з більш загальних та абстрактних макетів і переходячи до більш деталізованих і конкретних.

У випадку створення дуже простого продукту з одним можливим користувацьким сценарієм, можна обмежитися лише створенням мокапу або вайрфрейму разом з мокапом. Однак у складніших проектах процес проєктування включає всі три етапи розробки макетів.

Ефективне розташування елементів інтерфейсу є ключовим для підвищення зручності використання продукту та важливо для того щоб зробити його більш привабливим для користувачів. Важливо чітко визначити очікування від програми та відображення необхідних елементів на кожному екрані.

У процесі розробки інтерфейсів для застосунків використовується створення макетів, яке спрямоване на оптимізацію витрат часу та ресурсів. Вже на початкових етапах проекту складно досягти ідеального продукту, який би повністю задовольняв потреби користувачів. Макети надають можливість тестувати обрані рішення з невеликими витратами і вносити зміни за необхідності перед розпочатком процесу розробки та програмування.

Створення макетів не лише сприяє визначенню та розвитку основного напряму дизайну, але й дозволяє економити значну кількість часу. Якщо на створення концепції може бути витрачений один день, то на розробку та дизайн не

доведеться витрачати цілий тиждень. Однак це не єдиний мотив для виділення часу на макетування.

Процес макетування вирішує кілька важливих завдань. По-перше, він сприяє пошуку найкращих ідей, оскільки макети можуть бути розроблені значно швидше, надаючи можливість підготувати кілька варіантів для перевірки гіпотез та вибрати найбільш успішний серед них. По-друге, макети допомагають виявляти помилки на етапі їх виявлення, що дозволяє зекономити час, кошти та зусилля на їх подальше виправлення на ранніх етапах розробки. Крім того, макети дозволяють оцінити зручність інтерфейсу та провести тестування користувацьких сценаріїв, що є важливою перевагою на ранніх стадіях проектування.

Вайрфрейм використовується для позначення раннього концептуального макету скелетної структури продукту [47]. Він служить для швидкого візуалізування та фіксації основних ідей щодо організації компонентів інтерфейсу, забезпечуючи орієнтацію для команди розробників та визначаючи загальний каркас, на якому подальше розроблення вебдодатку або сайту буде ґрунтуватися.

Створення вайрфрейму передбачає врахування цілей продукту, цільової аудиторії та аналізу конкурентів. Для простих проектів це може відбуватися без великого обсягу попередніх досліджень, але для більш складних проектів необхідно скласти технічне завдання з визначенням необхідних функцій. При цьому ступінь деталізації може бути різним.

Низькодеталізований вайрфрейм часто використовується на початкових етапах проекту та може бути швидко складений під час брейнштормінгу або з командою, зафіксований шляхом швидкого скетчу від руки. Однак в такому випадку розуміння структури та взаємозв'язків елементів може бути відносно обмеженим без додаткової пояснювальної презентації, оскільки розкриття контексту вимагатиме участі автора вайрфрейму.

Високодеталізований вайрфрейм представляє собою більш докладний план, де кожен елемент має підписи та пояснення значень. Такий вайрфрейм може бути

показаний клієнту без додаткових пояснень, оскільки він вже містить достатньо інформації для розуміння структури та функціоналу інтерфейсу.

Необхідно створити простий у використанні інтерфейс, який би був доступним і зрозумілим навіть для початківців. Такий інтерфейс має бути розроблений з урахуванням вимог зручності та естетики, щоб привернути увагу користувачів, які шукають інноваційні та стильні рішення. Для розробки початкового макету було обрано програмне забезпечення Figma, яке надає широкий набір функцій для створення гарних та зручних інтерфейсів, а також дозволяє оптимізувати процес їх розробки завдяки інтуїтивно зрозумілому інтерфейсу та ефективним інструментам дизайну.

Для створення застосунку спочатку було розроблено низькодеталізований вайрфрейм. Ціллю було зробити простий мінімалістичний застосунок, щоб будь-хто зміг розібратись в ньому. На рис. 3.1 зображений результат.

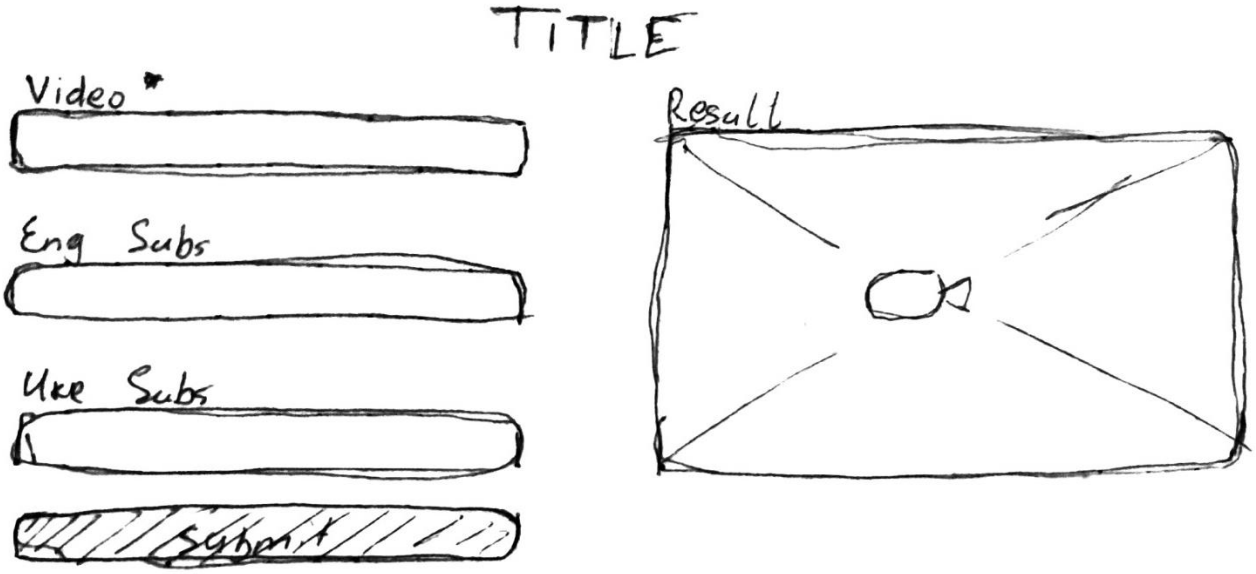

Рисунок 3.1 – Низькодеталізований вайрфрейм застосунку

Далі був створений Високодеталізований вайрфрейм, з чітким текстовим контентом і блоками. Його можна побачити на рисунку 3.2.

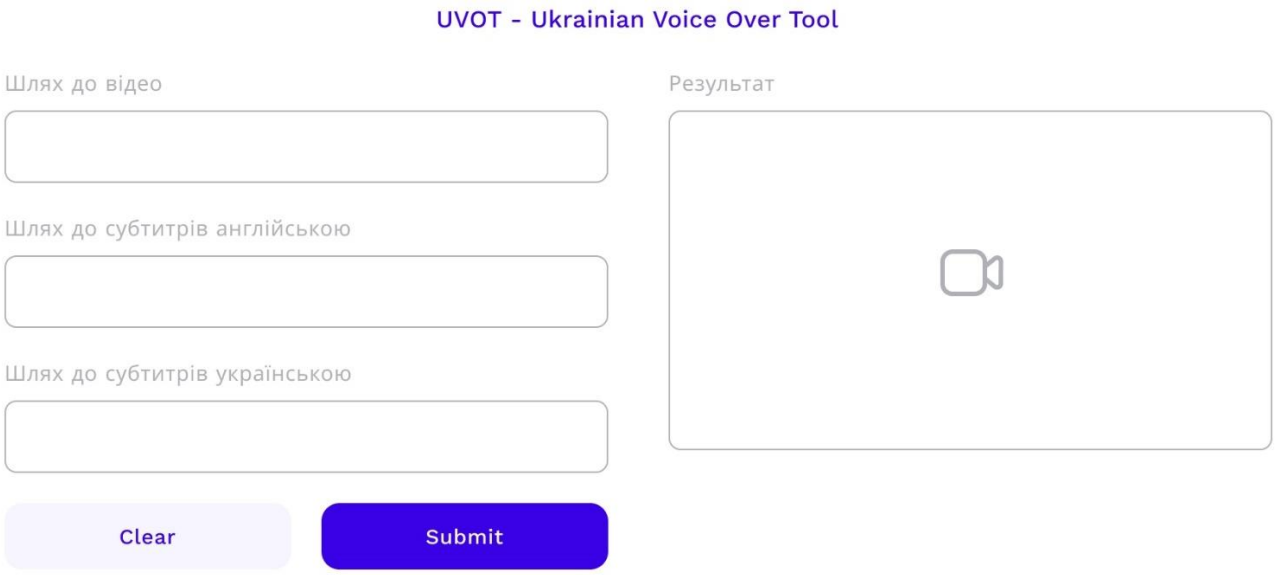

Рисунок 3.2 – Високодеталізований вайрфрейм

Після завершення тестування та затвердження остаточної структури розпочинається розробка дизайну інтерфейсу. Задачею дизайнерів є не лише створення привабливого зовнішнього вигляду продукту, але й розуміння його психологічного впливу на користувача. Наприклад, відчуття, що виникають при використанні інтерфейсу з яскравими кольорами, можуть бути значно відмінні від тих, які виникають при використанні інтерфейсу з темними, стриманими кольорами. Це лише кілька прикладів рішень, які розглядають дизайнери інтерфейсу користувача.

Дизайн інтерфейсу є ключовим елементом в створенні сприятливої атмосфери для користувачів програмного забезпечення. Колірна палітра, форми, шрифти та кнопки, які використовуються після запуску програми, можуть вплинути на сприйняття користувача та його бажання продовжувати використання продукту або видалити його. Важливо, щоб атмосфера інтерфейсу привертала увагу користувача від першого взаємодії.

Гарний дизайн інтерфейсу користувача полегшує виконання поставленого завдання, не привертаючи до себе зайвої уваги. Графічний дизайн і типографіка використовуються для підтримки зручності використання, впливаючи на те, як

користувач виконує певні взаємодії та покращуючи естетичну привабливість дизайну; естетика дизайну може покращити або зменшити можливість користувачів використовувати функції інтерфейсу [48].

Для розробки дизайну також було використано програмне забезпечення Figma, яке забезпечує зручний та ефективний процес створення інтерфейсів завдяки широкому спектру функцій та інтуїтивно зрозумілому інтерфейсу.

Розроблений фінальний дизайн застосунку показано на рисунку 3.3.

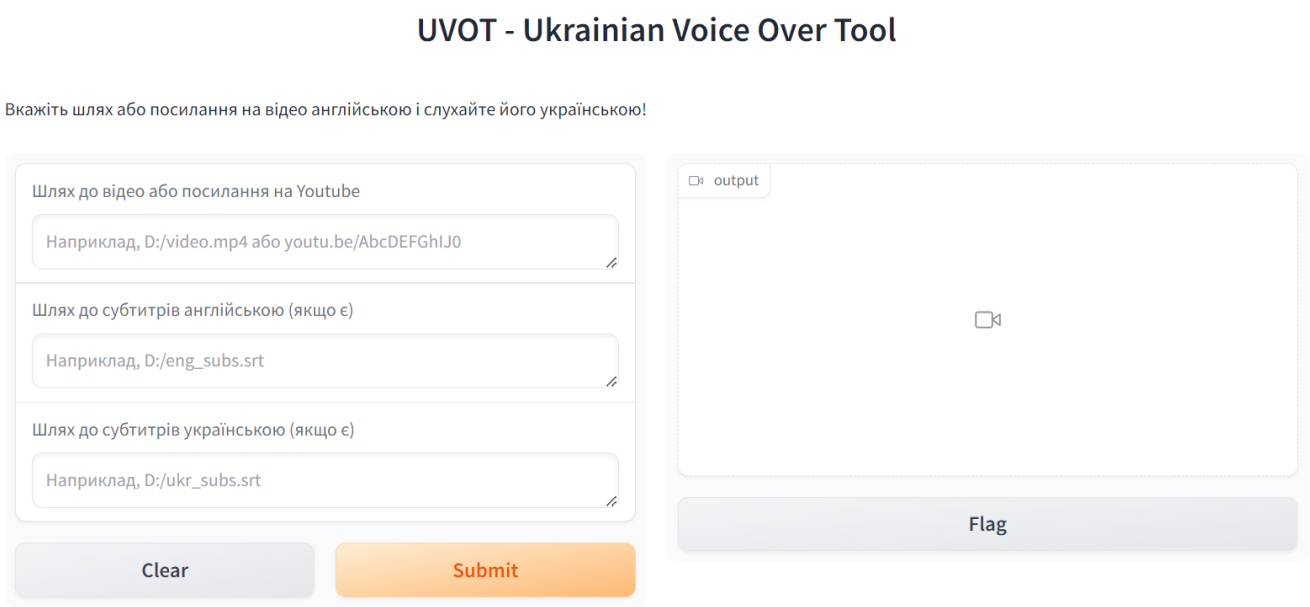

Рисунок 3.3 – Фінальний дизайн застосунку

Відповідно до представленого на рисунку 3.3, можна зазначити, що для отримання перекладеного відео достатньо ввести шлях до відео або посилання на YouTube, й натиснути кнопку «Submit». Додатково, для більш точного налаштування результату, користувач може самостійно вказати субтитри оригіналу, що відобразиться на згенерованому відео. Або, у випадку наявності, можна вказати вже перекладені субтитри, що спростить роботу програми до синтезу вказаного тексту та обробки аудіо.

Зазначений дизайн відзначається своєю простотою та зрозумілістю. Весь функціонал зосереджено на одній сторінці браузера, що дозволяє користувачеві швидко ознайомитися з можливостями програми. Колірна палітра проста і приваблива. Акцент лише на одній кнопці «Submit» робить використання легким.

3.2 Архітектура програмного забезпечення

Після завершення етапу прототипування та проектування, наступним кроком в розробці програмного забезпечення є створення його архітектури. Архітектура програмного забезпечення представляє собою стратегічний план створення та підтримки програмної системи, який визначає її структуру, розташування компонентів та надає вказівки для безперебійної роботи.

Якщо програмна система має добре пророблену архітектуру, це може сприяти досягненню високої продуктивності та надійності. Проте, у випадку поганого проектування системи можуть виникнути декілька поширених проблем:

– *складність*. Погано розроблена архітектура робить систему важкою для зрозуміння та підтримки, що може призвести до неефективності та втрати часу;

– *крихкість*. Нестабільна та помічена помилками система, яка має недоліки у своїй архітектурі, може стати схильною до аварійного зупинення та втрати функціональності при виникненні проблем з одним з її компонентів;

– *складні взаємозалежності*. Недоліки у архітектурі зазвичай призводять до складних взаємозалежностей між різними компонентами системи. Це може призвести до швидкого поширення проблем та збитків.

Отже, важливість правильної архітектури програмного забезпечення важко переоцінити, оскільки вона визначає майбутню ефективність, надійність та стійкість програмної системи.

Для розробки архітектури застосунку буде використана діаграма прецедентів (Use Case діаграма) у межах уніфікованої мови моделювання (UML). UML є стандартом, який дозволяє моделювати різні аспекти системи, відображаючи об'єкти та їх взаємодії [49]. Термін "уніфікований" вказує на застосування цієї мови

у широкому спектрі сфер, включаючи проектування систем, різні галузі застосування, організаційні структури та рівні складності проектів. UML визначає об'єкти за єдиним синтаксисом, що спрощує спілкування між учасниками проекту незалежно від їхнього місцезнаходження.

Деякі типи діаграм UML є специфічними для конкретних систем і програм. Діаграми прецедентів в рамках UML дозволяють узагальнити інформацію про користувачів системи, їх взаємодію з нею та цілі, які вони преслідують через цю взаємодію. Ці діаграми використовують спеціальний набір символів та з'єднувачів для моделювання цих взаємодій.

Ефективне використання діаграм прецедентів може полегшити комунікацію в команді розробників, дозволяючи їм обговорювати та демонструвати сценарії взаємодії системи з користувачами, організаціями або зовнішніми системами. Вони надають високорівневий огляд взаємодії між варіантами використання, акторами та системами, що сприяє зрозумінню обсягу та функціональності системи.

Використання діаграм прецедентів доповнює більш детальний текстовий опис варіантів використання, адже вони надають візуальне представлення цих взаємодій [50]. На діаграмах прецедентів актори зображуються у вигляді фігур, а взаємодія між актором та прецедентом позначається лінією. Щоб вказати межі системи, використовується прямокутник, який обмежує прецедент. Такий підхід допомагає розуміти структуру та функціональність системи у візуальній формі.

Нижче наведена діаграма прецедентів застосунку. На ній зображений користувач, якому щоб отримати результат у вигляді перекладеного відео, потрібно його завантажити. Далі застосунок сам дістане аудіо, розпізнає текст, перекладе, озвучить, обробить аудіо і надасть фінальний результат.

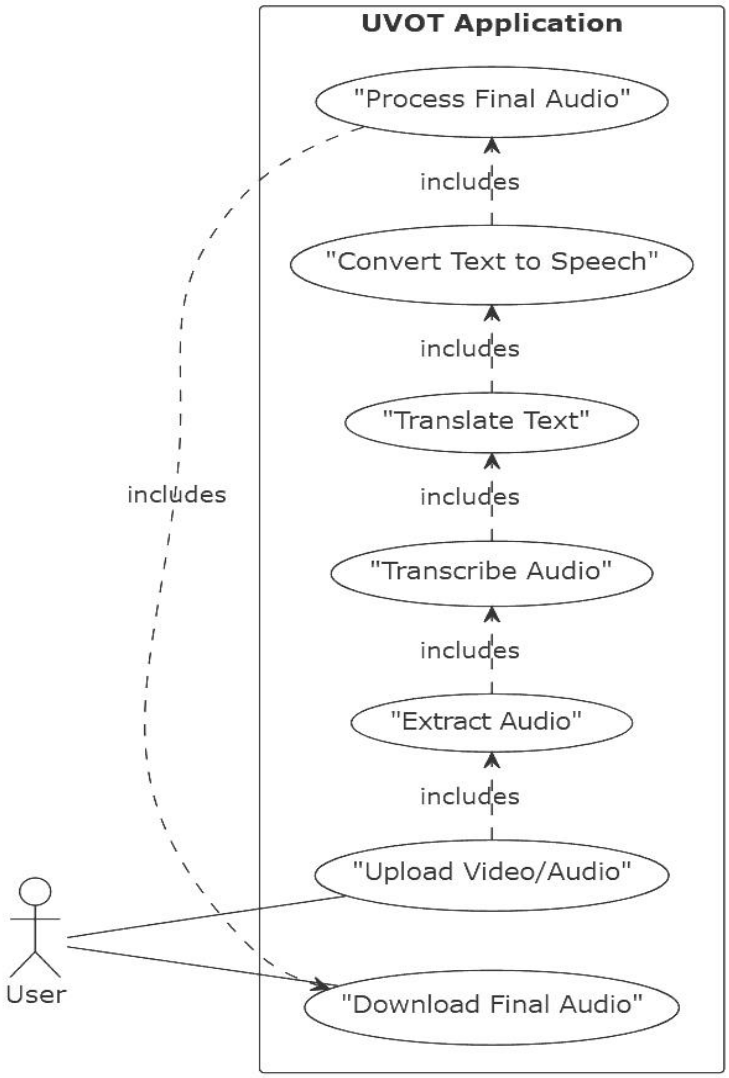

Рисунок 3.4 – Діаграма прецедентів

Далі, для розробки застосунку після створення діаграми прецедентів буде корисним створення діаграми послідовності (Sequence Diagram). Діаграма послідовності – це тип діаграми в рамках уніфікованої мови моделювання (UML), яка відображає послідовність взаємодій між об'єктами в системі в часі [51]. Ця діаграма ілюструє, як об'єкти взаємодіють один з одним в рамках конкретного сценарію використання чи дії.

Головна мета діаграми послідовності полягає у візуалізації потоку виконання операцій та обміну повідомленнями між об'єктами системи впродовж часу. Це дозволяє розробникам та аналітикам краще розуміти, як система працює на рівні

об'єктів та які взаємодії відбуваються між ними. Діаграма послідовності може допомогти виявити потенційні проблеми у логіці програми або у послідовності операцій. Основні переваги діаграми послідовності наступні:

– *візуалізація послідовності дій*. Діаграма послідовності надає чітку візуальну репрезентацію того, як об'єкти взаємодіють один з одним у реальному часі;

– *виявлення проблем та недоліків*. Аналіз діаграми послідовності може допомогти виявити можливі проблеми у логіці програми, недоліки в послідовності операцій або відсутність необхідних взаємодій між об'єктами;

– *комунікація*. Діаграма послідовності може служити інструментом комунікації між учасниками проекту, дозволяючи краще зрозуміти взаємодію об'єктів у системі;

– *документація*. Вона може бути корисною частиною документації проекту, яка описує взаємодії між об'єктами та порядок виконання операцій.

Діаграма послідовності, як ключовий інструмент в рамках уніфікованої мови моделювання, відіграє важливу роль у розробці програмних застосунків. Її використання сприяє уникненню потенційних проблем, які можуть виникнути під час взаємодії об'єктів у системі. Особливо ситуацій, коли очікуваний результат відрізняється від фактичного. Шляхом чіткого відображення послідовності операцій та обміну повідомленнями між об'єктами, вона дозволяє розробникам та аналітикам аналізувати потенційні точки невідповідності та вчасно коригувати їх.

Діаграма послідовності є потужним інструментом, який допомагає забезпечити якість та ефективність програмного застосунку, а також сприяє покращенню комунікації та співпраці у рамках розробки проекту.

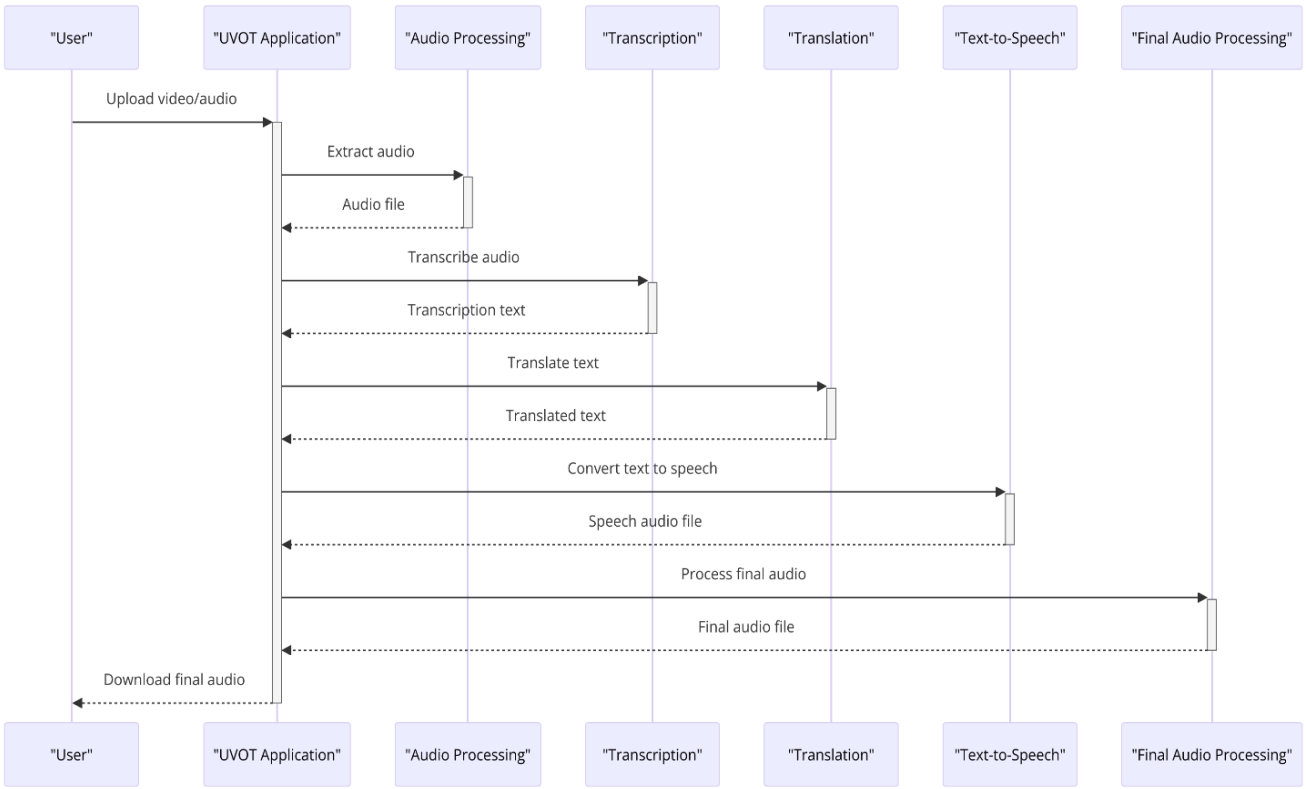

Рисунок 3.5 – Діаграма послідовності

Створена діаграма послідовності окреслює операційний потік створеного застосунку. Ця діаграма використовує нотацію діаграми послідовності Уніфікованої мови моделювання для артикуляції часових взаємодій між різними компонентами системи та користувачем. Основна мета цієї діаграми – надати чітку, покрокову візуалізацію послідовності операцій, ініційованих дією користувача, і того, як ці операції обробляються різними модулями системи для отримання кінцевого результату.

Починає все взаємодія з користувачем. Процес починається з завантаження користувачем відео- або аудіофайлу в систему, що означає початок взаємодії між користувачем і застосунком.

Далі, отримавши мультимедійний файл, система перенаправляє його до модуля «Обробка аудіо». Цей модуль відповідає за вилучення аудіо компоненту з завантаженого контенту, гарантуючи, що подальші операції виконуються виключно з аудіо даними.

Витягнутий аудіофайл передається до модуля «Транскрипція». Цей модуль перетворює аудіоконтент у текстову форму, створюючи транскрипцію розмовного контенту в аудіофайлі. Після транскрибування текстові дані передаються до модуля «Переклад». Цей модуль перекладає транскрибований текст на визначену цільову мову, полегшуючи доступ до контенту для ширшої аудиторії.

Перекладений текст проходить ще одну трансформацію в модулі «Текст до мовлення», де він знову перетворюється в формат аудіо. Однак тепер це аудіо вже мовою перекладу, що фактично створює дубльовану версію оригінального контенту.

Перед завершенням процесу дубльований аудіо файл піддається фінальній обробці. Цей етап об’єднує синтезовану доріжку українською з аудіо розмов англійською, зробивши його тихіше де потрібно. Далі об’єднує їх з фоновими звуками. Оброблений аудіофайл додається до відео і стає доступним для завантаження користувачем, позначаючи кінець послідовності. Цей файл є кульмінацією серії складних операцій, призначених для транскрибування, перекладу та дублювання аудіоконтенту іншою мовою.

Таким чином, діаграма послідовності забезпечує структуроване і детальне представлення процедурної логіки, притаманної розроблюваному додатку, висвітлюючи послідовний і логічний потік операцій від початкового введення даних користувачем до кінцевого результату. Ця візуалізація не тільки допомагає зрозуміти функціональність системи, але й слугує основою для подальшого розвитку, оптимізації та досліджень у галузі обробки мультимедійних даних та мовних технологій.

Висновки до розділу 3

У третьому розділі детально проаналізовано кожен етап створення інтелектуальної системи перекладу відео, включаючи як дизайн, так і архітектуру програмного забезпечення.

Перш ніж перейти до розробки програмного забезпечення, розглянуто процес проектування інтерфейсу. Макети (wireframe) були використані для визначення структури та візуального вигляду інтерфейсу програми. Починаючи з низькодеталізованих макетів і переходячи до більш деталізованих варіантів, було забезпечено оптимальну організацію елементів інтерфейсу та привабливий вигляд програми. Такий підхід дозволяє забезпечити зручність використання програми та привабливий зовнішній вигляд, що важливо для користувачів.

Далі в розділі розглянуто архітектуру програмного забезпечення. Використовуючи методологію Unified Modeling Language (UML), були побудовані діаграми використання та послідовності, що деталізували процес взаємодії користувача з програмою та логіку її роботи. Такий підхід дозволяє створити стратегічний план для створення та підтримки програмного забезпечення, що визначає його структуру та забезпечує належну роботу.

Зазначені етапи розробки є важливою передумовою для успішного створення інтелектуальної системи перекладу відео з англійської на українську, яка забезпечить зручний та ефективний переклад контенту для користувачів, які погано розуміють англійську мову. Ретельне проектування і розробка інтерфейсу та архітектури програмного забезпечення дозволяють створити продукт, що відповідає вимогам користувачів і забезпечує найвищу якість обслуговування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СТВОРЕННОЇ СИСТЕМИ ДЛЯ ПЕРЕКЛАДУ ВІДЕО

4.1 Опис програмної реалізації системи перекладу відео

Розглянемо як реалізована система.

1. Файл separate_video_audio_subs.py.

Цей скрипт призначений для відокремлення відео, аудіо та субтитрів із заданого вхідного файлу. Розглянемо його детальніше.

Скрипт використовує бібліотеку ffmpeg, потужний інструмент для обробки мультимедійних даних. Він дозволяє працювати з медіа файлами і надає можливість розділити відео на потоки, що допоможе розділити аудіо та відео без перекодування.

Функція separate_video_audio_subs(input_file) – основна функція, яка приймає вхідний файл і обробляє його. Все починається з аналізу вхідного файлу за допомогою ffmpeg.probe. Якщо під час цього процесу виникає помилка, вона виводить помилку і повертається.

Далі скрипт виконує ітерації над кожним потоком у вхідному файлі. Він перевіряє наявність аудіо потоків англійською (eng) та українською (ukr) мовами, а також потоків субтитрів. Для кожного звукового потоку він витягує і зберігає звук у форматі WAV. Для субтитрів програма перевіряє мову та формат (наприклад, srt або ass) і зберігає найбільший файл субтитрів для кожної мови. Це зроблено для випадків, коли надаються субтитри з усіма діалогами та субтитри тільки з написами.

2. Файл youtube_download.py.

Цей скрипт використовується для завантаження відео, аудіо та субтитрів з YouTube. Розглянемо як він працює.

Функція download_youtube_video(url) отримує URL-адресу YouTube і завантажує відео, аудіо та субтитри.

Спочатку перевіряє, чи існує каталог splited_video, і створює його, якщо ні. Далі скрипт використовує yt-dlp, програму командного рядка для завантаження відео з YouTube. Він завантажує відео найкращої якості та англійський звук у форматі WAV. Далі завантажуються англійські та українські субтитри у форматі SRT. Якщо субтитри згенеровані автоматично, програма пропускає завантаження. Якщо завантажені субтитри мають інший формат (наприклад, vtt), код конвертує їх у формат SRT за допомогою pysubs2.

3. Файл transcribe_audio.py.

Цей скрипт призначений для транскрибування аудіофайлів. Він використовує пакет subsai – інструмент для автоматичного створення субтитрів за допомогою whisper і похідних бібліотек. Також дозволяє більш точно синхронізувати отриманий результат.

Функція transcribe_audio(audio_file) приймає аудіофайл і транскрибує його.

Спочатку код ініціалізує об'єкт SubsAI і створює модель для транскрибування, вказуючи тип моделі (наприклад, large-v2). Далі аудіофайл транскрибується, а отримані субтитри автоматично синхронізуються з аудіо за допомогою Tools.auto_sync. Синхронізовані субтитри зберігаються у форматі SRT.

4. Файл normalize_subs.py.

Цей скрипт призначений для нормалізації файлів субтитрів, щоб забезпечити їхню узгодженість і читабельність. Також це необхідно щоб надалі його було простіше перекласти і краще синтезувати звук. Давайте заглибимося в його функціональність.

Скрипт використовує json для обробки даних JSON, pysubs2 для маніпуляцій з субтитрами і re для операцій з регулярними виразами, необхідних для обробки тексту.

Основна функція normalize_subs(subs_file) починається із завантаження файла субтитрів за допомогою pysubs2, бібліотеки, призначеної для обробки субтитрів. Скрипт ітераційно переглядає кожен рядок субтитрів, застосовуючи регулярні вирази для очищення і нормалізації тексту. Це включає видалення

спеціального форматування субтитрів, коментарів у квадратних дужках, видалення непотрібних символів і забезпечення однорідної структури тексту.

Також код фільтрує рядки, що не відповідають стилю «Default», і видаляє різні події. Оброблені субтитри зберігаються у новому файлі, забезпечуючи чистіший і стандартизований результат. Приклад роботи наведено на рисунку нижче, на якому видно, що тексти в квадратних дужках видалені, переноси видалені, пусті рядки теж убрані.

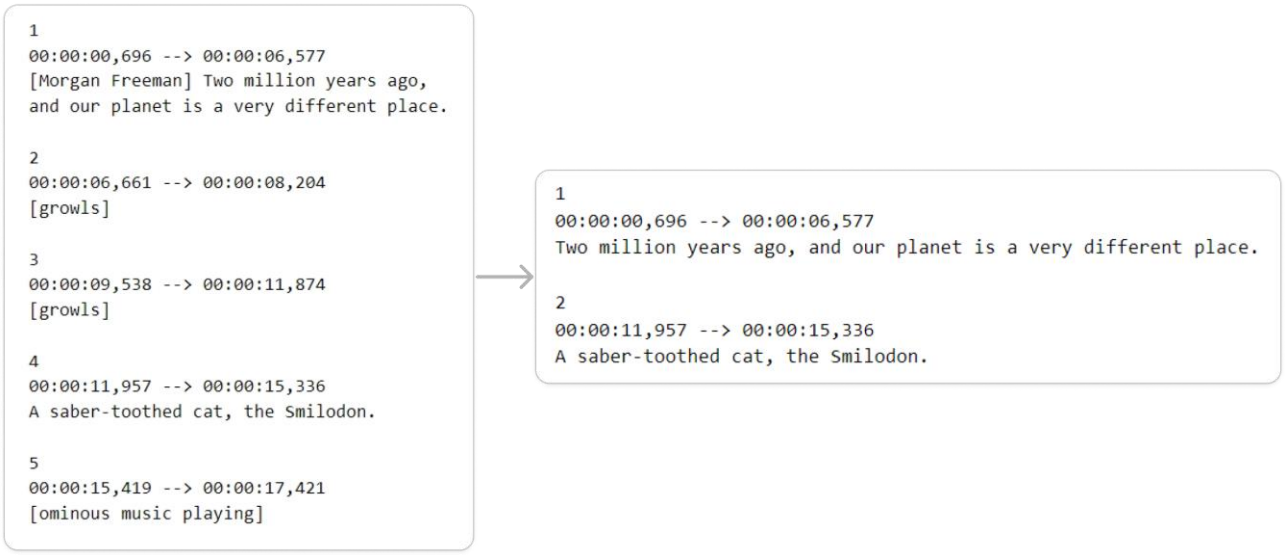

Рисунок 4.1 – Приклад роботи нормалізації субтитрів

Додаткова функція – json_to_srt(file_path) – призначена для перетворення даних субтитрів у форматі JSON у формат SRT, який є стандартом для файлів субтитрів. Це потрібно при використанні whisper, так як в чистому вигляді він на виході видає такий формат.

Спочатку код читає дані JSON, створює новий файл субтитрів і заповнює його подіями субтитрів на основі даних JSON, конвертуючи мітки часу у відповідний формат. Скрипт містить кілька допоміжних функцій для більш тонкої обробки субтитрів, таких як визначення повних речень, знаходження кінців речень і налаштування синхронізації субтитрів для кращої синхронізації зі звуком.

5. Файл translate_text.py.

Цей скрипт перекладає субтитри, на українську мову. Він використовує модуль translatesubs для перекладу. Це інструмент для перекладу субтитрів до фільмів з однієї мови на іншу, або навіть для спільного показу субтитрів кількома мовами. Інструмент працює на основі Google Translate, тому, переклад підтримує широкий спектр мов.

Основна функція translate_text(subs) створює команду для виклику translatesubs, вказуючи вихідну і цільову мови для перекладу.

Принцип роботи перекладу зображений на рисунку нижче.

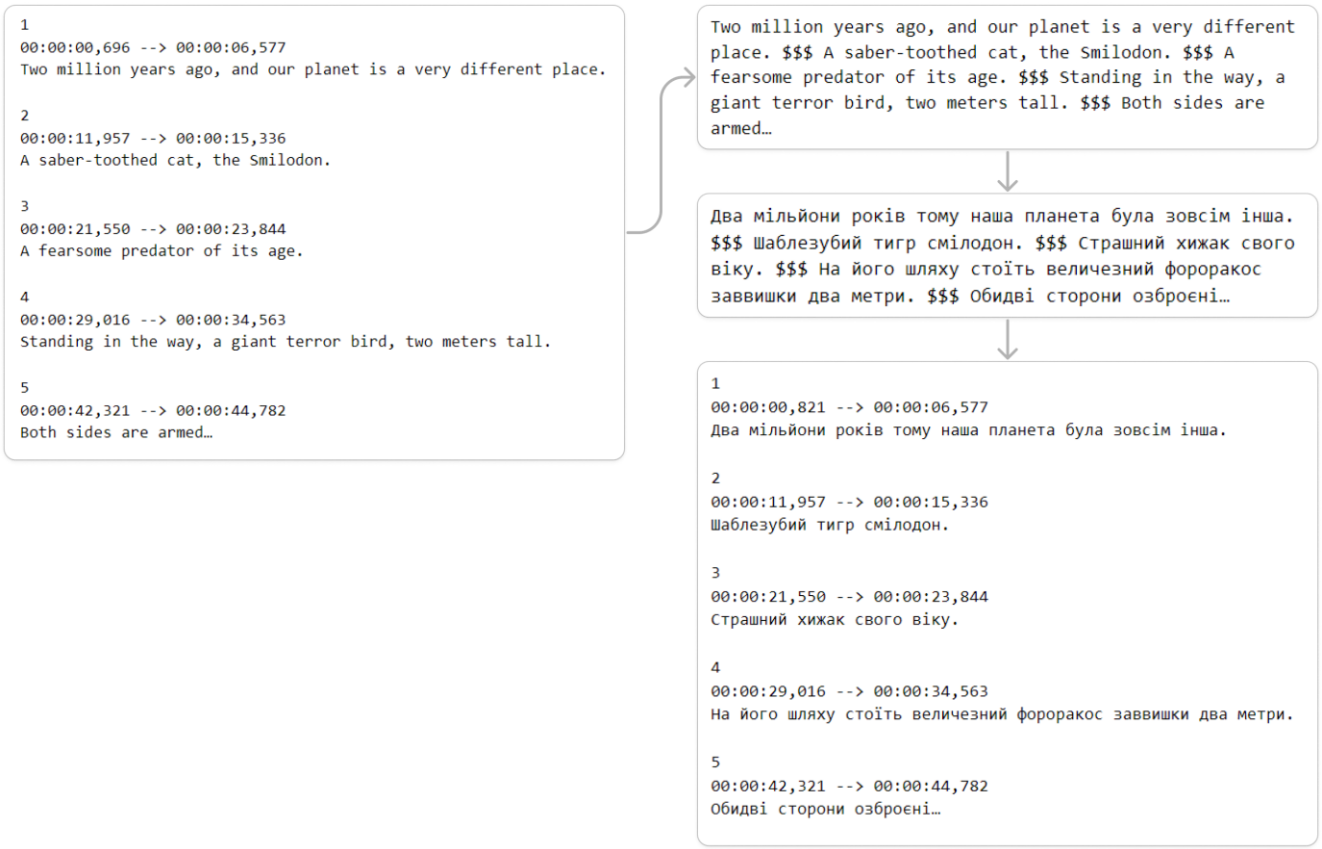

Рисунок 4.2 – Принцип роботи перекладу

З рис. 4.2 видно, що оригінальний текст спочатку збирається в один суцільний, зі спеціальним розподільником. Потім так він і перекладається, і якщо кількість розподільників однакова, то розбиває результат по субтитрам. Це

зроблено для того, щоб перекладач мав попередній контекст. Тому що якщо перекладати по одному реченню, то переклад отримується гірше. Також, іноді в субтитрах одне речення розбивається на декілька, що може при перекладі втратити початкову думку.

6. Файл split_audio.py.

Цей скрипт призначений для розділення аудіо на вокальні та інструментальні доріжки. Спочатку він інтегрує різні бібліотеки, такі як os, audio_separator, pydub, ffmpeg та time для комплексної обробки аудіо.

Основна функція split_audio(ім'я_файлу) ініціалізує audio_separator з певними параметрами, включаючи шлях до аудіофайлу, вихідні шляхи та назву моделі. Після цього обрана модель розділяє аудіо на окремі вокальні та інструментальні файли.

Інша функція, convert_to_2_channels, призначена для перетворення аудіо файлів у 2-канальний формат, щоб не виникало проблем з використанням моделі розподілення аудіо.

Також є функція split_6ch_audio(file_name), яка з шести каналів витягує третій канал, що зазвичай і є каналом з діалогами. Такий підхід дозволяє швидко дістати доріжку з мовленням без використання спеціальної моделі. Це економить час та дає більш якісний результат. Але не завжди доріжки шестиканальні.

Скрипт також містить функцію для тестування різних моделей розділення звуку, вимірювання їхньої продуктивності та часу виконання, демонструючи експериментальний підхід до обробки звуку.

7. Файл tts.py.

Цей файл призначений для синтезу тексту в мовлення (TTS), а саме для перетворення тексту з субтитрів у розмовне аудіо. Тут використовується бібліотека ukrainian_tts.tts для синтезу TTS, pydub для роботи зі звуком, os для взаємодії з операційною системою та re для регулярних виразів. Розглянемо докладніше компоненти коду.

Спочатку скрипт ініціалізує об'єкт TTS, вказуючи пристрій (CPU або GPU) для обробки. Функція synthesize_audio(text, output_filename) приймає текст і ім'я вихідного файлу, синтезує вказаний текст, використовуючи заданий голос і тип визначення наголосів, і зберігає результат як аудіофайл.

Функція read_subtitles_file(subtitles_file) читає файл субтитрів і повертає його рядки, полегшуючи обробку тексту субтитрів.

Функція split_subtitles_into_phrases(subtitles_lines) розбиває рядки субтитрів на фрази, корисні для сегментації тексту для TTS.

Функція combine_audio_files(phrases) об'єднує синтезовані аудіофайли в одну звукову доріжку, на основі часових міток субтитрів, вставляючи тишу там, де це необхідно для узгодження з оригінальним хронометражем субтитрів.

Функція generate_audio_fragments(subtitles_file) генерує аудіофрагменти для кожної фрази в субтитрах, синтезуючи аудіо для кожного сегмента, використовуючи попередні функції.

Функція combine_audio_fragments(subtitles_file) об'єднує всі аудіофрагменти в один аудіофайл, вирівнюючи його за хронометражем з оригінальними субтитрами.

Функція speed_up_audio_in_folder (folder_path, speedup_percentage) прискорює всі аудіофайли у вказаній теці на заданий відсоток, регулюючи швидкість відтворення. Заведено так, що всі аудіо прискорюються на 15%. Згенероване аудіо досить повільне, а прискорення дозволяє більше насолодитись оригінальною звуковою доріжкою.

8. Файл adjust_audio_volume.py.

Цей скрипт зосереджений на регулюванні гучності аудіодоріжок. Для обробки аудіо використовується pydub – універсальна бібліотека для роботи з аудіо.

Функція adjust_audio_volume(audio_path, vocals_path) регулює гучність оригінальної доріжки відносно згенерованої звукової доріжки українською. Спочатку функція визначає тихі сегменти в аудіо українською мовою та робить

тихішими ті фрагменти оригінального аудіо, де відсутня тишу в українському аудіо.

Далі код поєднує скориговану оригінальну доріжку та згенеровану звукову доріжку, роблячи озвучення окраїнською. Фінальне відкориговане аудіо зберігається у вказаний шлях.

9. Файл combine_all.py.

Цей скрипт використовується для об'єднання отриманих аудіо та відео файлів в один вихідний файл. Він знов використовує ffmpeg для обробки відео та аудіо, а pydub – для маніпуляцій з аудіосегментами.

Функція get_first_audio_format (input_video) визначає формат першого аудіопотоку у відеофайлі, щоб надалі використовувати його і максимально зберегти оригінальну якість.

Функція combine_all(input_video, result_name) конвертує аудіо у формат, сумісний зі звуковим потоком відео, а потім поєднує відео з новою звуковою доріжкою. Об'єднані відео та аудіо експортуються у вказаний вихідний файл.

Також є додаткові функції. Combine_2ch_audio() об'єднує аудіо у двоканальний файл. Make_3ch(vocals_adjusted, combined_audio, result) для об’єднання файлу, який буде використаний як третій канал у шестиканальному аудіо. Combine_6ch_audio(ch6_audio, ch1_audio, output) об'єднує шестиканальний аудіо файл з моно аудіофайлом, вирівнюючи розбіжності у довжині.

Так, у комбінації split_audio, tts, adjust_audio_volume, combine_all можна обробити звук для отримання фінального аудіо. На рис. 4.3 візуально зображено принцип роботи обробки звуку.

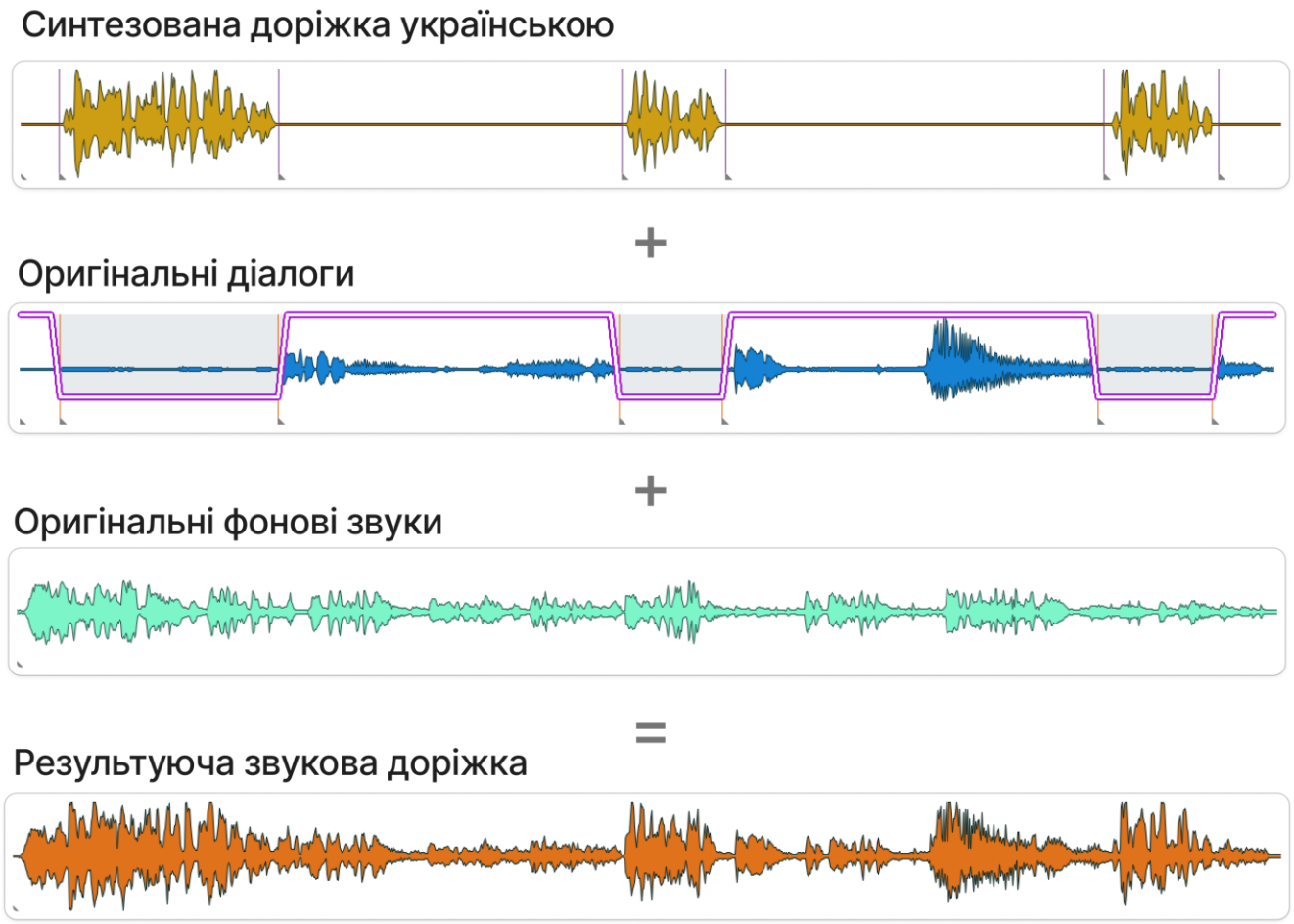

Рисунок 4.3 – Принцип обробки звуку

Простими словами, робиться синтез українських реплік, потім оригінальна доріжка розділяється на фонові звуки та звуки розмов. Потім, у часових фрагментах де в українських репліках є звук, в тих же фрагментах оригіналу звук плавно знижається. Таким чином оригінал чутно тихіше коли йде розмова українською. І якщо поєднати українські репліки, заглушені оригінальні репліки та фонові шуми без змін, отримується фінальна звукова доріжка.

Також присутня логіка накладання аудіо. Бувають випадки, коли переклад значно довше або коротше оригінальної фрази. Зі вторим випадком проблем немає – фраза українською завершується, а англійська продовжується в оригінальному темпі. А коли переклад довше за оригінал, і якщо це відбувається декілька разів, то існує ймовірність що доріжки українською перекриють одна на одну. Для цього можна було б прискорити синтезовану доріжку, проте тоді втрачається темп, і часто

це звучить погано. Тому в програмі синтезовані доріжки всі дещо пришвидшені, щоб такі випадки були рідше, і темп був всюди однаковий. І якщо вже відбувається складна ситуація, то накладання не буде, і наступна фраза буде йти після попередньої. В такому випадку дещо постраждає таймінг щодо оригіналу, але на практиці це не заважає сприйняттю відео. Візуалізація цього процесу зображена на рисунку нижче.

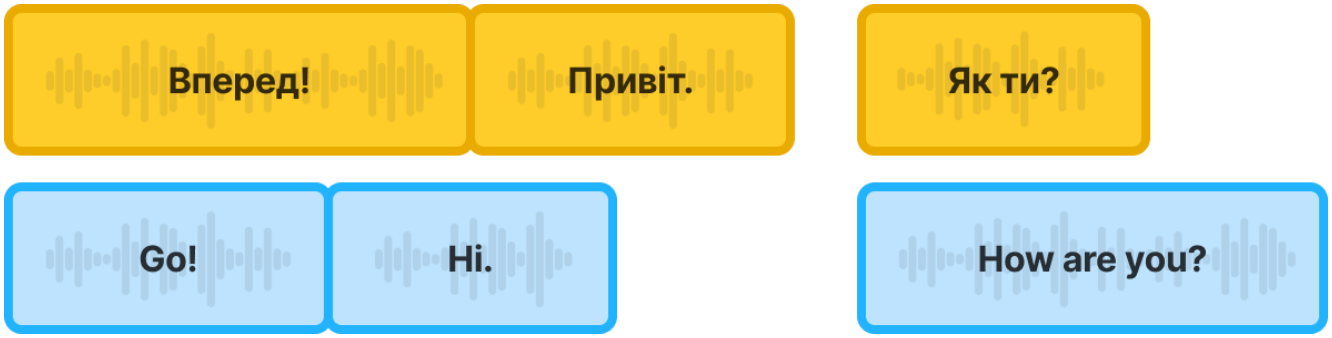

Рисунок 4.4 – Візуалізація зміщення довгих фраз українською

10. Файл find_files.py.

Цей скрипт призначений для пошуку певних файлів у каталозі. Іноді невідомо який формат файлу буде після його отримання, але відома його назва. Для пошуку цього файлу і потрібен цей код. Він використовує модуль os для взаємодії з файловою системою. Функція find_file(ім'я_файлу) задає шлях до теки, в якій він буде шукати файли. Далі код виконує ітерацію по каталогу, перевіряючи кожен файл. Якщо файл починається з вказаного імені і є файлом (а не каталогом), повертає шлях до нього. Якщо файл знайдено, повертається шлях до нього, інакше повертається значення None.

11. Файл delete_files.py.

Цей скрипт використовується для видалення тимчасових файлів із зазначених каталогів, щоб не забруднювати диск. Код використовує модуль os для операцій з файлами і каталогами.

Функція delete_temp_files() визначає список папок, з яких буде видалено файли. Потім проходить по кожній теці, отримуючи список усіх файлів. Для кожного файлу він перевіряє, чи це файл (а не каталог), а потім видаляє його.

12. Файл main.py.

Це головний код проекту, який керує різними функціональними можливостями. Він імпортує різні модулі та функції з інших скриптів проекту, а також pydub, time, gradio, os і re.

Функція main(input_video) залежно від того, чи подається на вхід шлях до файлу або посилання на YouTube, вона або відокремлює відео, аудіо та субтитри від вхідного файлу, або завантажує відео з YouTube.

Далі функція перевіряє наявність англійських субтитрів, за необхідності транскрибує аудіо, нормалізує та перекладає субтитри, а також в залежності від кількості каналів в аудіо, виконує відповідні функції для розділення аудіо файлу на мовлення та інші звуки.

Далі скрипт генерує аудіофрагменти з перекладених субтитрів і регулює їхню швидкість. Потім регулює гучність звуку оригінального аудіо та об'єднує аудіо фрагменти. Залежно від обраного шляху розділення аудіо, поєднує оброблене аудіо з оригінальним відео.

Код використовує інтерфейс Gradio для взаємодії з користувачем, дозволяючи користувачам вводити шлях до відео або посилання на YouTube і відаючи оброблене відео.

Створена програма чудово справляється з поставленою задачею – перекладом відео з англійської на українську мову. В подальшому можна вдосконалити і оптимізувати код. Далі можна додати в синтез жіночій голос і використовувати його там, де в оригіналі теж жіночий голос. Можна піти ще далі, і зробити синтез оригінальними голосами. Для цього потрібно додатки функцію ідентифікації мовця та клонування голосу на основі фрагменту. Також, можна зробити більш детальний інтерфейс для ще більш гнучкого налаштування. Загалом далі можна рухатись в напрямку не простого озвучення, а створення саме

автоматичного дубляжу відео. Проте, це може призвести до більш довгої роботи програми.

Створена програма чудово справляється з поставленою задачею, і перекладає відео з англійської на українську мову. Перспективи подальших досліджень у цій сфері наступні:

1) оптимізація алгоритму. Один із можливих напрямків – це подальша оптимізація коду програми. Швидше і ефективніше використання ресурсів обчислювального обладнання дозволить покращити продуктивність і зменшити час перекладу великих відеофайлів;

2) синтез жіночого голосу. Розширення функціоналу програми включає в себе можливість синтезу жіночого голосу для тих випадків, коли в оригіналі використовується жіночий голос. Це може зробити переклад більш природним та приємним для користувачів;

3) підтримка більшої кількості мов. Можна також подбати про переклад не тільки з англійської, а й з інших мов. Більшість інструментів підтримують багатомовність, і впровадження цього функціоналу було б не сильно складним. Також можна додати можливість перекладати не тільки на українську, але для цього слід знати й інші мови, щоб змогти перевірити на скільки якісно все працює;

4) розширений інтерфейс. Розробка більш детального інтерфейсу може дати користувачам більше можливостей для налаштування перекладу, включаючи вибір голосу, швидкість, інтонацію та інші параметри;

5) клонування голосу. Можна піти ще далі, і зробити синтез оригінальними голосами. Для цього потрібно додатки функцію ідентифікації мовця та клонування голосу на основі фрагменту. Це дозволить програмі створювати переклади з ще більшою якістю, а користувачі отримають ще більше занурення у відео;

6) автоматичний дубляж відео. Однією з найбільших перспектив є рух у напрямку створення автоматичного дубляжу відео. Це вимагатиме розробки додаткових алгоритмів для синхронізації тривалості перекладеного тексту та

оригінального. Також, потрібно точно клонувати емоції, та звукові ефекти оригінальної звукової доріжки.

Загалом, подальший розвиток програми може призвести до значного покращення якості перекладу та зробити її більш універсальною та корисною для різних видів відео контенту. Однак, слід враховувати, що більш досконалі функції можуть потребувати більше часу для обробки, тому ефективність та швидкість роботи програми також будуть важливими аспектами у подальшому вдосконаленні.

4.2 Тестування та аналіз результатів

В результаті роботи отримано вебзастосунок, який може перекласти відео у форматі mkv або mp4, а також будь-яке відео з YouTube з англійської на українську.

Для початку роботи з додатком, користувачу необхідно ввести шлях до відеофайлу або посилання на відео з YouTube у відповідному полі, що позначене як «Шлях до відео або посилання на YouTube» (див. рис. 4.5). Це може бути або локальний шлях до файлу, наприклад, «D:/Files/Video.mp4», або ж URL-посилання на відео в мережі YouTube.

Після введення інформації та натискання на кнопку «Submit», програма починає виконувати процес обробки та перекладу відео. В результаті роботи програми користувач отримує готовий відеофайл, який може бути переглянутий негайно або завантажений (див. рис. 4.6).

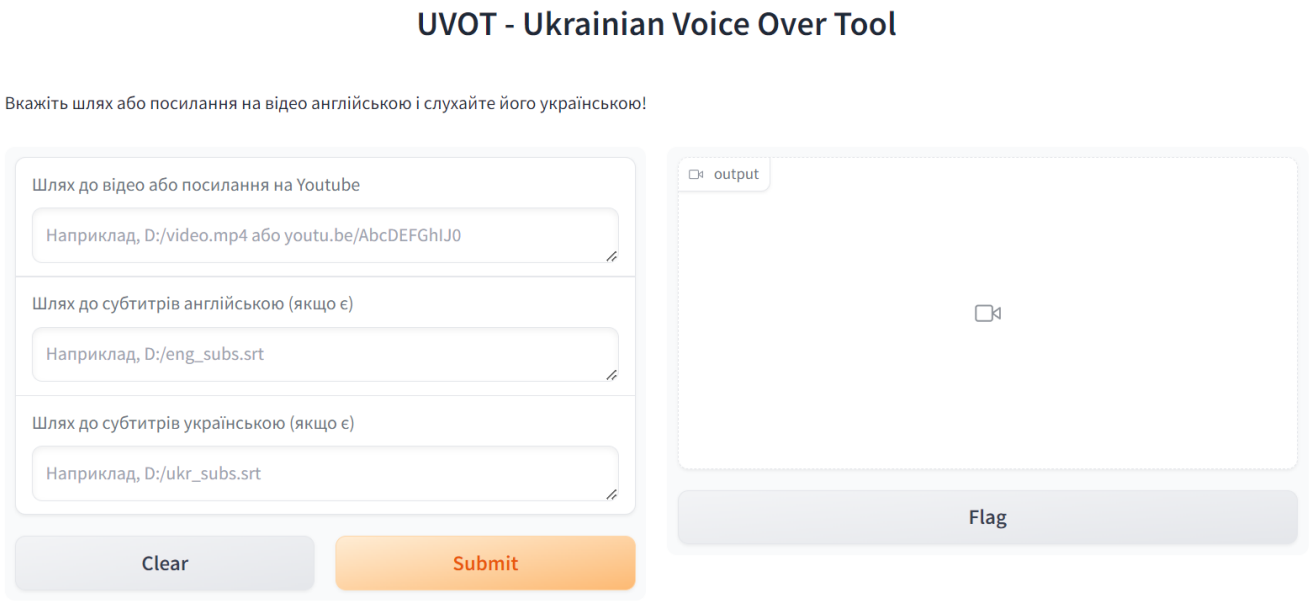

Рисунок 4.5 – Початковий інтерфейс застосунку

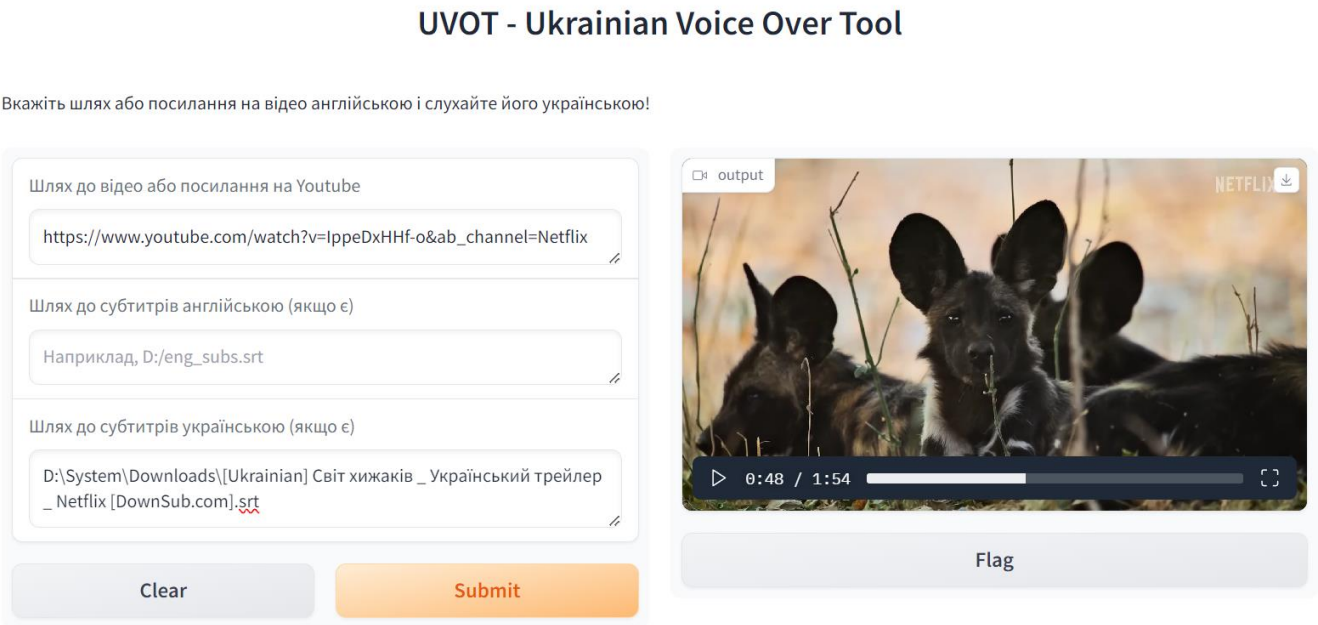

Рисунок 4.6 – Результат роботи застосунку

Користувач також має можливість самостійно вказати шлях субтитрів оригіналу чи перекладу (див. рис. 4.5). В разі обрання оригінальних субтитрів, це сприяє досягненню більшої точності перекладу та більш точних часових міток. В разі вибору перекладених субтитрів, переклад стає найточнішим, і залишається

лише синтезувати голос та обробити результат. Таким чином, користувач має можливість налаштувати результат в гнучкий спосіб.

Час обчислення в значній мірі піддається впливу технічних характеристик комп'ютера. Переклад тексту був виконаний на двох різних комп'ютерах. Перший з них мав відеокарту NVIDIA Tesla T4, яка доступна для безкоштовного використання в середовищі Google Colab. Другий комп'ютер не мав відеокарти, і всі обчислення були виконані на процесорі AMD Ryzen 5 4600H.

У таблиці нижче наведено результати часу, необхідного для перекладу відео тривалістю 1 хвилина 54 секунди.

Таблиця 4.1 – Час виконання етапів перекладу

Дія	**GPU (T4), с**	**CPU (4600H), с**
Завантаження, stt	87,77	308,13
Розподілення аудіо	148,81	55,44
Синтез мовлення	13,23	68,35
Зведення	2,99	3,12
Всього	(4:12) 252,83	(7:15) 435,07
Час на переклад відносно оригіналу	2,21	3,81

4.3 Керівництво користувача

Керівництво користувача представляє інформацію про функції та можливості вебзастосунку, розробленого для перекладу відео. Цей вебзастосунок розроблено з метою надати користувачам зручний і потужний інструмент для перекладу відео у форматах mkv або mp4, а також для перекладу відеоконтенту з англійської мови на українську.

Далі будуть інструкції щодо використання програми. Вам надано зручний та інтуїтивно зрозумілий інтерфейс для впевненого використання програми, щоб можна було легко отримати бажаний результат.

У цьому керівництві буде надано інформацію про можливості налаштування результату, включаючи вибір субтитрів оригіналу чи перекладу. Це дозволяє вам забезпечити більшу точність перекладу та налаштувати відображення субтитрів за власним бажанням.

Встановлення.

Початок роботи із системою передбачає відкриття публічного посилання, доступного для користувачів. Для досягнення цього, програмний засіб має бути активований на сервері. Загальновживаний формат стандартного публічного посилання має вигляд «[випадковий код].gradio.live». Однак, посилання також може мати будь-яку іншу форму, залежно від конкретного домену, який використовується.

Основні функції та операції.

При відкритті публічного посилання ви одразу потрапляєте в основне меню програми (див. рис. 4.7).

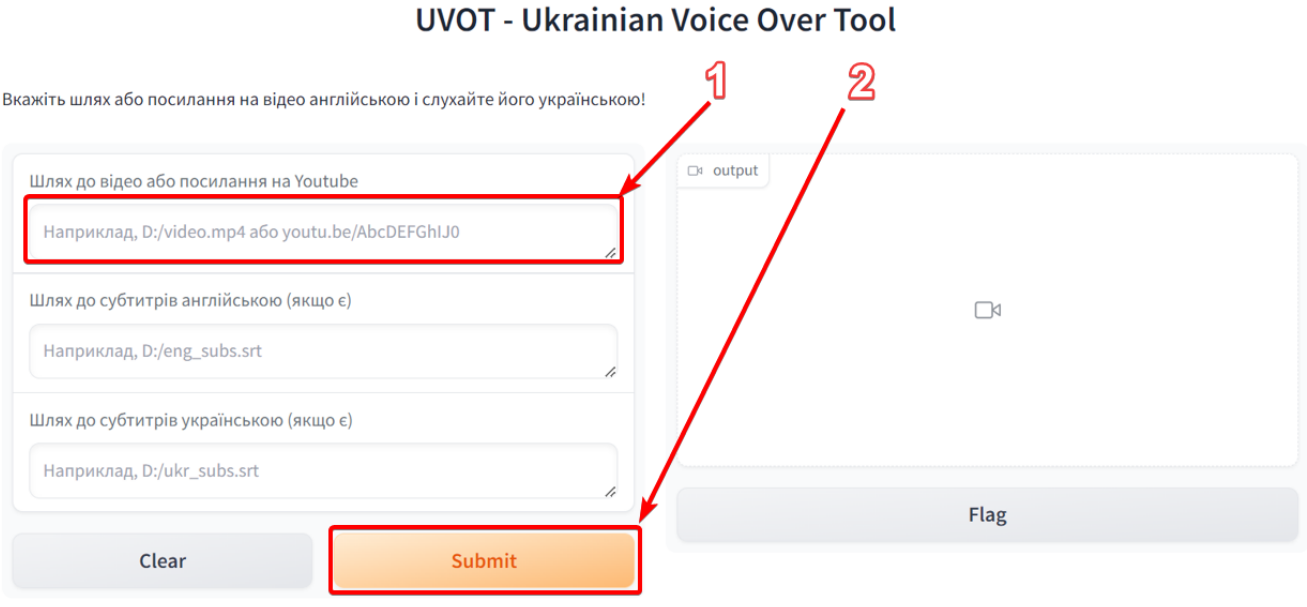

Рисунок 4.7 – Перші кроки у застосунку

Для початку роботи у вікно «Шлях до відео або посилання на YouTube» (рис. 4.7, позначка 1) слід ввести URL-адресу з відеохостингу YouTube. Якщо сервер це ваш комп'ютер, то можна ввести шлях до локального файлу. Коли це зроблено, для початку роботи достатньо натиснути «Submit» (рис. 4.7, позначка 2). Після цих дій програма почне перекладати відео. Час виконання зображено у правому верхньому куті (рис. 4.8, позначка 3), також видно індикатор виконання (рис. 4.8, позначка 4).

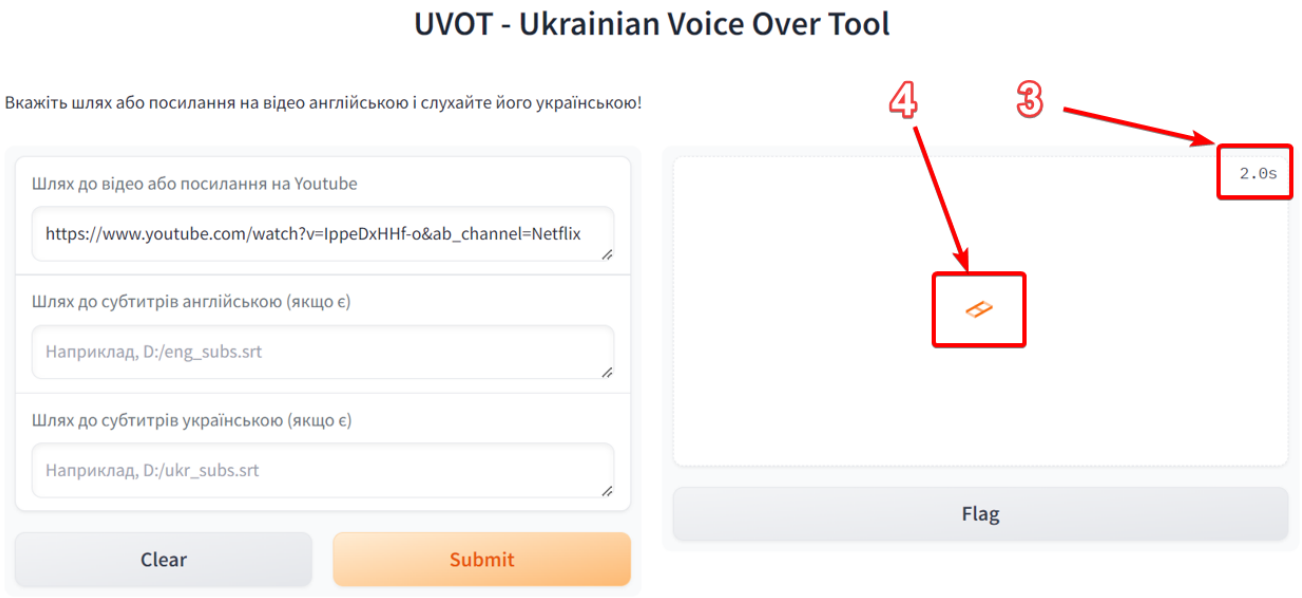

Рисунок 4.8 – Індикатори завантаження

Через певний час у вікні «Output» з'явиться перекладене відео (рис. 4.9, позначка 5). Це відео можна одразу переглянути. Для цього доступна панель керування переглядом відео (рис. 4.9, позначка 6), де можна поставити відео на паузу або продовжити перегляд, видно поточний час і тривалість відео, шкала прогресу перегляду відео та можливість увімкнути відео на весь екран.

Також, тут доступна можливість завантажити отримане відео на свій комп'ютер (рис. 4.9, позначка 7) для подальшого перегляду або використанню у власних цілях.

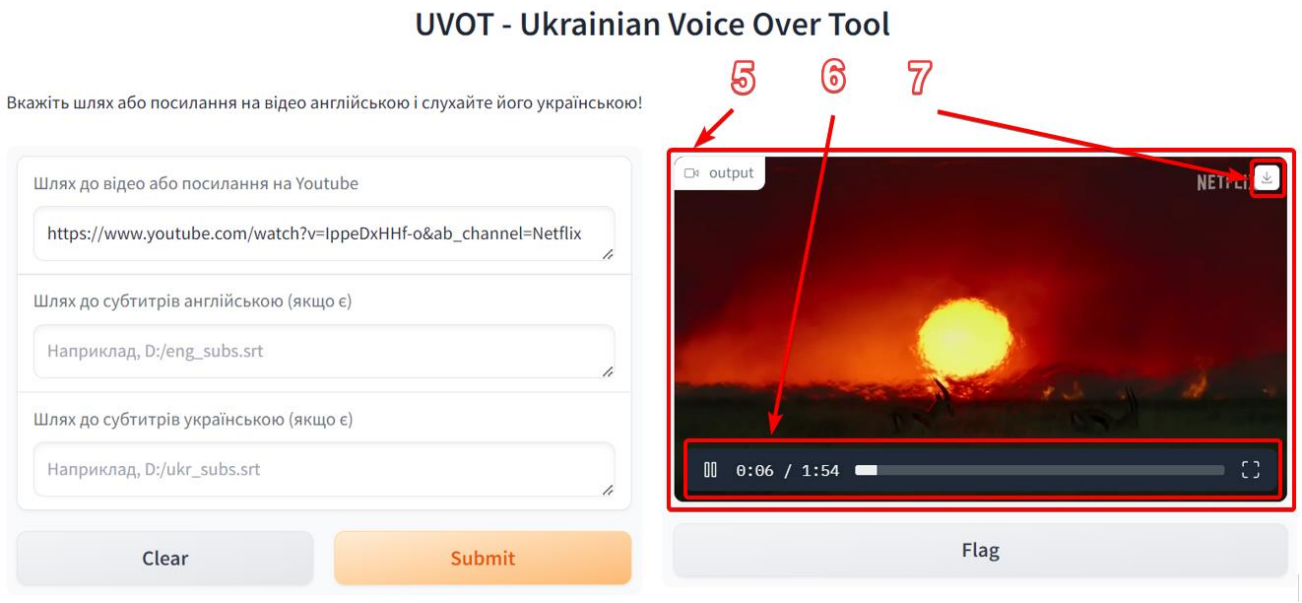

Рисунок 4.9 – Результат роботи застосунку

Якщо під час роботи застосунку щось піде не так, з’явиться повідомлення помилки (рис. 4.10, позначка 8).

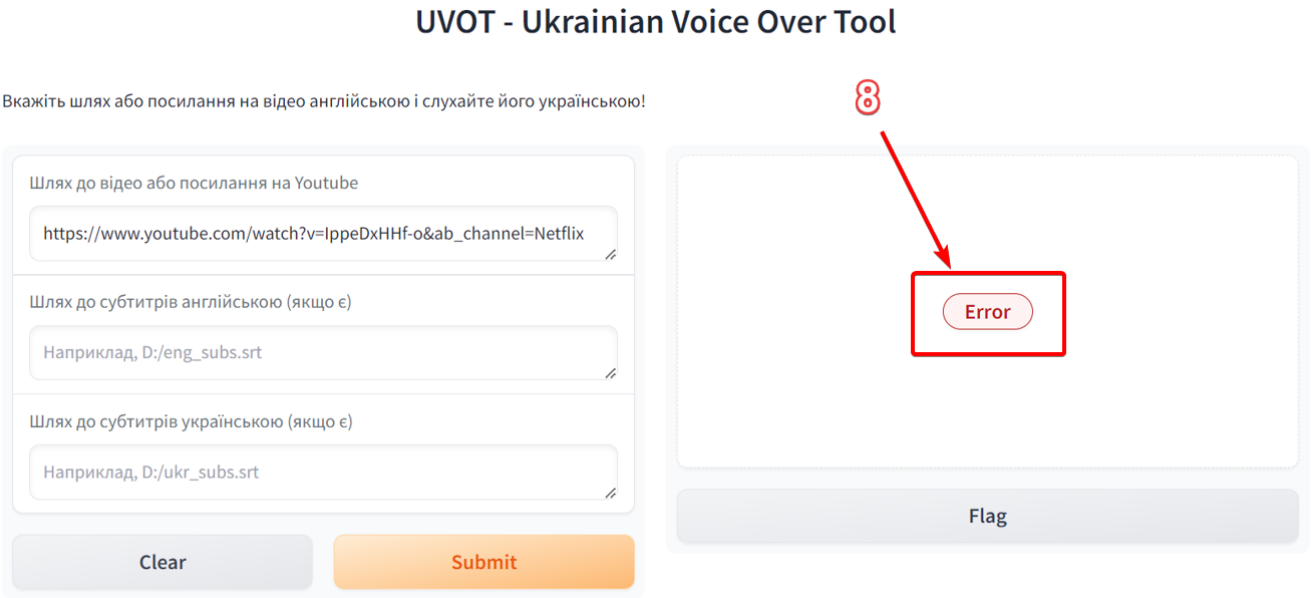

Рисунок 4.10 – Повідомлення про помилку

Розширені функції та налаштування.

Окрім основного функціоналу доступний і додатковий. Для більш чіткого результату можна вказати субтитри оригіналу (рис. 4.11, позначка 9). Вони матимуть найточніші репліки та часові мітки. Це зробить автоматичний переклад чіткішим та синтезований голос буде попадати у точні часові мітки.

Також, можна вказати вже перекладені субтитри (рис. 4.11, позначка 10). Їх можна або знайти у вільному доступі, або зробити самостійно. Це зробить переклад максимально точним, бо його переклад буде створений вручну, і часові мітки теж будуть розміщені максимально чітко.

Для повторної роботи, або для швидкого очищення введених даних можна натиснути кнопку «Clear» (рис. 4.11, позначка 11), яка очистить текстові поля.

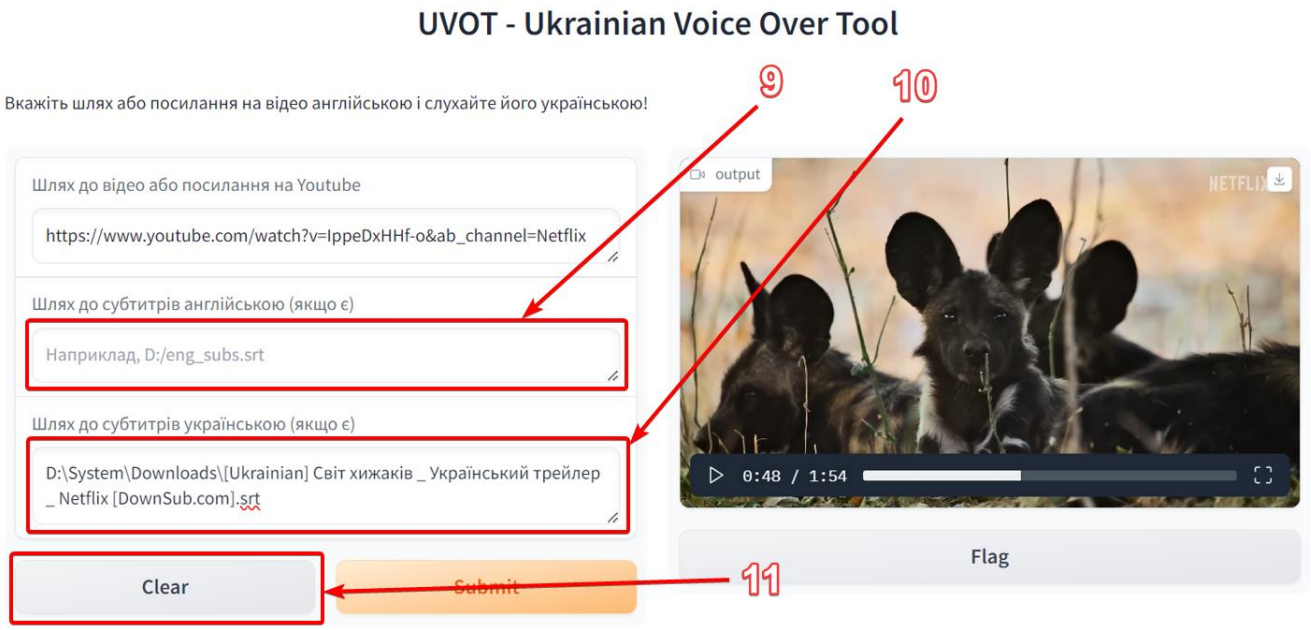

Рисунок 4.11 – Додатковий функціонал

Підказки та поради:

– за можливості шукайте відео із субтитрами, або надайте їх окремо. Це значною мірою підвисить точність перекладу;

– якщо у відео є вбудовані субтитри, не потрібно вказувати їх окремо – застосунок сам все знайде;

– якщо відео в YouTube матиме автоматичні субтитри, вони будуть проігноровані. Застосунок сам робить більш точні автоматичні субтитри;

– використовуйте більш потужний сервер для швидшого перекладу відео.

Сподіваюсь, що це керівництво надало вам необхідну інформацію для успішного використання вебзастосунку для перекладу відео. Бажаю вам приємного користування вебзастосунком та чудових перекладених відео!

Висновки до розділу 4

В четвертому розділі було розглянуто реалізацію програмного забезпечення для системи перекладу відео. Зокрема, було детально описано кожний скрипт та його функції, які виконують різноманітні завдання в процесі обробки відеоданих. Починаючи від розділення відео, аудіо та субтитрів у вхідному файлі, завантаження відео з YouTube, транскрибування аудіо, нормалізації субтитрів, до перекладу тексту та синтезування аудіо.

Кожен скрипт використовує різні бібліотеки та модулі для виконання своїх завдань. Наприклад, для розділення аудіо використовуються бібліотеки pydub, ffmpeg, audio_separator, а для синтезу аудіо з тексту використовується бібліотека ukrainian_tts.tts. Кожен етап обробки відео має свою власну функціональність, але вони всі інтегруються разом у фінальній програмі для забезпечення комплексного перекладу відео з англійської на українську мову.

Після детального розгляду програмного забезпечення було проведено тестування та аналіз результатів. За результатами роботи було розроблено вебзастосунок, який успішно виконує переклад відео у форматах mkv або mp4, а також будь-якого відео з YouTube.

Час виконання програми залежить від технічних характеристик комп'ютера, але у середньому для відео тривалістю 1 хвилину 54 секунди час виконання на GPU

складає 252,83 секунди, на CPU - 435,07 секунди, що відповідно довше у 2,21 та 3,81 рази відносно оригінального часу відео.

Керівництво користувача надає інформацію про роботу вебзастосунку та розглядає основні функції та можливості користувача. Після вивчення цього керівництва користувач повинен бути готовий до ефективного використання програми для перекладу відео.

Отже, в результаті даного дослідження було успішно розроблено програмне забезпечення, яке забезпечує зручний та потужний інструмент для перекладу відео з англійської на українську мову. Проєкт відкриває шлях для подальших досліджень у напрямку оптимізації алгоритмів, розширення функціональності, підтримки більшої кількості мов та розвитку більш деталізованого інтерфейсу для користувача.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи магістра було виконано мету та реалізовано інтелектуальну систему перекладу відео з англійської на українську мову. Завдання, поставлені в розділі вступу, були вирішені шляхом створення програмного забезпечення, яке здатне перекладати відео з англійської мови, завантажувати відео з різних джерел, забезпечувати якісний звук без очевидних дефектів та зрозумілу синтезовану аудіо доріжку українською мовою, передавати основне значення перекладу, а також має інтерфейс користувача для завантаження та отримання перекладеного відео.

У розділі першому було проведено аналіз існуючих технологій автоматичного перекладу відео та їх застосування. Було відзначено, що доступ до інформації для української аудиторії обмежений через нерівномірний розподіл контенту між українською та англійською мовами. Автоматичний переклад та озвучування відео з англійської на українську може значно розширити доступ до інформації та сприяти глибшому розумінню культурних особливостей союзників України на Заході. Також було відзначено проблему низького рівня англійської мови багатьох українців та складнощі отримання інформації через це.

У другому розділі було розроблено концепцію майбутнього додатка та вирішено, які завдання потрібно вирішити. Потім було розглянуто завдання автоматичного розпізнавання мовлення (ASR), завдання розділення звуків у аудіо на окремі компоненти, завдання створення субтитрів та синтезу мовлення українською мовою. Також було вибрано технології для розробки програмного забезпечення, включаючи мову програмування Python, середовище розробки Visual Studio Code та систему контролю версій GitHub.

У третьому розділі було проаналізовано кожен етап створення інтелектуальної системи перекладу відео, включаючи дизайн і архітектуру програмного забезпечення. Перед розробкою програмного забезпечення був розглянутий процес проектування інтерфейсу. Наступною стадією було

обговорення архітектури програмного забезпечення за допомогою методології Unified Modeling Language (UML), що дозволило побудувати використання випадків та послідовність дій.

У четвертому розділі було розглянуто реалізацію програмного забезпечення для системи перекладу відео. Кожний скрипт та його функції були детально описані, а також були проведені тести та аналіз результатів. На основі цього було розроблено вебзастосунок, який успішно виконує переклад відео у форматах mkv або mp4, а також з YouTube. Проаналізовано, що час виконання програми залежить від технічних характеристик комп'ютера, але у середньому для відео тривалістю 114 секунд час перекладу на GPU складає 252,83 секунди, на CPU - 435,07 секунди, що відповідно довше у 2,21 та 3,81 рази відносно оригінального часу відео.

Загальні результати вказують на успішне вирішення завдань, поставлених у вступі. Розроблене програмне забезпечення відповідає вимогам, забезпечуючи користувачам зручний та ефективний інструмент для перекладу відео з англійської на українську мову. Рекомендації щодо впровадження результатів полягають у подальшому розміщені на більш потужних серверах, розширенні можливостей та підтримці користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Usage Statistics and Market Share of Content Languages for Websites. W3Techs : вебсайт. URL: https://w3techs.com/technologies/overview/content_language (дата звернення: 08.10.2023).
2. Yang B. 6 Common Features Of Top 250 Channels On YouTube. Twinword : вебсайт. URL: https://www.twinword.com/blog/features-of-top-250-youtube-channels/ (дата звернення: 10.10.2023).
3. Культурні практики населення України: поведінка та ставлення : Аналітичний звіт : Київський міжнародний інститут соціології : Київ, 2023, 29 с.
4. Оцінка рівня володіння іноземними мовами дорослого населення України : Звіт кількісного соціологічного дослідження : Київський міжнародний інститут соціології : Київ, 2023, 84 с.
5. The unifying role of culture and cultural heritage in European integration. European movement. : вебсайт URL: https://europeanmovement.eu/policy/policy-position-on-culture-and-cultural-heritage/ (дата звернення: 23.12.2023).
6. Text to speech & ai voice generator - Elevenlabs. Text to speech & ai voice generator - Elevenlabs : вебсайт. URL: https://elevenlabs.io/dubbing (дата звернення: 18.12.2023).
7. Staniszewski M. A Comprehensive guide to ai dubbing for film and tv. Elevenlabs blog. : вебсайт URL: https://elevenlabs.io/blog/a-comprehensive-guide-to-ai-dubbing-for-film-and-tv/#introduction-to-ai-dubbing (дата звернення: 23.12.2023).
8. Вільний відеоперекладач. Telegram : вебсайт. URL: https://t.me/free_video_translator_bot (дата звернення: 08.12.2023).
9. Introducing a foundational multimodal model for speech translation. AI at Meta. : вебсайт. URL: https://ai.meta.com/blog/seamless-m4t/ (дата звернення: 23.12.2023).
10. Foundational models for state-of-the-art speech and text translation. GitHub : вебсайт. URL: https://github.com/facebookresearch/seamless_communication (date of access: 16.12.2023).

11. Dietrich K., Peters J. Testing the correlation of word error rate and perplexity. *Speech Communication*. 2002, Vol. 38 №1–2 P. 19–28. DOI:10.1016/S0167-6393(01)00041-3. ISSN 0167-6393.

12. Open ASR Leaderboard - a Hugging Face Space by hf-audio. Hugging Face – The AI community building the future : вебсайт. URL: https://huggingface.co/spaces/hf-audio/open_asr_leaderboard (дата звернення: 10.02.2024).

13. Wang, Y., Acero, A., Chelba, C. Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy. *IEEE Workshop on Automatic Speech Recognition and Understanding*. 2003. CiteSeerX: 10.1.1.89.424.

14. Robust Speech Recognition via Large-Scale Weak Supervision / Alec Radford та ін. ; 2022, 28 с.

15. Uhlich Stefan. and others. The Sound Demixing Challenge 2023 - Cinematic Demixing Track. arXiv.org : вебсайт. URL: https://arxiv.org/abs/2308.06981 (дата звернення: 14.02.2024).

16. Fabbro Giorgio and others. The Sound Demixing Challenge 2023 - Music Demixing Track. arXiv.org : вебсайт. URL: https://arxiv.org/abs/2308.06979 (дата звернення: 15.02.2024).

17. Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation : вебсайт. URL: https://arxiv.org/pdf/1505.04597.pdf (дата звернення: 22.12.2023).

18. Music Source Separation in the Waveform Domain / A. Défossez, N. Usunier, L. Bottou, and F. Bach. arXiv.org : вебсайт. URL: https://arxiv.org/abs/1911.13254 (дата звернення: 06.02.2024).

19. Stoller D., Ewert S., Dixon S. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. arXiv.org : вебсайт. URL: https://arxiv.org/abs/1806.03185 (дата звернення: 04.02.2024).

20. Défossez A. Hybrid Spectrogram and Waveform Source Separation. arXiv.org: вебсайт. URL: https://arxiv.org/abs/2111.03600 (дата звернення: 20.01.2024).

21. Hennequin R., Khlif A., Voituret F., Moussallam M. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*. 2020. Vol. 5, №. 50, p. 2154, DOI: 10.21105/joss.02154.

22. Rouard S., Massa F., Défossez A. Hybrid Transformers for Music Source Separation. arXiv.org : вебсайт. URL: https://arxiv.org/abs/2211.08553 (дата звернення: 27.01.2024).

23. KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing / M. Kim та ін. arXiv.org : вебсайт. URL: https://arxiv.org/abs/2111.12203 (дата звернення: 11.01.2024).

24. Anjok07. Ultimate Vocal Remover GUI v5.6: GUI for a Vocal Remover that uses Deep Neural Networks. GitHub : вебсайт. URL: https://github.com/Anjok07/ultimatevocalremovergui (дата звернення: 01.02.2024).

25. Translations Made Simple: The Usefulness of Translation Apps – Ulatus. Ulatus Translation Blog : вебсайт. URL: https://www.ulatus.com/translation-blog/most-globally-used-translated-apps/ (дата звернення: 12.01.2024).

26. Caswell I. Google Translate learns 24 new languages. Google : вебсайт. URL: https://blog.google/products/translate/24-new-languages/ (дата звернення: 03.01.2024).

27. Turovsky B. Ten years of Google Translate. Google : вебсайт. URL: https://blog.google/products/translate/ten-years-of-google-translate/ (дата звернення: 20.01.2024).

28. Lock S. What is AI chatbot phenomenon ChatGPT and could it replace humans?. The Guardian : вебсайт. URL: https://www.theguardian.com/technology/2022/dec/05/what-is-ai-chatbot-phenomenon-chatgpt-and-could-it-replace-humans (дата звернення: 01.02.2024).

29. Rider E. How ChatGPT Will Dramatically Change the Influencer Space | Entrepreneur. Entrepreneur : вебсайт. URL: https://www.entrepreneur.com/science-technology/how-chatgpt-will-dramatically-change-the-influencer-space/448386 (дата звернення: 03.02.2024).

30. DeepL press information | setting records!. DeepL Translate: The world's most accurate translator : вебсайт. URL: https://www.deepl.com/press.html (дата звернення: 23.12.2023).

31. DeepL Pro | Translate Text, Word Docs & Other Docs Securely. DeepL Translate: The world's most accurate translator : вебсайт. URL: https://www.deepl.com/pro#developer (дата звернення: 22.12.2023).

32. Pricing | Cloud Translation | Google Cloud. Google Cloud : вебсайт. URL: https://cloud.google.com/translate/pricing#charged-characters (дата звернення: 20.12.2023).

33. Pricing. OpenAI : вебсайт. URL: https://openai.com/pricing (дата звернення: 22.12.2023).

34. Tokenizer. OpenAI : вебсайт. URL: https://platform.openai.com/tokenizer (дата звернення: 22.12.2023).

35. Allen, Jonathan, Hunnicutt M. Sharon, Klatt Dennis. From Text to Speech: The MITalk system: Book, Cambridge University Press, New York, 1987. 216 p.

36. Text to speech. Microsoft Azure : вебсайт. URL: https://azure.microsoft.com/en-us/products/ai-services/text-to-speech (дата звернення: 08.02.2024).

37. Text-to-Speech AI: Lifelike Speech Synthesis | Google Cloud : вебсайт. URL: https://cloud.google.com/text-to-speech/?hl=uk (дата звернення: 08.02.2024).

38. Paniv Y. Robinhad/ukrainian-tts: Ukrainian TTS (text-to-speech) using ESPNET. GitHub : вебсайт. URL: https://github.com/robinhad/ukrainian-tts (дата звернення: 23.12.2023).

39. Shinji Watanabe and others. ESPnet: End-to-End Speech Processing Toolkit. *Interspeech 2018*. 2018. p. 2207-2211, DOI: 10.21437/Interspeech.2018-1456.

40. Guido V. R., Python Reference Manual : book, release 2.4.4, Scotts Valley, 2006. 242 p.

41. Kuhlman D. A Python Book: Beginning Python, Advanced Python, and Python Exercises : book, Platypus Global Media, 2011, 200 p.

42. What is a Web Application?. StackPath : вебсайт. URL: https://www.stackpath.com/edge-academy/what-is-a-web-application/ (дата звернення: 15.12.2023).

43. Gradio. Gradio : вебсайт. URL: https://www.gradio.app/ (дата звернення: 03.02.2024).

44. Microsoft. Visual Studio Code - Code Editing. Redefined. Visual Studio Code: вебсайт. URL: https://code.visualstudio.com (дата звернення: 17.12.2023).

45. Build software better, together. GitHub : вебсайт. URL: https://github.com/about (дата звернення: 16.12.2023).

46. About Us. Figma : вебсайт. URL: https://www.figma.com/about/ (дата звернення: 30.12.2023).

47. Brown D. M. Communicating Design: Developing Web Site Documentation for Design and Planning : book, 2nd ed. Indianapolis, 2011. 299 p.

48. Norman, D. A. Emotion & Design: Attractive things work better. *Interactions Magazine*. 2002. Vol. 9, № 4. P. 36–42. DOI: 10.1145/543434.543435.

49. Booch G., Rumbaugh J., Jacobson I., Unified Modeling Language User Guide, The : book, 2 ed., Addison-Wesley. 2005. 496 p.

50. Dr. Ivar Jacobson; Ian Spence; Kurt Bittner. Use-Case 2.0 ebook : ebook, Ivar Jacobson International 2011. 55 p.

51. Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure. : book, 2011. 748 p.

ДОДАТОК А

Open ASR Leaderboard

Таблиця 1 – Open ASR Leaderboard станом на 27.01.2024

Модель	WER	RTF (1e-3)
nvidia/parakeet-rnnt-1,1b	7,04	14,4
nvidia/parakeet-ctc-1,1b	7,58	5
nvidia/parakeet-rnnt-0,6b	7,63	12,3
openai/whisper-large-v3	7,7	7,45
nvidia/parakeet-ctc-0,6b	7,99	2,9
nvidia/stt_en_fastconformer_transducer_xlarge	8,06	12,3
openai/whisper-large-v2	8,06	7,45
nvidia/stt_en_fastconformer_transducer_xxlarge	8,07	14,4
distil-whisper/distil-large-v2	8,31	2,08
nvidia/stt_en_fastconformer_ctc_xxlarge	8,34	5
nvidia/stt_en_conformer_ctc_large	8,39	7,5
openai/whisper-medium,en	8,5	4,2
nvidia/stt_en_fastconformer_ctc_xlarge	8,52	2,9
nvidia/stt_en_fastconformer_ctc_large	8,9	1,8
nvidia/stt_en_fastconformer_transducer_large	8,94	10,4
openai/whisper-large	9,2	7,45
nvidia/stt_en_conformer_transducer_large	9,27	21,8
distil-whisper/distil-medium,en	9,32	1,26
openai/whisper-small,en	9,34	1,78
nvidia/stt_en_conformer_transducer_small	10,81	17,7
openai/whisper-base,en	11,67	1
nvidia/stt_en_conformer_ctc_small	11,77	3,2
patrickvonplaten/wav2vec2-large-960h-lv60-self-4-gram	13,65	20,1
facebook/wav2vec2-large-960h-lv60-self	14,47	2,5
openai/whisper-tiny,en	14,96	2,74
patrickvonplaten/hubert-xlarge-ls960-ft-4-gram	15,11	24,5
speechbrain/asr-wav2vec2-librispeech	15,61	2,6
facebook/hubert-xlarge-ls960-ft	15,81	6,3
facebook/mms-1b-all	15,85	5,9

Закінчення таблиці 1

facebook/hubert-large-ls960-ft	15,93	2,6
facebook/wav2vec2-large-robust-ft-libri-960h	16,07	2,7
facebook/wav2vec2-conformer-rel-pos-large-960h-ft	17	5,2
facebook/wav2vec2-conformer-rope-large-960h-ft	17,06	7,8
facebook/wav2vec2-large-960h	21,76	1,8
facebook/wav2vec2-base-960h	26,41	1,2
facebook/mms-1b-fl102	36,08	5,7

ДОДАТОК Б

Лістинг коду системи

Файл main.py

```
from uvot.split_audio import split_audio, convert_to_2_channels, split_6ch_audio
from uvot.normalize_subs import normalize_subs
from uvot.separate_video_audio_subs import separate_video_audio_subs
from uvot.tts import generate_audio_fragments, combine_audio_fragments,
speed_up_audio_in_folder
from uvot.combine_all import combine_all, combine_6ch_audio, combine_2ch_audio,
make_3ch
from uvot.adjust_audio_volume import adjust_audio_volume
from uvot.translate_text import translate_text
from uvot.delete_files import delete_temp_files
from uvot.find_files import find_file
from uvot.transcribe_audio import transcribe_audio
from uvot.youtube_download import download_youtube_video
from uvot.custom_subs import move_subtitles

from pydub import AudioSegment
import time
import gradio as gr
import os
import re

def main(input_video, custom_eng_subs=None, custom_ukr_subs=None):
    start_time_full = time.time()

    if os.path.isfile(input_video) and input_video.endswith(('.mp4', '.mkv')):
        separate_video_audio_subs(input_video)
    else:
        download_youtube_video(input_video)

    move_subtitles(custom_eng_subs, "splited_video/ENG_Subs.srt")
    move_subtitles(custom_ukr_subs, "splited_video/UKR_Subs.srt")

    if find_file("UKR_Subs") is None:
        if find_file("ENG_Subs") is None:
            transcribe_audio("splited_video/ENG_Audio.wav")

        normalize_subs(find_file("ENG_Subs"))
        translate_text("Temp_files/norm_subs.srt")
    else:
        normalize_subs(find_file("UKR_Subs"))
        move_subtitles("Temp_files/norm_subs.srt", "Temp_files/subs_uk.srt")

    audio = AudioSegment.from_file("splited_video/ENG_Audio.wav")
```

```
    num_channels = audio.channels
    start_time = time.time()

    if num_channels == 6:
        split_6ch_audio("splited_video/ENG_Audio.wav")
    elif num_channels == 2:
        split_audio("splited_video/ENG_Audio.wav")
    else:
        convert_to_2_channels("splited_video/ENG_Audio.wav",
"splited_video/ENG_Audio_stereo.wav")
        split_audio("splited_video/ENG_Audio_stereo.wav")

    generate_audio_fragments("Temp_files/subs_uk.srt")
    speed_up_audio_in_folder("audios", 15)
    combine_audio_fragments("Temp_files/subs_uk.srt")
    adjust_audio_volume("Temp_files/combined_audio.wav", "Temp_files/Vocal.wav")

    if num_channels == 6:
        make_3ch("Temp_files/vocals_adjusted.wav", "Temp_files/combined_audio.wav")
        combine_6ch_audio("splited_video/ENG_Audio.wav", "Temp_files/3_channel.wav",
"Temp_files/result_audio.wav")
    else:
        combine_2ch_audio()

    if os.path.isfile(input_video) and input_video.endswith(('.mp4', '.mkv')):
        combine_all(input_video, "Output/result.mkv")
    else:
        combine_all("Input/YT_Video.mp4", "Output/result.mkv")
    elapsed_time_full = time.time() - start_time_full
    print(f"Full time: {elapsed_time_full} seconds")
    delete_temp_files()
    return "Output/result.mkv"

# Create a Gradio interface
iface = gr.Interface(
    fn=main,
    inputs=[
        gr.Textbox(placeholder="Наприклад, D:/video.mp4 або youtu.be/AbcDEFGhIJ0",
label="Шлях до відео або посилання на Youtube"),
        gr.Textbox(placeholder="Наприклад, D:/eng_subs.srt", label="Шлях до субтитрів
англійською (якщо є)"),
        gr.Textbox(placeholder="Наприклад, D:/ukr_subs.srt", label="Шлях до субтитрів
українською (якщо є)"),
    ],
    outputs="video",
    live=False,
    title="UVOT - Ukrainian Voice Over Tool",
    description="Вкажіть шлях або посилання на відео англійською і слухайте його
українською!"
```

```
)

iface.launch(share=True)
```

Файл adjust_audio_volume.py

```
from pydub import AudioSegment

def adjust_audio_volume(audio_path, vocals_path):
    audio = AudioSegment.from_wav(audio_path)
    vocals_audio = AudioSegment.from_wav(vocals_path)

    # Зменшення гучності на 12 дБ в аудіофайлі vocals.wav
    reduction_in_db = -8
    vocals_audio = vocals_audio.apply_gain(reduction_in_db)

    # Поріг для визначення тиші (в мілісекундах)
    silence_threshold = -50  # Ви можете змінити це значення в залежності від
амплітуди вашого аудіофайлу

    # Мінімальна тривалість тиші, щоб враховувати її (в мілісекундах)
    min_silence_duration = 1000

    # Знаходження проміжків тиші з кроком 0,001 секунди
    silent_ranges = []
    start_silence = None
    step = 1  # Крок в мілісекундах (0,001 секунди)

    for i in range(0, len(audio), step):
        chunk = audio[i:i + step]

        if chunk.dBFS < silence_threshold:
            if start_silence is None:
                start_silence = i
        else:
            if start_silence is not None:
                end_silence = i
                duration = end_silence - start_silence
                if duration >= min_silence_duration:  # Якщо тиша триває 1 секунду
або довше
                    silent_ranges.append((start_silence, end_silence))
                start_silence = None

    # Збільшення гучності на 12 ДБ у проміжках тиші зі зміщенням на 200 мілісекунд
    increase_in_db = 12
    offset = 200  # зміщення в мілісекундах

    for start, end in silent_ranges:
```

```
        start_time = start / 1000 + offset / 1000
        end_time = end / 1000 - offset / 1000
        if start_time < 0:
            start_time = 0
        if end_time > len(vocals_audio) / 1000:
            end_time = len(vocals_audio) / 1000
        vocals_audio = vocals_audio.overlay(vocals_audio[start_time * 1000:end_time *
1000].apply_gain(increase_in_db), position=start_time * 1000)

    # Збереження зміненого аудіофайлу зі збільшеною гучністю у проміжках тиші
    output_path = "Temp_files/vocals_adjusted.wav"
    vocals_audio.export(output_path, format="wav")
```

Файл combine_all.py

```
import ffmpeg
import subprocess
from pydub import AudioSegment
import os


if not os.path.exists("Output"):
    os.makedirs("Output")

def get_first_audio_format(input_video):
    # Use ffprobe to get the format of the first audio stream in the input video
    probe = ffmpeg.probe(input_video, v='quiet', select_streams='a:0')
    return probe['streams'][0]['codec_name']

def combine_all(input_video, result_name):
    # Get the format of the first audio stream in the input video
    input_audio_format = get_first_audio_format(input_video)
    codec = input_audio_format
    if codec == "opus":
        codec = "libopus"
    # Convert the WAV audio to the same format as the first audio stream
    audio_conversion_command = [
        'ffmpeg', '-y', '-i', 'Temp_files/result_audio.wav',
        '-c:a', codec,
        'Temp_files/result_audio.' + input_audio_format
    ]
    subprocess.run(audio_conversion_command, check=True)

    # Combine video and new audio
    combine_command = [
        'ffmpeg', '-y', '-i', input_video,
        '-i', 'Temp_files/result_audio.' + input_audio_format, '-map', '0:v', '-map',
'1:a',
```

```
        '-c:v', 'copy', '-c:a', codec, result_name
    ]
    subprocess.run(combine_command, check=True)

def combine_2ch_audio():
    # Combine audio files
    command = [
        'ffmpeg', '-y', '-i', 'Temp_files/combined_audio.wav',
        '-i', 'Temp_files/Instrumental.wav',
        '-i', 'Temp_files/vocals_adjusted.wav',
        '-filter_complex', '[0:a][1:a][2:a]amix=inputs=3:duration=longest',
        'Temp_files/result_audio.wav'
    ]
    subprocess.run(command, check=True)

def make_3ch(vocals_adjusted, combined_audio, result='Temp_files/3_channel.wav'):
    # Combine audio files
    command = [
        'ffmpeg', '-y', '-i', vocals_adjusted,
        '-i', combined_audio,
        '-filter_complex', '[0:a][1:a]amix=inputs=2:duration=longest',
        result
    ]
    subprocess.run(command, check=True)

def combine_6ch_audio(ch6_audio, ch1_audio, output):
    audio6 = AudioSegment.from_file(ch6_audio, channels=6)
    audio1 = AudioSegment.from_file(ch1_audio, channels=1)
    channels = audio6.split_to_mono()
    target_length = len(channels[2])

    # Перевірка на однакову довжину
    if len(audio1) > target_length:
        audio1 = audio1[:target_length]
    elif len(audio1) < target_length:
        silence = AudioSegment.silent(duration=target_length - len(audio1))
        audio1 += silence

    channels[2] = audio1
    combined = AudioSegment.from_mono_audiosegments(*channels)
    combined.export(output, format='wav')
    print(f"Combined audio saved to {output}")
```

Файл custom_subs.py

```
import os
import pysubs2
```

```
def move_subtitles(input_subs, output_subs):
    try:
        if not os.path.isfile(input_subs):
            print(f"Файл {input_subs} не існує.")
            return
        # Завантажуємо субтитри за допомогою pysubs2
        subs = pysubs2.load(input_subs)

        # Зберігаємо субтитри у вказаний файл output_subs
        subs.save(output_subs)

        print(f"Субтитри збережено у файлі {output_subs}.")
    except Exception as e:
        print(f"Помилка при обробці субтитрів: {e}")
```

Файл delete_files.py

```
import os

def delete_temp_files():
    # List of folders to delete files from
    folders_to_clean = ["splited_video", "Temp_files", "audios"]

    for folder_name in folders_to_clean:

        # Get a list of all files in the folder
        files = os.listdir(folder_name)
        for file in files:
            file_path = os.path.join(folder_name, file)
            # Check if it's a file (not a directory) and delete it
            if os.path.isfile(file_path):
                os.remove(file_path)
```

Файл find_files.py

```
import os

def find_file(file_name):
    folder_path = "splited_video"  # Встановіть шлях до вашої папки
    file_name = file_name  # Назва файлу без розширення

    for root, dirs, files in os.walk(folder_path):
        for file in files:
            if file.startswith(file_name) and os.path.isfile(os.path.join(root,
file)):
                eng_subs = os.path.join(root, file)
                return eng_subs  # Повертаємо адресу файлу, якщо знайдено
```

```
    return None  # Повертаємо None, якщо файл не було знайдено
```

Файл normalize_subs.py

```
import json
import pysubs2
import re
import os

if not os.path.exists("Temp_files"):
    os.makedirs("Temp_files")

def normalize_subs(subs_file):
    subs = pysubs2.load(subs_file)
    for line in subs:
        line.text = re.sub(r'\\N', ' ', line.text)
        line.text = re.sub(r'{.*?}', '', line.text)
        line.text = re.sub(r'\[.*?\]', '', line.text)
        line.text = line.text.strip()

    # Filter out lines that don't have the "Default" style
    subs.events = [line for line in subs.events if line.style == 'Default']
    subs.remove_miscellaneous_events()
    subs.save("Temp_files/norm_subs.srt")

def json_to_srt(file_path):
    # Load JSON data
    with open(file_path, 'r') as file:
        data = json.load(file)

    # Create a new Subs2 object
    subs = pysubs2.SSAFile()

    # Loop through each chunk in the JSON data
    for chunk in data['chunks']:
        start, end = chunk['timestamp']
        text = chunk['text']

        # Convert the timestamps to milliseconds
        start_ms = int(start * 1000)
        end_ms = int(end * 1000)

        # Create a new subtitle event
        event = pysubs2.SSAEvent(start=start_ms, end=end_ms, text=text)
        subs.append(event)

    # Save the subtitles as an SRT file
    subs.save("Input/output.srt", encoding="utf-8")
```

```
def is_complete_sentence(text):
    return re.search(r"(\.\.\.|[.!?])", text.strip()) is not None

def find_sentence_end(text):
    matches = list(re.finditer(r"(\.\.\.|[.!?])", text))
    if matches:
        return matches[-1].end()
    return -1

def is_time_difference_large(sub1, sub2, threshold=1000):
    return sub2.start - sub1.end > threshold

def split_sentences(text, start_time, end_time):
    sentence_end_positions = [m.end() for m in re.finditer(r"(\.\.\.|[.!?])", text)]
    sentences = []
    prev_end = 0
    for end_pos in sentence_end_positions:
        sentences.append((text[prev_end:end_pos].strip(), start_time, end_time))
        prev_end = end_pos
    return sentences

def process_subtitles(subs):

    for i in range(len(subs) - 1):
        if is_time_difference_large(subs[i], subs[i + 1]):
            continue

        current_text = subs[i].text.strip()
        next_text = subs[i + 1].text.strip()

        if is_complete_sentence(current_text):
            sentence_end_idx = find_sentence_end(current_text)
            remaining_text = current_text[sentence_end_idx:].strip()

            if remaining_text:
                proportion_remaining = len(remaining_text) / len(current_text)
                time_adjustment = int(proportion_remaining * (subs[i].end -
subs[i].start))
                subs[i].end -= time_adjustment
                subs[i + 1].start -= time_adjustment

            subs[i + 1].text = remaining_text + " " + next_text if remaining_text
else next_text
            subs[i].text = current_text[:sentence_end_idx]
        else:
            if is_complete_sentence(next_text):
                sentence_end_idx = find_sentence_end(next_text)
                subs[i].text += " " + next_text[:sentence_end_idx].strip()
                subs[i + 1].text = next_text[sentence_end_idx:].strip()
```

```
                proportion_added = sentence_end_idx / len(next_text)
                time_adjustment = int(proportion_added * (subs[i + 1].end - subs[i +
1].start))
                subs[i].end += time_adjustment
                subs[i + 1].start += time_adjustment

    # Handle multiple sentences in the last line
    last_line_text = subs[-1].text.strip()
    if is_complete_sentence(last_line_text):
        new_lines = split_sentences(last_line_text, subs[-1].start, subs[-1].end)
        subs.pop()  # Remove the original last line
        for sentence, start_time, end_time in new_lines:
            new_line = pysubs2.SSAEvent(start=start_time, end=end_time,
text=sentence)
            subs.append(new_line)

    # Remove any empty lines and create a new SSAFile object
    filtered_subs = pysubs2.SSAFile()
    for sub in subs:
        if sub.text.strip():
            filtered_subs.append(sub)

    return filtered_subs
```

Файл separate_video_audio_subs.py

```
import ffmpeg

def separate_video_audio_subs(input_file):
    # Analyze the input file
    try:
        probe = ffmpeg.probe(input_file)
    except ffmpeg.Error as e:
        print('Error:', e.stderr)
        return

    audio_streams = [stream for stream in probe['streams'] if stream['codec_type'] ==
'audio']

    largest_eng_sub = None
    largest_ukr_sub = None

    for stream in probe['streams']:
        if len(audio_streams) == 1:
            # If there is only one audio stream, save it as ENG_Audio.wav
            ffmpeg.input(input_file).output('splited_video/ENG_Audio.wav').run(overwr
ite_output=True)
```

```
        else:
            # Check if the stream is audio and in English
            if stream['codec_type'] == 'audio' and stream['tags'].get('language') ==
'eng':
                ffmpeg.input(input_file).output('splited_video/ENG_Audio.wav',
map=f"0:{stream['index']}").run(overwrite_output=True)

            # Check if the stream is audio and in Ukrainian
            elif stream['codec_type'] == 'audio' and stream['tags'].get('language')
== 'ukr':
                ffmpeg.input(input_file).output('splited_video/UKR_Audio.wav',
map=f"0:{stream['index']}").run(overwrite_output=True)

        # Check if the stream is a subtitle and in English or Ukrainian
        if stream['codec_type'] == 'subtitle':
            lang = stream['tags'].get('language')
            if lang in ['eng', 'ukr']:
                # Determine subtitle format
                sub_format = stream.get('codec_name', 'srt')
                sub_extension = {
                    'subrip': 'srt',
                    'ass': 'ass',
                    # Add more mappings as needed
                }.get(sub_format, 'ass')

                output_filename =
f"splited_video/{lang.upper()}_Subs.{sub_extension}"

                # Check if it's the largest subtitle so far
                if lang == 'eng':
                    if largest_eng_sub is None or stream.get('NUMBER_OF_FRAMES', 0) >
largest_eng_sub.get('NUMBER_OF_FRAMES', 0):
                        largest_eng_sub = stream
                elif lang == 'ukr':
                    if largest_ukr_sub is None or stream.get('NUMBER_OF_FRAMES', 0) >
largest_ukr_sub.get('NUMBER_OF_FRAMES', 0):
                        largest_ukr_sub = stream

    # After processing all streams, save the largest English subtitle file
    if largest_eng_sub:
        output_filename = f"splited_video/ENG_Subs.{sub_extension}"
        ffmpeg.input(input_file).output(output_filename,
map=f"0:{largest_eng_sub['index']}").run(overwrite_output=True)

    # After processing all streams, save the largest Ukrainian subtitle file
    if largest_ukr_sub:
        output_filename = f"splited_video/UKR_Subs.{sub_extension}"
        ffmpeg.input(input_file).output(output_filename,
map=f"0:{largest_ukr_sub['index']}").run(overwrite_output=True)
```

Файл split_audio.py

```
import os
from audio_separator.separator.separator import Separator
from pydub import AudioSegment
import ffmpeg
import time
def split_audio(file_name):
    separator = Separator(
        audio_file_path=file_name,
        primary_stem_path="Temp_files/Vocal.wav",
        secondary_stem_path="Temp_files/Instrumental.wav",
        output_format="WAV",
        model_name="UVR_MDXNET_9482"
    )
    output_files = separator.separate()
    print("Output files:", output_files)
def convert_to_2_channels(file_name, output_file_name):
    input_file = ffmpeg.input(file_name)
    output_file = ffmpeg.output(input_file, output_file_name, ac=2)
    ffmpeg.run(output_file, overwrite_output=True)
def split_6ch_audio(file_name):
    ffmpeg.input(file_name).output("Temp_files/Vocal.wav",
af=f'pan=1c|c0=c2').run(overwrite_output=True)


def split_models_test(file_name, model_names):
    execution_times = {}
    for model_name in model_names:
        start_time = time.time()
        separator = Separator(
            audio_file_path=file_name,
            primary_stem_path=f"Temp_files/{model_name}_Instrumental.wav",
            secondary_stem_path=f"Temp_files/{model_name}_Vocal.wav",
            output_format="WAV",
            model_name=model_name
        )
        output_files = separator.separate()
        end_time = time.time()
        execution_time = end_time - start_time
        execution_times[model_name] = execution_time

    # Print execution times after all work is done
    for model_name, exec_time in execution_times.items():
        print(f'For "{model_name}" time: {exec_time:.2f} sec')
```

Файл transcribe_audio.py

```
from faster_whisper import WhisperModel
import torch
import pysubs2

def transcribe_audio(audio_file):
    model_size = "Systran/faster-whisper-large-v2"
    model = WhisperModel(model_size)
    def check_device():
        if torch.cuda.is_available():
            model = WhisperModel(model_size, device="cuda", compute_type="float16")
        else:
            model = WhisperModel(model_size, device="cpu", compute_type="int8")

    segments, info = model.transcribe(audio_file, beam_size=5)

    results= []
    for s in segments:
        segment_dict = {'start':s.start,'end':s.end,'text':s.text}
        results.append(segment_dict)

    pysubs2.load_from_whisper(results).save('splited_video/ENG_Subs.srt')
```

Файл translate_text.py

```
import subprocess

def translate_text(subs):
    command = [
        'translatesubs', subs, 'Temp_files/subs_uk.srt', '--to_lang', 'uk', '--
translator', 'google_trans_new'
    ]
    subprocess.run(command, check=True)
# translate_text("Temp_files/norm_subs_wwww3.srt")
```

Файл tts.py

```
from ukrainian_tts.tts import TTS, Voices, Stress
from pydub import AudioSegment
from pydub import silence
import os
import re
import torch

def check_device():
    if torch.cuda.is_available():
```

```
        return "cuda"
    else:
        return "cpu"

# subtitles_file = "Temp_files/ukr_orig.srt" #@param {type:"string"}
tts = TTS(device=check_device()) # can try gpu, mps


def synthesize_audio(text, output_filename):
    with open(output_filename, mode="wb") as file:
        _, _ = tts.tts(text, Voices.Mykyta.value, Stress.Dictionary.value, file)

def read_subtitles_file(subtitles_file):
    with open(subtitles_file, 'r', encoding='utf-8') as file:
        lines = file.readlines()
    return lines

def split_subtitles_into_phrases(subtitles_lines):
    phrases = []
    current_phrase = []
    for line in subtitles_lines:
        if line.strip().isdigit():
            if current_phrase:
                phrases.append(current_phrase)
                current_phrase = []
        else:
            current_phrase.append(line.strip())
    if current_phrase:
        phrases.append(current_phrase)
    return phrases

def combine_audio_files(phrases):
    output_folder = "audios"
    combined_audio = AudioSegment.empty()

    for idx, phrase in enumerate(phrases, 1):
        phrase_start_time = extract_time(phrase[0].strip().split(" --> ")[0])
        audio_filename = os.path.join(output_folder, f"{idx}.wav")
        audio_segment = AudioSegment.from_file(audio_filename, format="wav")
        silence_duration = phrase_start_time - len(combined_audio)
        if silence_duration > 0:
            silence_segment = AudioSegment.silent(duration=silence_duration)
            combined_audio += silence_segment
        combined_audio += audio_segment

    return combined_audio

def extract_time(time_string):
```

```
    hours, minutes, seconds, milliseconds = map(int,
re.findall(r"(\d+):(\d+):(\d+)[,.](\d+)", time_string)[0])
    total_milliseconds = hours * 60 * 60 * 1000 + minutes * 60 * 1000 + seconds *
1000 + milliseconds
    return total_milliseconds

def generate_audio_fragments(subtitles_file):
    phrases = split_subtitles_into_phrases(read_subtitles_file(subtitles_file))

    output_folder = "audios"
    os.makedirs(output_folder, exist_ok=True)  # Create the "audios" folder if it
doesn't exist

    for idx, phrase in enumerate(phrases, 1):
        phrase_text = " ".join(phrase[1:])
        audio_filename = os.path.join(output_folder, f"{idx}.wav")
        synthesize_audio(phrase_text, audio_filename)

def combine_audio_fragments(subtitles_file):
    phrases = split_subtitles_into_phrases(read_subtitles_file(subtitles_file))

    combined_audio = combine_audio_files(phrases)
    combined_audio.export("Temp_files/combined_audio.wav", format="wav")

def speed_up_audio_in_folder(folder_path, speedup_percentage):
    # Check if the folder exists
    if not os.path.exists(folder_path):
        print(f"Folder '{folder_path}' does not exist.")
        return

    # Iterate through all files in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith(".mp3") or filename.endswith(".wav"):
            file_path = os.path.join(folder_path, filename)

            # Load the audio file
            audio = AudioSegment.from_file(file_path)

            # Calculate the new playback speed
            new_speed = 1.0 + (speedup_percentage / 100)

            # Speed up the audio
            sped_up_audio = audio.speedup(playback_speed=new_speed)

            # Save the sped-up audio to a new file
            output_filename = os.path.splitext(filename)[0] + f".wav"
            output_path = os.path.join(folder_path, output_filename)
            sped_up_audio.export(output_path, format="wav")
            print(f"Saved '{output_filename}'")
```

Файл youtube_download.py

```
import os
import subprocess
import pysubs2

def download_youtube_video(url):
    # Create directory for split video if it doesn't exist
    if not os.path.exists('splited_video'):
        os.makedirs('splited_video')

    # Download the video in the best quality
    subprocess.run(['yt-dlp', '-f', 'bestvideo+bestaudio', '--merge-output-format',
'mp4', '-o', 'Input/YT_Video.mp4', url])

    # Download the English audio in the best quality
    subprocess.run(['yt-dlp', '-f', 'bestaudio', '--extract-audio', '--audio-format',
'wav', '--audio-quality', '0', '-o', 'splited_video/ENG_Audio.wav', url])

    # Download English subtitles in SRT format if they are not automatically
generated
    subprocess.run(['yt-dlp', '--write-sub', '--sub-langs', 'en', '--skip-download',
'--sub-format', 'srt', '-o', 'splited_video/ENG_Subs.srt', url])

    # Download Ukrainian subtitles in SRT format if they are not automatically
generated
    subprocess.run(['yt-dlp', '--write-sub', '--sub-langs', 'uk', '--skip-download',
'--sub-format', 'srt', '-o', 'splited_video/UKR_Subs.srt', url])

    eng_subs_path = "splited_video/ENG_Subs.srt.en.vtt"
    ukr_subs_path = "splited_video/UKR_Subs.srt.en.vtt"

    # Convert English subtitles
    if os.path.exists(eng_subs_path):
        subs = pysubs2.load(eng_subs_path, encoding="utf-8")
        subs.save("splited_video/ENG_Subs.srt")

    # Convert Ukrainian subtitles
    if os.path.exists(ukr_subs_path):
        subs1 = pysubs2.load(ukr_subs_path, encoding="utf-8")
        subs1.save("splited_video/UKR_Subs.srt")
```