Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра обчислювальної техніки
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«КОМП'ЮТЕРНА СИСТЕМА АВТОМАТИЧНОГО ПЕРЕКЛАДУ ЗВУКОВОЇ ДОРІЖКИ У ВІДЕОФАЙЛАХ»

08-54.МКР.004.00.000 ПЗ

Виконала: студентка 2-го курсу, групи
1КІ-23м спеціальності 123 «Комп'ютерна інженерія »
(шифр і назва напряму підготовки, спеціальності)
Водолазська Д.В.
(прізвище та ініціали)

Керівник: к. т. н., доц. каф. ОТ
Крупельницький Л.В.
(прізвище та ініціали)
«_____»_____2024 р.

Опонент: к.т.н., доц. каф.
Войтко В.В.
(прізвище та ініціали)
«_____»_____2024 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
(прізвище та ініціали)
«_____»_____2024 р.

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 123 « Комп'ютерна інженерія »

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., професор Азаров О.Д.
"18" вересня 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТКИ

Водолазської Дар'ї Вікторівни
(прізвище, ім'я, по батькові)

1 Тема роботи: Комп'ютерна система автоматичного перекладу звукової доріжки у відеофайлах, керівник роботи к. т. н., доц. кафедри ОТ Крупельницький Л.В., затверджені наказом вищого навчального закладу від "17" вересня 2024 року № 310.

2 Строк подання студентом роботи: 16.12.2024

3 Вихідні дані до роботи: бібліотека Coqui TTS, Whisper, фраємворк FastAPI, об'єкт для перекладу (відеофайл іноземною мовою), операційна система Windows.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз предметної області, проєктування системи для автоматичного перекладу відеофайлів, програмна реалізація системи автоматичного перекладу звукової доріжки у відеофайлах, економічна частина, література.

5 Перелік ілюстративного матеріалу: структурна схема, блок-схема алгоритму роботи, лістинг модуля транскипції та діаризації, лістинг модуля синтезу мовлення, протокол перевірки кваліфікаційної роботи на наявність текстових запозичень.

6 Консультанти розділів роботи представлено в таблиці 1.

Таблиця 1 – Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завда прий
1–3	Крупельницький Л.В., к. т. н., доцент каф. ОТ		
4	Небава М. І., доцент, к.е.н., професор каф. ЕПВМ		

7 Дата видачі завдання: 18.09.2024

8 Календарний план роботи зазначений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Постановка задачі роботи	19.09.2024	викона
2	Аналіз предметної області	20.09.2024–25.09. 2024	викона
3	Проєктування системи для автоматичного перекладу відеофайлів	26.09.2024 – 05.10.2024	викона
4	Програмна реалізація системи автоматичного перекладу звукової доріжки у відеофайлах	06.10.2024 – 26.10.2024	викона
5	Розрахунок економічної частини	11.11.2024	викона
6	Оформлення матеріалів до захисту МКР	12.11.2024	викона
7	Перевірка якості виконання магістерської роботи та усунення недоліків	13.11.2024 – 14.11.2024	викона
8	Підписи супроводжувальних документів у нормоконтролера, керівника, опонента	15.11.2024 – 16.11.2024	викона
9	Перевірка на антиплагіат та ШІ	17.11.2024	викона
10	Попередній захист роботи	18.11.2024	викона

Студент _____ Водолазська Д.В.
(підпис)

Керівник роботи _____ Крупельницький Л.
(підпис)

АНОТАЦІЯ

УДК 004.9

Водолазська Д.В. Комп’ютерна система автоматичного перекладу звукової доріжки у відеофайлах. Магістерська кваліфікаційна робота зі спеціальності 123 – Комп’ютерна Інженерія,. Вінниця: ВНТУ, 2024. 107 с.

На укр. мові. Бібліогр.: назв 60; рис.: 17; табл.9.

Магістерська кваліфікаційна робота розроблену комп’ютерну систему для автоматичного перекладу відеофайлів. У роботі використано мікросервісну архітектуру, яка забезпечує обробку аудіо, транскрипцію, переклад, діаризацію, синтез мовлення та формування субтитрів. Програмна реалізація виконана за допомогою Python, FastAPI, Whisper, FFmpeg та Coqui TTS. Технологія підтримує багатомовний контент і різні формати відео. Система протестована на практичних прикладах, продемонструвала точність розпізнавання, якість перекладу та синхронність роботи. Результати роботи можуть бути використані у медіа, освіті та міжнародній комунікації. Робота містить 109 сторінок, 17 рисунків, 9 таблиць.

Ключові слова: автоматичний переклад, транскрипція, діаризація, синтез мовлення, субтитри.

ABSTRACT

Vodolazska D.V. Computer System for Automatic Translation of Soundtracks in Video Files. Master’s Qualification Thesis in Specialty 123 – Computer Engineering. Vinnytsia: VNTU, 2024. 107p.

In the Ukrainian language. Bibliogr .: titles 60; fig .17; table 9.

This master's qualification thesis is dedicated to the development of an information technology for automatic video file translation. The work proposes a microservice architecture that enables audio processing, transcription, translation, diarization, speech synthesis, and subtitle generation. The software implementation was carried out using Python, FastAPI, Whisper, FFmpeg, and Coqui TTS. The technology supports multilingual content and various video formats. The system was tested on practical examples, demonstrating recognition accuracy, translation quality, and synchronization. The results of this work can be applied in media, education, and international communication. The thesis contains 108 pages, 17 figures, and 9 tables.

Keywords: automatic translation, transcription, diarization, speech synthesis, subtitles.

ЗМІСТ

ВСТУП

У сучасному світі зростає **актуальність** технологічних рішень у сфері обміну інформацією між різними культурами та мовами. Зі зростанням кількості відеоконтенту в інтернеті та на платформах стрімінгу переклад аудіо та відеоматеріалів стає все більш важливим для забезпечення доступу до інформації для користувачів з різних країн.

Традиційні методи перекладу вимагають значних часових та людських ресурсів, що ускладнює процес перекладу великої кількості відеофайлів. Зокрема, переклад звукових доріжок у відео потребує не лише розуміння контексту, але й синхронізації перекладеного тексту або озвучення з візуальним рядом. У зв'язку з цим, автоматизація процесу перекладу за допомогою комп'ютерних систем є важливою задачею.

Комп’ютерні системи автоматичного перекладу здатні значно підвищити ефективність перекладу аудіо та відеоконтенту, зменшуючи часові витрати та підвищуючи точність. Сучасні технології штучного інтелекту, такі як розпізнавання мовлення (ASR) та синтез голосу (TTS), дозволяють автоматизувати процес перекладу звукових доріжок у відеофайлах з мінімальним втручанням людини.

Метою дослідження даної роботи є вдосконалення комп’ютерної системи для автоматичного перекладу звукової доріжки у відеофайлах. Така система дозволить автоматизувати процес перекладу, забезпечуючи зручний інтерфейс для завантаження відео та отримання перекладених версій звукових доріжок.

Для досягнення мети роботи потрібно виконати такі **задачі**:

— проаналізувати методи та технології автоматичного перекладу аудіо;

— дослідити існуючі аналоги систем автоматичного перекладу відео;

— обрати оптимальні засоби реалізації системи;

— розробити прототип комп’ютерної системи;

— протестувати систему на різних мовах і форматах відеофайлів.

Об'єкт дослідження – процес автоматичного перекладу звукової доріжки у відеофайлах, що включає використання методів автоматизованого аналізу та перетворення аудіо сигналу.

Предмет дослідження – технології та методи обробки аудіо та відео для автоматичного перекладу, алгоритми розпізнавання мови, системи машинного перекладу та інтерфейси для інтеграції з відео контентом.

У роботі використані такі **методи дослідження**: методи та моделі подання знань, методи математичного моделювання, методи тестування програмних засобів, методи об’єктно-орієнтованого програмування.

Новизна дослідження – вдосконалено адаптивний метод перекладу звукової доріжки у відеофайлах, що забезпечує збільшену якість та достовірність перекладу за рахунок інтеграції сучасних алгоритмів розпізнавання мови та машинного перекладу. Цей адаптивний метод дає можливість роботи з різними мовами та форматами відео.

Практичне значення – розроблена система застосовується для автоматизованого перекладу звукових доріжок в різноманітних сферах, таких як медіа, навчання та міжнародна комунікація. Це дозволяє полегшити доступ до багатомовної інформації та сприяє глобалізації контенту.

Запропонована система сприяє підвищенню ефективності процесу перекладу відеофайлів:

— розроблено алгоритм інтеграції системи з мультимедійними файлами, який дозволяє ефективно обробляти різні формати відео та аудіо для подальшого аналізу та перекладу;

— розроблено алгоритм роботи системи на основі трансформерів, що оптимізує процес перекладу та адаптації тексту до мовних особливостей цільової аудиторії;

— програмний комплекс для автоматизації перекладу відеофайлів.

Апробація результатів роботи була здійснена на міжнародних науково-практичних інтернет-конференціях «Молодь в науці: дослідження, проблеми, перспективи – 2024,2025».

Публікації за результатами досліджень опубліковано тези доповідей на науково-технічній конференції [1, 2]:

Водолазська, Д., Крупельницький Л.В;. Інтеграція систем на основі штучного інтелекту в розробку програмного забезпечення Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21685/0

Водолазська, Д., Крупельницький Л.,. Порівняння алгоритмів кластеризації для задач розпізнавання голосу Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22267.

.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ТЕХНОЛОГІЙ ПЕРЕКЛАДУ ЗВУКОВОЇ ДОРІЖКИ

1.1 Обгрунтування необхідності автоматизації перекладу

Еволюція перекладу мультимедійного контенту тісно пов’язана із розвитком цифрових технологій. З моменту появи перших ручних методів створення субтитрів і дубляжу до сучасних автоматизованих систем, ця сфера зазнала значного прогресу. У сучасному світі, що характеризується глобалізацією, переклад мультимедіа відіграє вирішальну роль у забезпеченні доступу до інформації для користувачів із різними мовними уподобаннями. Водночас стрімке зростання обсягів цифрового контенту створює нові виклики для традиційних методів перекладу, які стають дедалі менш ефективними.

Згідно зі звітом "Global Digital Content Creation Market", обсяг ринку створення цифрового контенту, включаючи текст, аудіо, відео та графіку, очікується на рівні $50,2 мільярда до 2028 року зі щорічним темпом зростання у 13,2% рисунок 1.1.

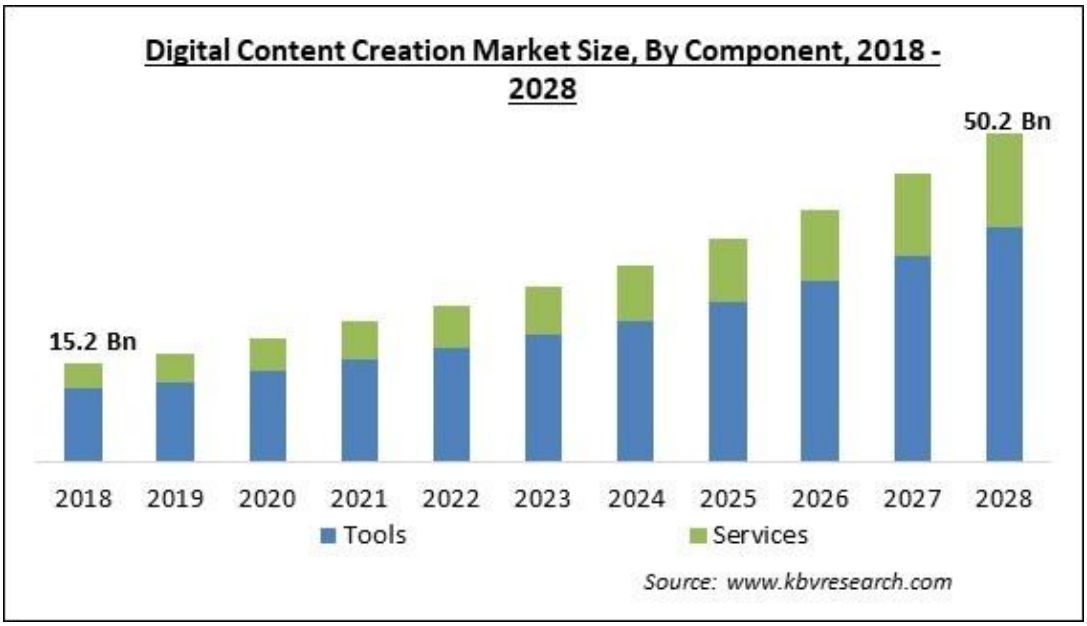

Рисунок 1.1 — Розмір ринку цифрового контенту.

Збільшення інтернет-трафіку, активне впровадження цифрових маркетингових стратегій та інтеграція штучного інтелекту сприяють цьому стрімкому росту. Наприклад, у 2021 році обсяг даних, збережених у хмарі, склав 50% від загальної кількості даних, що є суттєвим зростанням порівняно з 25% у 2015 році[3, 4].

Однак зростання обсягів контенту створює значні виклики для традиційних методів перекладу. Вони вимагають багато часу та ресурсів, що не дозволяє забезпечити переклад у масштабах, необхідних для задоволення сучасного попиту. Наприклад, ручне створення субтитрів є трудомістким процесом, який стає надзвичайно дорогим у разі обробки великого обсягу мультимедійних даних.

На додачу до цих викликів, пандемія COVID-19 стала каталізатором ще швидшого переходу до цифрових платформ. Багато компаній змушені були переглянути свої маркетингові стратегії та посилити свою присутність у цифровому середовищі, що збільшило попит на автоматизацію процесів, зокрема перекладу. Переваги автоматизованих систем перекладу, такі як масштабованість, швидкість і зниження витрат, зробили їх ключовим інструментом у подоланні сучасних викликів.

Диспропорція між обсягами контенту, створеного різними мовами, і кількістю носіїв цих мов стала однією з ключових проблем, що впливають на доступ до інформації. Англійська мова, наприклад, домінує в інтернет-просторі, представляючи понад 60% контенту у глобальній мережі. Водночас лише близько 25% населення світу володіє англійською мовою на достатньому рівні для сприйняття складної інформації. Для інших мов, таких як китайська, іспанська чи арабська, співвідношення між обсягами контенту та числом носіїв є ще більш несприятливим дивись рисунок 1.2.

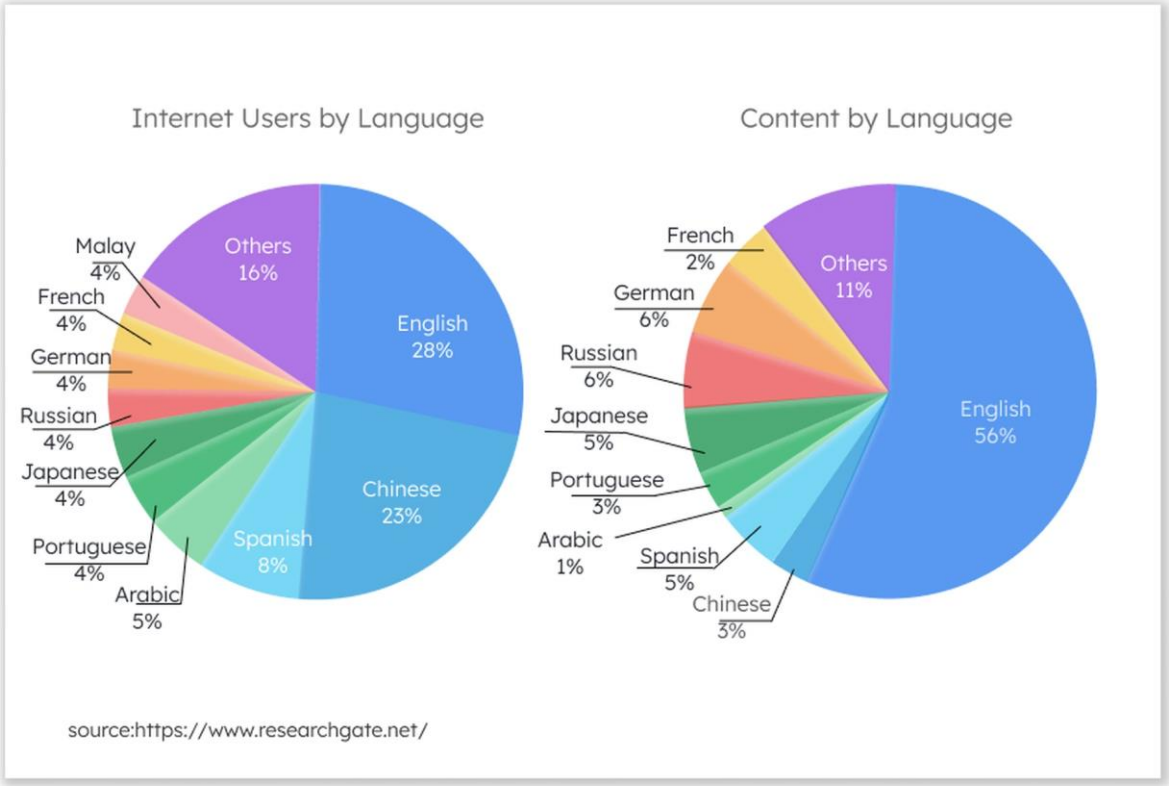

Рисунок 1.2 — Частка мовного контенту в інтернеті у порівнянні з чисельністю носіїв мов

Ця ситуація ускладнюється ще більше для малоресурсних мов, які не мають значних обсягів цифрового контенту. Для таких мов, як українська, хінді чи суахілі, наявність освітнього, культурного чи новинного контенту в інтернеті є вкрай обмеженою. Це створює бар'єри для доступу до інформації та можливостей для розвитку як окремих людей, так і цілих суспільств.

Приріст користувачів інтернету значною мірою пов’язаний із поширенням смартфонів, які зробили доступ до цифрових ресурсів дешевшим і простішим. У багатьох країнах, що розвиваються, смартфони стали основним засобом виходу в інтернет. Проте дисбаланс у доступності контенту на місцевих мовах залишається значною проблемою, оскільки більшість нових користувачів не володіють англійською мовою.

Таким чином, розвиток технологій автоматичного перекладу є не лише логічним етапом еволюції цієї галузі, а й необхідною умовою для підтримки глобального доступу до мультимедійного контенту в умовах зростання його обсягів [5, 6].

1.2 Аналіз існуючих проблем і перспектив розвитку автоматичного перекладу мультимедіа

Автоматичний переклад стикається із значними викликами, які впливають на його якість і точність. Проблеми, пов'язані з обмеженою підтримкою мов, діалектів, низькою якістю аудіозаписів і фоновими шумами, суттєво ускладнюють досягнення високої точності в обробці мовлення. Ці виклики є особливо критичними для малоресурсних мов, де нестача даних ускладнює створення якісних моделей. Однак сучасні підходи, такі як перенесення навчання (transfer learning), багатомовне навчання (multilingual learning) та використання автоматизованого розширення даних (data augmentation), відкривають нові можливості для вирішення цих проблем і забезпечують перспективи для розвитку технологій автоматичного перекладу мультимедіа.

Однією з основних проблем автоматичного перекладу мультимедіа є обмежена підтримка малоресурсних мов рисунок 1.3. Для цих мов часто відсутні достатні корпуси текстових і аудіоданих. Аннотовані дані, необхідні для навчання моделей машинного навчання, є дорогими і трудомісткими у створенні, оскільки вимагають ручного маркування тисяч прикладів. Однак навіть неаннотовані дані, які є основою для створення базових моделей, також є дефіцитними, що ускладнює роботу з такими мовами.

Перспективи вирішення цієї проблеми відкривають технології перенесення навчання. Вони дозволяють використовувати знання, отримані під час роботи з багаторесурсними мовами, для покращення роботи моделей на малоресурсних мовах.

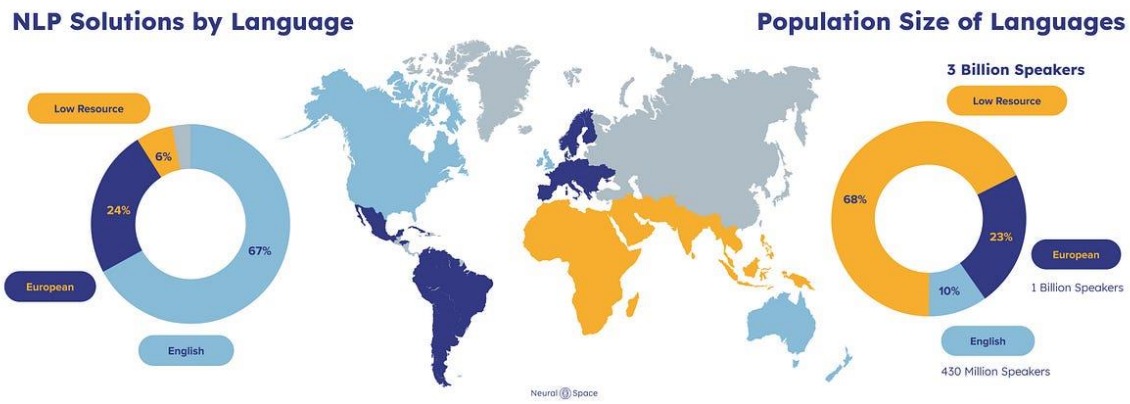

Рисунок 1.3 — Мапа мов із недостатньою підтримкою автоматизованих систем

Наприклад, моделі, попередньо навчені на англійській мові, можуть бути донавчені для роботи зі схожими мовами, такими як українська чи польська. Це значно знижує витрати на створення нових моделей і підвищує їх точність навіть за умов обмеженої кількості даних [10–14].

Особливо складною є підтримка діалектів, які можуть значно відрізнятися навіть у межах однієї мови. Україна, наприклад, має три основні діалекти: північний, південно-західний і південно-східний рисунок 1.4. Кожен з них має унікальні фонетичні та лексичні особливості, які ускладнюють розпізнавання мовлення стандартними моделями [15]. Ця проблема типова і для багатьох інших мов, наприклад, арабської, де більшість доступних даних стосується лише сучасної стандартної мови, яка значно відрізняється від розмовних діалектів [16].

Багатомовне навчання пропонує вирішення цієї проблеми. Цей підхід дозволяє одній моделі одночасно вивчати кілька мов і діалектів. Такі моделі здатні передавати знання між мовами, що суттєво покращує роботу з малоресурсними діалектами. Наприклад, моделі, навчені на даних англійської, можуть автоматично покращувати свою роботу з діалектами інших мов германської групи через спільні лінгвістичні особливості.

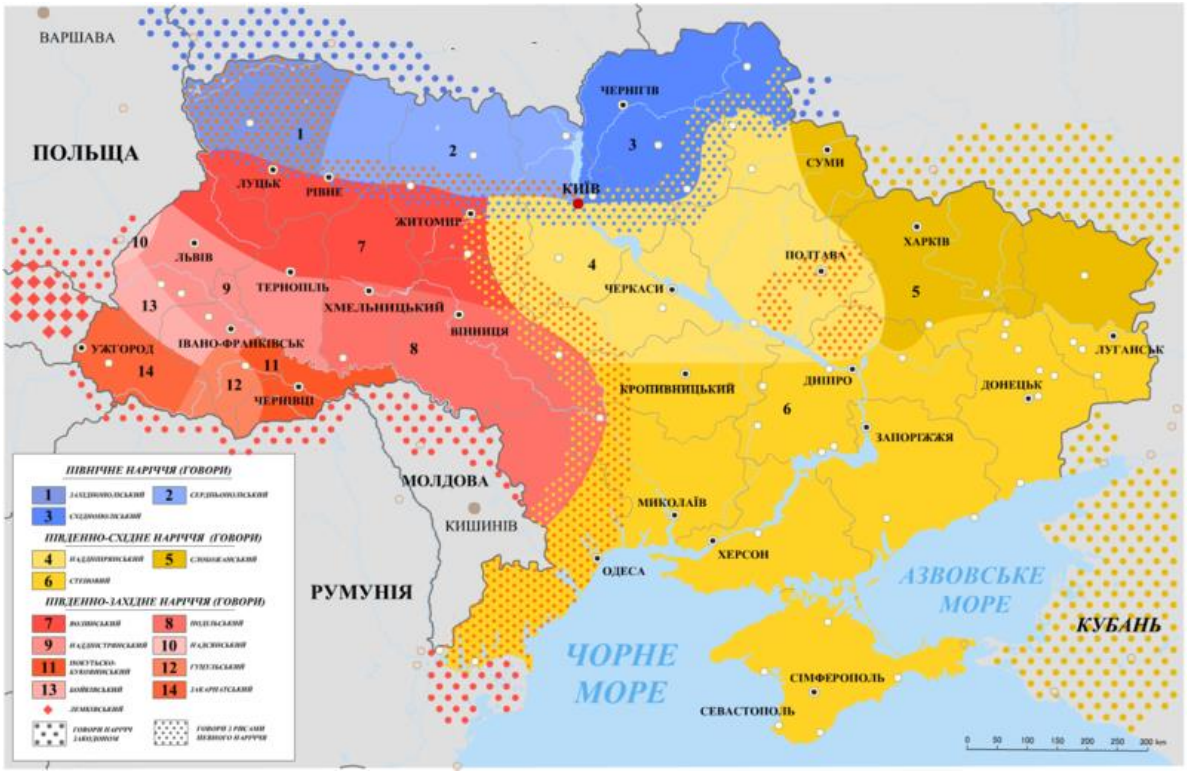

Рисунок 1.4 — Мапа діалектів України

Ще одним дієвим інструментом для роботи з малоресурсними мовами є автоматизоване розширення даних. Цей метод дозволяє створювати нові навчальні дані без їх ручного збирання. Наприклад, речення «Сьогодні гарна погода» можна переформулювати як «Сьогодні чудовий день». Використання таких технік значно збільшує обсяг даних, доступних для навчання моделі, і підвищує її точність. Інші способи включають зворотний переклад і використання синонімів.

Поєднання методів перенесення навчання, багатомовного підходу та автоматизованого розширення даних дає змогу ефективно вирішувати проблеми малоресурсних мов і діалектів. Це відкриває нові горизонти для розвитку автоматичного перекладу мультимедіа, роблячи його доступним для більшої кількості користувачів [17–19].

Фоновий шум є одною з перешкод для автоматичного розпізнавання мовлення та перекладу мультимедіа. У реальних умовах аудіозаписи часто

містять значний рівень шумів, таких як навколишні звуки, музика чи голоси інших людей. Це суттєво знижує точність роботи систем автоматичного перекладу, оскільки моделі розпізнавання мовлення сприймають шум як частину мовного сигналу, що призводить до помилкової інтерпретації.

Один із ключових показників, який використовується для оцінки впливу шуму на точність розпізнавання мовлення, є WER (Word Error Rate). Ця метрика показує відсоток помилок у розпізнаному тексті порівняно з реальним текстом. Високий рівень шуму значно збільшує WER, що можна побачити на рисунку 1.5, де залежність точності від рівня шуму представлена для аудіо лише зі звуком (AO) та аудіо з відео (AV)

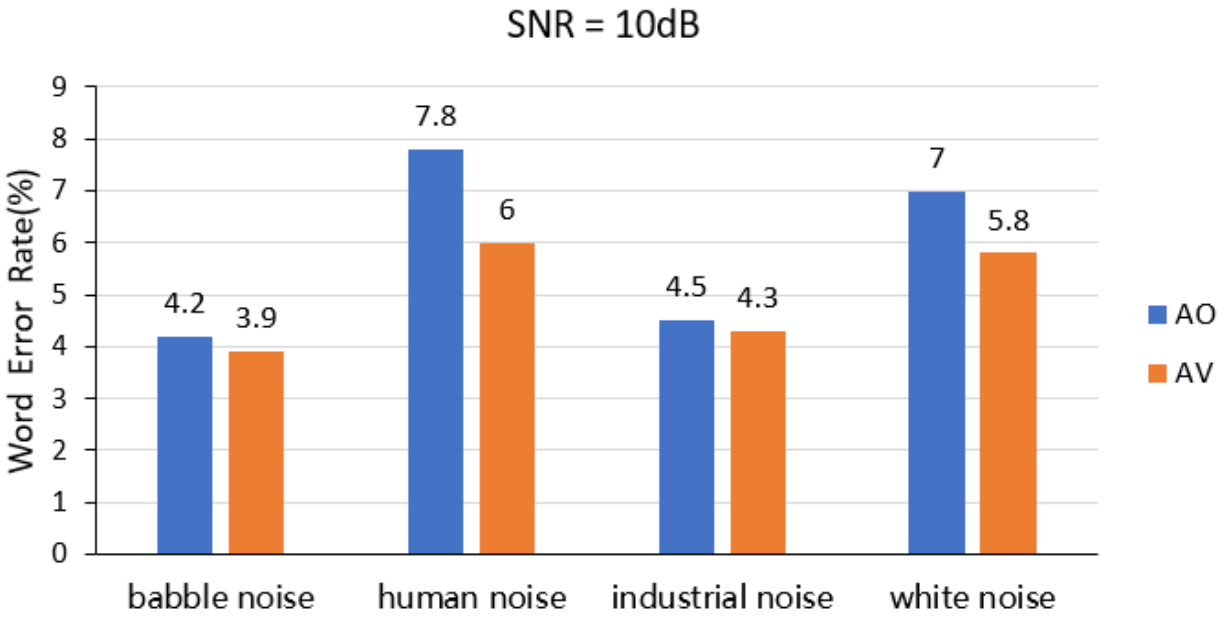

Рисунок 1.5 — Вплив шуму на точність розпізнавання

Для зниження впливу шуму сучасні системи використовують методи попередньої обробки аудіо, такі як фільтрація шумів і нормалізація сигналу. Фільтри низьких і високих частот дозволяють видаляти небажані звуки, які знаходяться поза діапазоном людського голосу. Крім того, сучасні алгоритми

машинного навчання, такі як глибокі нейронні мережі, здатні адаптуватися до шумових умов шляхом додаткового навчання на шумних даних.

Особливо ефективними є моделі, які інтегрують аудіовізуальну інформацію (AV). Такі системи використовують дані відео, наприклад, рух губ мовця, для покращення точності розпізнавання мовлення навіть за умов значного рівня шуму. Це забезпечує зниження впливу фонових шумів і покращує розпізнавання в реальних умовах.

Синхронізація тексту з відео є однією з найскладніших задач у створенні систем автоматичного перекладу мультимедіа. Основна мета цього процесу — забезпечити точну відповідність між моментами звучання аудіо та текстовими субтитрами, які відображаються на екрані. Ця відповідність є критично важливою для створення якісного користувацького досвіду, адже навіть найменші розбіжності у часових мітках можуть викликати дезорієнтацію глядача.

Сучасні технології, такі як механізми машинного навчання і алгоритми вирівнювання, допомагають досягти високої точності синхронізації, забезпечуючи плавний і природний показ тексту.

Крім того, синхронізація тексту є особливо важливою для контенту, що транслюється в реальному часі, наприклад, вебінарів чи стрімів, де затримка навіть у кілька секунд може суттєво вплинути на взаємодію з аудиторією. Таким чином, точна синхронізація тексту і відео залишається ключовим етапом у створенні ефективних систем автоматичного перекладу мультимедіа.

Основний виклик полягає у тому, що мовлення не є рівномірним процесом. Темп і тривалість вимови різних слів чи фраз можуть значно відрізнятися. Рисунок 1.6 ілюструє яка непостійна звукова хвиля аудіозапису. Додатково, паузи, зміни інтонації або акцент на певних словах ускладнюють точне визначення початку та кінця мовних одиниць. Для автоматичних систем це

означає, що лише розпізнати текст недостатньо — потрібно правильно розподілити його у часі.

Ще одним важливим аспектом є робота з багатопотоковим контентом, наприклад, відео із кількома мовцями. У таких випадках для точного відображення тексту необхідно враховувати, хто саме говорить у певний момент. Це додає складності процесу синхронізації, оскільки потребує інтеграції діаризації (визначення мовців) із часом відображення субтитрів.

Сучасні системи використовують кілька підходів для забезпечення точності субтитрів із часовими мітками. Одним із найефективніших є використання моделей, які виконують транскрипцію з одночасним визначенням часових меж для кожного сегменту мовлення. Whisper, наприклад, інтегрує часові мітки у процесі розпізнавання, що дозволяє уникнути додаткового етапу синхронізації.

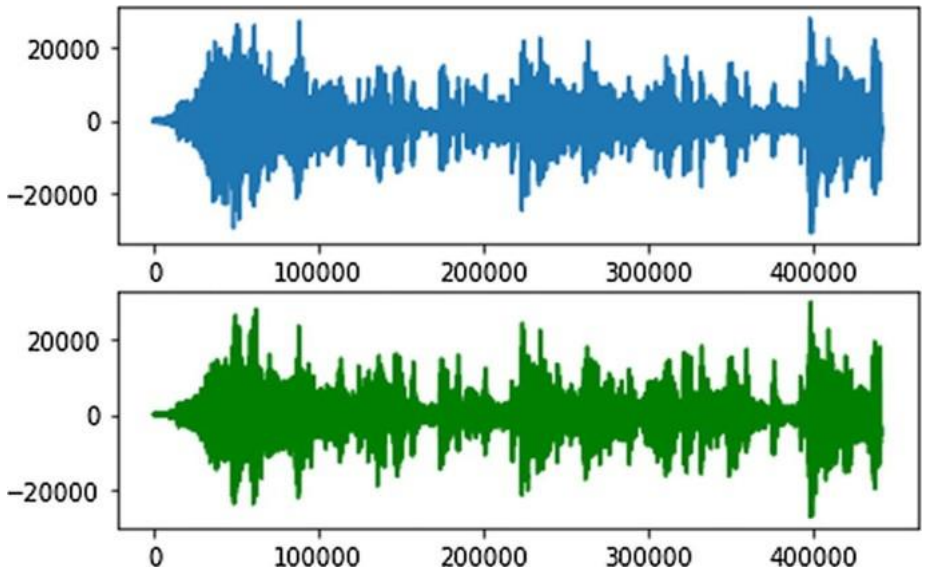

Рисунок 1.6 — Ілюстрація звукової хвилі в середовищі Pandas

Для підвищення точності також використовуються алгоритми вирівнювання тексту з аудіо. Ці алгоритми аналізують спектральні характеристики звуку та порівнюють їх із розпізнаним текстом, уточнюючи час

початку та завершення кожного слова чи фрази. WhisperX, наприклад, дозволяє інтегрувати таке вирівнювання у процес обробки, що особливо ефективно для субтитрування фільмів чи лекцій [20–24].

1.3 Огляд інструментів автоматичного перекладу аудіо та відео

Rask AI — це інноваційна платформа, яка дозволяє користувачам перекладати відеоконтент на різні мови за допомогою штучного інтелекту. Користувачі можуть завантажувати відео, обирати цільову мову, а система автоматично генерує перекладені субтитри та дубльовані звукові доріжки. Це значно спрощує процес перекладу порівняно з традиційними методами. Rask AI знаходить застосування в електронному навчанні, розвагах, маркетингу та корпоративних комунікаціях, допомагаючи подолати мовні бар'єри та охопити ширшу аудиторію. Вартість послуг залежить від обсягу використання та обраного тарифного плану [25].

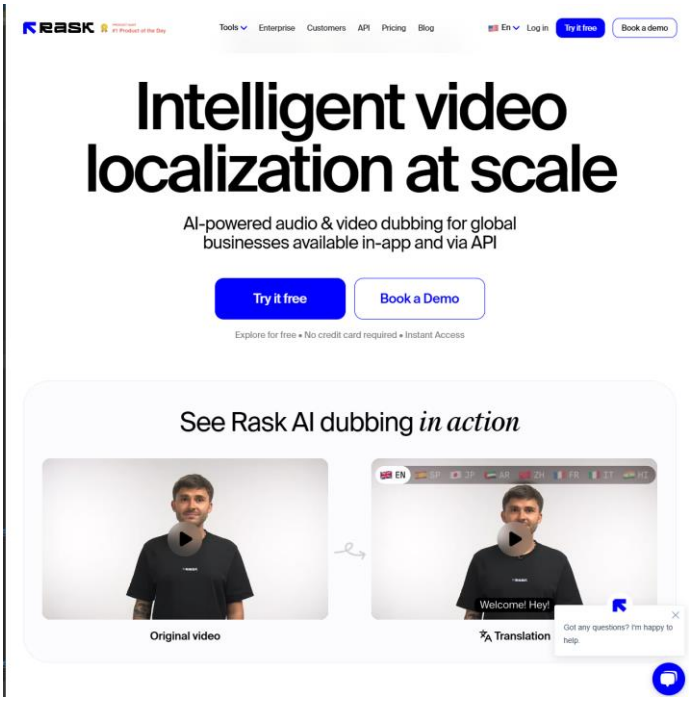

Рисунок 1.7 — Загальний вигляд інтерфейсного вікна. Rask AI

HeyGen спеціалізується на безшовній локалізації відео з підтримкою понад 40 мов. Система пропонує функції клонування голосу та синхронізації губ, що забезпечує природне звучання перекладеного контенту. Це дозволяє створювати відео, які виглядають і звучать автентично на різних мовах. HeyGen підходить для творців контенту, які прагнуть зробити свої відео доступними для глобальної аудиторії. Вартість послуг залежить від обраного тарифного плану та обсягу використання [26].

Maestra AI — це інструмент, який пропонує переклад аудіо та відео з високою точністю. Він використовує вдосконалений штучний інтелект для безперебійного перекладу аудіоконтенту кількома мовами. Maestra AI ідеально підходить для творців контенту, які хочуть зробити свої відео доступними для глобальної аудиторії. Ключові особливості включають підтримку понад 50 мов та автоматичну генерацію субтитрів. Вартість послуг залежить від обраного тарифного плану та обсягу використання [27].

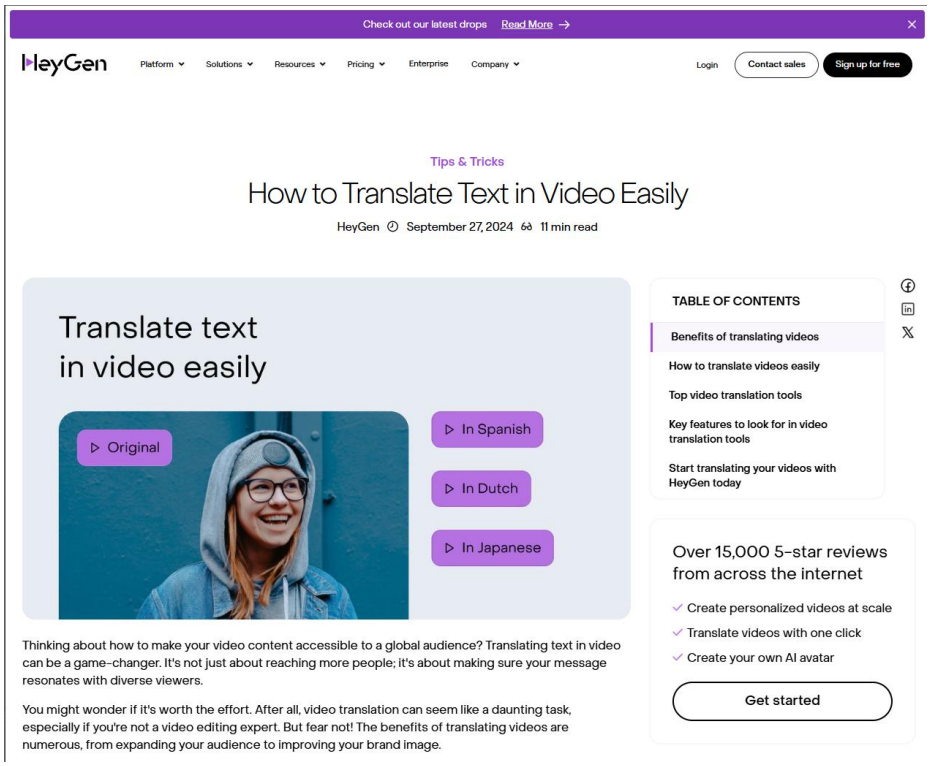

Рисунок 1.8 — Загальний вигляд інтерфейсного вікна HeyGen.

Verbalate — це програмне забезпечення для перекладу відео, яке пропонує функції повного клонування голосу та синхронізації губ. Це дозволяє перекладати відеоконтент кількома мовами, зберігаючи автентичність оригінального матеріалу. Verbalate підходить для онлайн-курсів, маркетингових відео та іншого контенту, спрямованого на міжнародну аудиторію. Вартість послуг залежить від обраного тарифного плану та обсягу використання [28].

Згідно таблиці 1.1 система автоматичного перекладу аудіо та відео вирізняється своєю ефективністю та доступністю. Обрані характеристики дозволяють комплексно оцінити ефективність систем автоматичного перекладу відео.

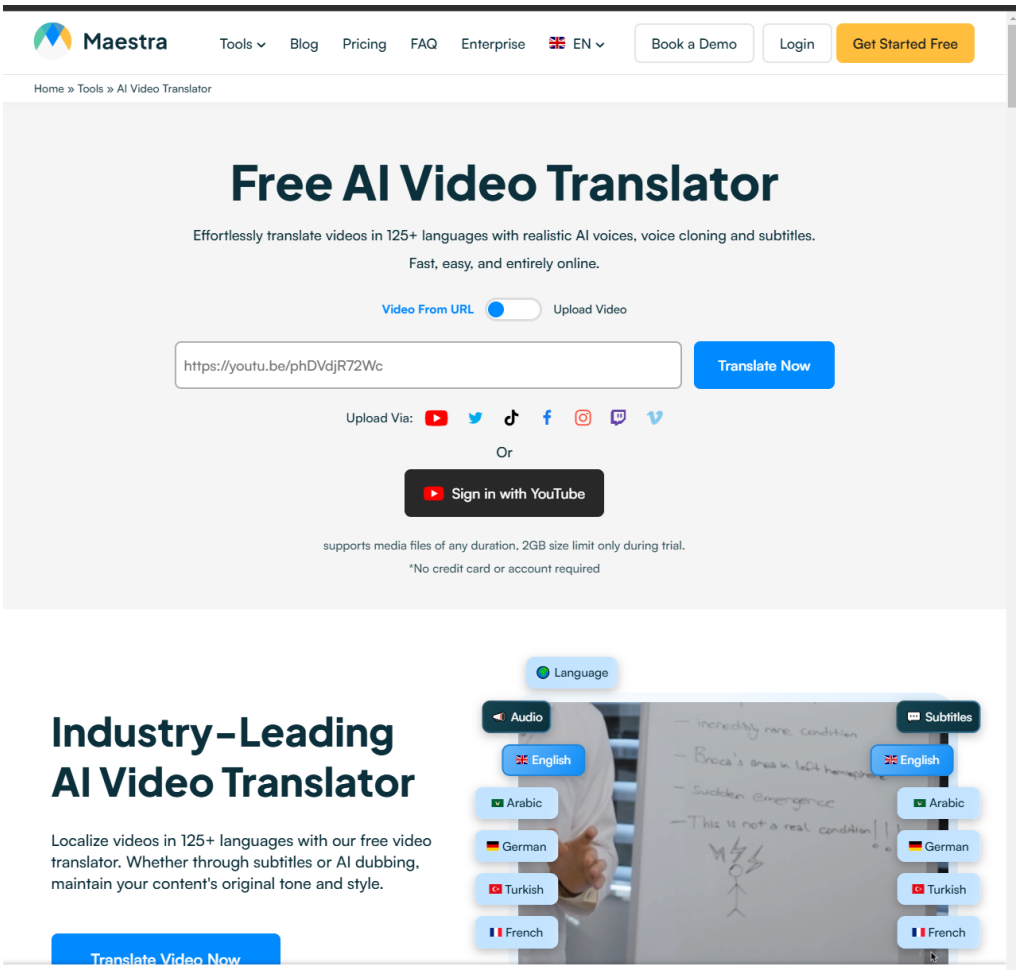

Рисунок 1.9 — Загальний вигляд інтерфейсного вікна. Maestra AI

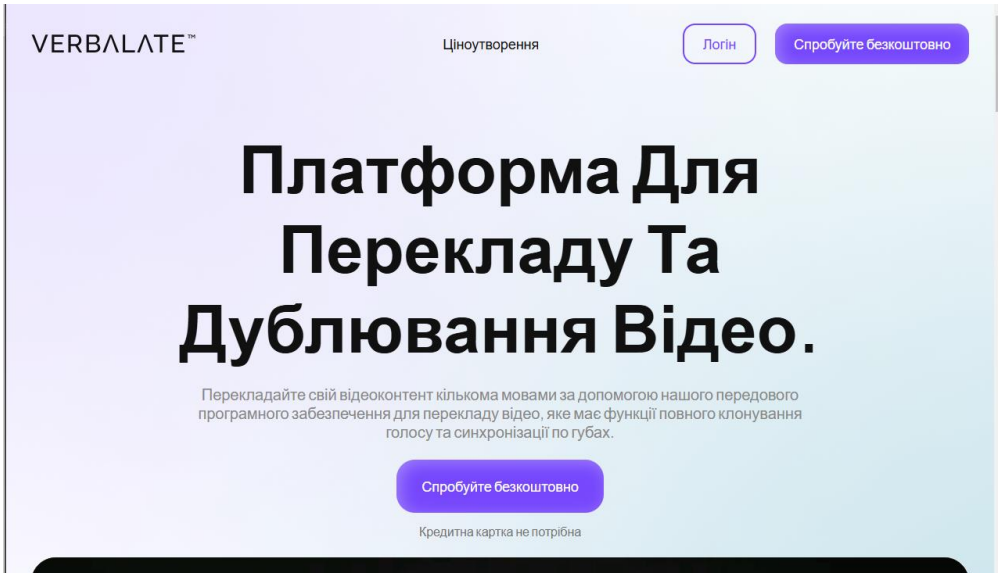

Рисунок 1.10 — Загальний вигляд інтерфейсного вікна. Verbalate.

Точність перекладу є ключовим фактором, що визначає якість кінцевого продукту. Обмеження на розмір файлу важливі для зручності використання, оскільки більші файли часто потребують додаткових ресурсів для обробки. Порівняно з аналогами, які часто мають обмеження, такі як висока вартість, обмеження на розмір файлу чи недостатня точність, дана система компенсує ці недоліки завдяки підтримці понад 50 мов, високій швидкості обробки та відсутності обмежень на розмір завантажуваних файлів.

Кількість мов важливим параметром, оскільки широкий мовний охоплює забезпечує доступ до глобальної аудиторії. Вартість визначає доступність системи для користувачів різного масштабу, від окремих осіб до великих компаній. Швидкість обробки впливає на продуктивність, особливо при роботі з великими обсягами даних. Інтеграція технологій Whisper і Coqui TTS забезпечує точність розпізнавання мовлення, діаризацію та генерацію субтитрів. Це робить систему універсальним рішенням, що сприяє розширенню доступу до мультимедійного контенту та зменшенню мовних бар'єрів у глобальному масштабі.

Таблиця 1.1 Порівняльна таблиця характеристик систем автоматичного перекладу відео

Характеристика	Rask AI	Maestra AI	HeyGen	Verbalate	Моя система
Підтримувані мови	100+	50+	Кілька	40+	70+
Вартість	Від $60/місяць	Залежить від плану	Залежить від плану	Від $9/місяць	Безкоштовно
Швидкість обробки	Висока	Висока	Середня	Висока	Висока
Точність перекладу	Середня	Висока	Середня	Висока	Висока
Обмеження на розмір файлу	До 200 МБ	До 1 ГБ	Залежить від плану	Залежить від плану	Без обмежень

2 ДОСЛІДЖЕННЯ МЕТОДІВ ДЛЯ АВТОМАТИЧНОГО ПЕРЕКЛАДУ ВІДЕОФАЙЛІВ

2.1 Алгоритми та моделі машинного перекладу

Машинний переклад, як область дослідження, виник у середині XX століття. Перші ідеї автоматизованого перекладу базувалися на прямому трансформуванні слів однієї мови в іншу за допомогою словників. Однак цей підхід швидко виявив свою обмеженість через складність синтаксичних і семантичних конструкцій мов. Це призвело до створення нових алгоритмічних підходів, що враховують правила граматики, семантику та контекст.

Ранні системи машинного перекладу використовували так звані прямі методи. Алгоритм перекладу полягав у простій заміні слів джерельної мови еквівалентними словами мови перекладу. Прикладом математичного опису такого підходу є формула для частотного аналізу слів:

$$P(e \mid f) = \frac{count(e, f)}{count(f)}, \tag{2.1}$$

де $P(e \mid f)$— ймовірність того, що слово f у тексті мови джерела відповідає слову e у цільовій мові;

$count(e, f)$— кількість спільних вживань цих слів у паралельних текстах.

Проте, цей підхід швидко ставав неефективним через обмеженість правил, що не враховували контексту.

У 1970–1980-х роках почали з'являтися статистичні моделі перекладу, які ґрунтувалися на паралельних корпусах текстів. Головною ідеєю стало використання теорії ймовірностей для визначення найкращого варіанту перекладу. Однією з перших таких моделей стала IBM Model 1, яка застосовувала принцип максимальної ймовірності:

$$P(e \mid f) = \prod_{i=1}^{n} P(e_i|f_i), \quad (2.2)$$

де e_i — слово цільової мови;

f_i — слово мови джерела;

$P(e_i|f_i)$ — ймовірність перекладу f_i у e_i.

Перевагою цієї моделі стала можливість автоматично будувати ймовірнісні відповідності між словами без необхідності жорстких правил. Проте вона не враховувала порядок слів, що було критично для мов із жорстким синтаксичним порядком.

У 1990-х роках з'явилися алгоритми, що враховували синтаксис і контекст. Одним із таких підходів стала фразова модель перекладу, яка працювала не зі словами, а з цілими фразами. Математично це описувалося через сегментацію джерельного тексту:

$$P(e \mid f) = \prod_{k=1}^{K} P(e_k|f_k) \cdot d(a_k, a_{k-1}), \quad (2.3)$$

де e_k і f_k— відповідні фрази цільової і оригінальної мов;

$d(a_k, a_{k-1})$ — функція вирівнювання, яка враховує відстань між фразами у текстах.

Паралельно із цим розвивалися статистичні моделі на основі декодування, такі як алгоритм Вітербі, що обчислював найімовірніший шлях перекладу через простір можливих відповідностей. Блок-схема такого підходу на рисунку 2.1.

У 2010-х роках відбувся значний прорив у машинному перекладі завдяки використанню нейронних мереж. Спочатку широко застосовували рекурентні

нейронні мережі (RNN), які були здатні моделювати залежності між словами у джерельному і цільовому текстах.

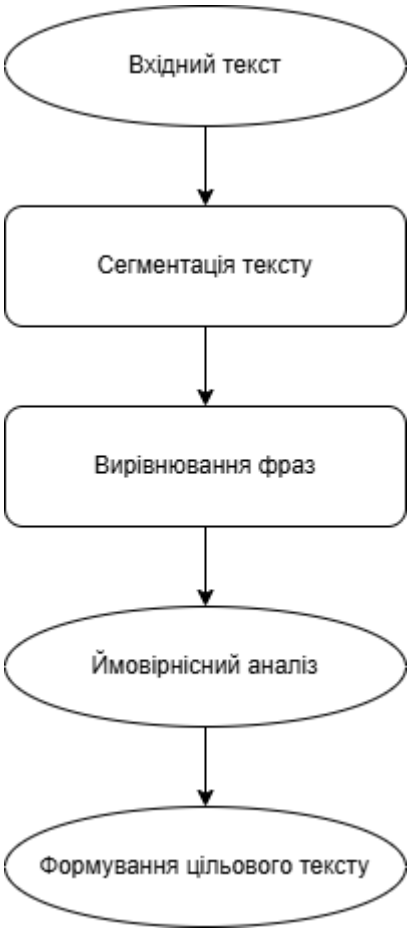

Рисунок 2.1 — Алгоритм Вітербі

Особливістю RNN було зберігання контексту, що дозволяло системам перекладу краще враховувати структуру мовних одиниць. Основною математичною моделлю для RNN була наступна формула:

$$h_t = \sigma(W_h \cdot x_t + U_h \cdot h_{t-1} + b_h), \quad (2.5)$$

де h_t— прихований стан у момент часу t;

x_t— вхідний сигнал;

W_h та U_h — ваги мережі;

b_h — зсув.

Для перекладу ці мережі доповнювалися механізмом декодування, що передбачав побудову послідовності вихідних слів на основі ймовірностей. Однак RNN мали труднощі з обробкою довгих залежностей, що впливало на якість перекладу, особливо для текстів із складною структурою [29–33].

Усунути ці обмеження вдалося завдяки впровадженню моделей на основі трансформерів. Трансформери використовують механізм самоуваги, який дозволяє моделі звертати увагу на релевантні частини тексту, незалежно від їхнього положення. Це робить їх значно ефективнішими для роботи з довгими текстами. Формула для обчислення механізму уваги:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.4)$$

де Q, K, V — матриці запитів, ключів та значень;

d_k — розмірність ключів.

Трансформери використовуються як для енкодера, що аналізує текст джерела, так і для декодера, який генерує текст перекладу, завдяки своїй універсальній архітектурі. Ця модель стала основою сучасних систем машинного перекладу, таких як BERT, GPT та T5, що демонструють високий рівень точності та продуктивності. Механізм уваги в трансформерах значно спрощує обчислення у порівнянні з рекурентними моделями, оскільки дозволяє обробляти вхідний текст паралельно, зберігаючи послідовність аналізу. Завдяки цьому час навчання значно скорочується, що дозволяє працювати з великими обсягами даних. Крім того, трансформери забезпечують кращу узгодженість перекладу, роблячи текст більш природним і контекстуально точним. Хронологічний розвиток перекладу показаний у таблиці 2.1.

Таблиця 2.1 — Таблиця порівняння підходів до машинного перекладу

Підхід	Часовий період	Головна особливість	Точність перекладу	Обчислювальна складність
Прямий переклад	1950–1960	Словники	Низька	Низька
Статистичні моделі	1970–1990	Ймовірності	Середня	Середня
Фразові моделі	1990–2010	Фразове вирівнювання	Вища	Висока
Рекурентні нейронні мережі	2010–2015	Контекстуальність	Висока	Висока
Трансформери	2015–дотепер	Самоувага	Дуже висока	Помірна

Для оцінки якості перекладу використовують кілька метрик, що враховують точність, плавність та адекватність перекладу.

— BLEU (Bilingual Evaluation Understudy) вимірює схожість між машинним перекладом і одним чи кількома референсними перекладами він використовує модифікований прецедент н-грам:

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \quad (2.6)$$

де BP — штраф за довжину;

p_n — точність н-грам;

w_n — ваги;

— METEOR (Metric for Evaluation of Translation with Explicit ORdering): ця метрика враховує не лише точність н-грам, але й синонімічні заміни та порядок слів;

— TER (Translation Error Rate): оцінює кількість правок, необхідних для перетворення машинного перекладу на референсний;

— ROUGE (Recall-Oriented Understudy for Gisting Evaluation) вимірює накладання між машинним перекладом і референсом, особливо для резюме;

— GLEU (Google's Bilingual Evaluation Understudy) модифікація BLEU, враховує унікальні помилки і гнучкість перекладу;

— CHR (Character-level Translation Error Rate) порівнює переклад на рівні символів, що є корисним для оцінки орфографії.

BLEU залишається найпопулярнішою метрикою, оскільки вона добре узгоджується з людськими оцінками. Проте METEOR більш гнучкий завдяки урахуванню синонімів. TER і GLEU використовуються для порівняння моделей, тоді як CHR актуальний для оцінки морфологічних мов [34–36].

2.2 Порівняння технологій для розпізнавання мови

Розпізнавання мови — це технологія, яка еволюціонувала протягом десятиліть, переходячи від базових алгоритмів обробки сигналів до сучасних моделей глибокого навчання. Її розвиток охоплює кілька ключових етапів.

На ранніх стадіях розпізнавання мови базувалося на статистичних моделях і алгоритмах обробки сигналів. Найперші системи були розроблені у 1950-х роках і мали обмежену функціональність, здатні розпізнавати лише окремі слова або цифри. Одним із перших підходів був алгоритм шаблонного зіставлення. У ньому кожне слово представлялося у вигляді акустичного шаблону, який порівнювався із введеним сигналом. Попри простоту, цей метод був обмеженим через високу чутливість до змін у вимові, шуму та акустичних умов.

Ключовою концепцією в обробці мовлення стало використання частотного аналізу. Для перетворення мовного сигналу з часової області у частотну застосовувалося швидке перетворення Фур’є :

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dtX(f), \quad (2.7)$$

де $X(f)$ — спектр сигналу;

$x(t)$ — мовний сигнал у часовій області;

f — частота.

Для обробки сигналу також почали використовувати мел-кепстральні коефіцієнти (MFCC), які забезпечували стійкість до шуму та дозволяли стисло описувати мовні ознаки:

$$C_n = \sum_{k=1}^{K} \log(S_k) \cos\left(n \cdot \frac{k-0.5}{K}\pi\right), \quad (2.8)$$

де S_k — енергія сигналу в k-му каналі мел-шкали;

C_n — коефіцієнт.

У 1970-х роках з’явилися перші системи на основі прихованих моделей Маркова (HMM). Вони дозволяли моделювати ймовірність переходу між станами, що відповідали звукам або фонемам у мовленні:

$$P(W \mid O) = \prod_{t=1}^{T} P(o_t|s_t)P(s_t|s_{t-1}) \quad (2.9)$$

де $P(o_t|s_t)$ — ймовірність спостереження o_t в стані s_t;

$P(s_t|s_{t-1})$ — ймовірність переходу між станами.

HMM стали основою багатьох систем завдяки їхній здатності враховувати часову структуру мовного сигналу. Ці моделі поєднувалися з гаусовими сумішами (GMM) для моделювання акустичних характеристик, що забезпечувало прийнятну точність у розпізнаванні мови.

З 2010-х років розпізнавання мови значно покращилося завдяки впровадженню глибоких нейронних мереж (DNN). Архітектури, такі як рекурентні нейронні мережі (RNN) та їх розширення — довготривалі короткочасні пам'яті (LSTM).. Рівняння для обчислення стану LSTM:

$$h_t = o_t \tanh(c_t), \tag{2.10}$$

де h_t— вихідний стан;

o_t— вихідний шлюз;

c_t— стан пам’яті.

LSTM дозволили значно підвищити точність систем розпізнавання мови, оскільки враховували довготривалі залежності у мовному сигналі [37–39].

Пізніше були введені згорткові нейронні мережі для обробки спектрограм мовлення. Важливим кроком стало створення трансформерів, таких як модель Whisper від OpenAI. Трансформери працюють з великими обсягами контексту, що дозволяє їм ефективно враховувати мовні залежності та забезпечувати точність навіть у складних акустичних умовах.

Етапи розпізнавання мови вказані на рисунку 2.2

— захоплення аудіосигналу: перший крок, де мовлення записується за допомогою пристроїв;

— попередня обробка: усунення шумів, нормалізація амплітуди сигналу;

— перетворення сигналу: перехід від часової до частотної області;

— виділення ознак (MFCC): втримання стійких до шумів ознак;

— моделювання мови: прогнозування найбільш імовірних послідовностей слів;

— трансформація в текст: пормування тексту з урахуванням мовних правил.

Розпізнавання мови пройшло довгий шлях від простих алгоритмів до потужних моделей на основі глибокого навчання. Сучасні системи, такі як Whisper, Google Speech-to-Text, IBM Watson Speech to Text, Microsoft Azure Speech Service чи DeepSpeech використовують трансформери, які забезпечують виняткову точність і можливість обробки багатомовного контенту. Еволюція цієї технології значно розширила можливості її застосування, відкривши шлях до нових інновацій.

Рисунок 2.2 — Етапи розпізнавання мови

Google Speech-to-Text є потужним сервісом для розпізнавання мови, розробленим компанією Google. Завдяки використанню глибоких нейронних мереж, цей інструмент забезпечує високу точність навіть у складних умовах запису. Технологія підтримує понад 120 мов і діалектів, що дозволяє застосовувати її в багатьох регіонах і для різних цільових аудиторій. Важливим аспектом роботи Google Speech-to-Text є його адаптація до шуму та можливість використання спеціальних голосових моделей, які налаштовуються для певних галузей або сценаріїв. Наприклад, це може бути корисно для автоматизації роботи контактних центрів або створення субтитрів у реальному часі.

Інтеграція через API робить сервіс доступним для широкого спектра розробників, які можуть включити функціонал розпізнавання мови в свої додатки або сервіси. Завдяки потужній інфраструктурі Google Cloud, обробка мовлення здійснюється швидко та без затримок. Особливо це важливо для завдань, які вимагають миттєвого реагування, таких як розпізнавання команд в інтерактивних системах або аналіз мовлення в поточному режимі.

Проте, незважаючи на численні переваги, Google Speech-to-Text має певні обмеження. По-перше, сервіс залежить від постійного інтернет-з’єднання, оскільки обробка мовлення відбувається на хмарних серверах. Це може бути проблемою для користувачів, які працюють у віддалених або ненадійних мережах. По-друге, для великих обсягів аудіоданих послуга може стати дорогою, що робить її менш привабливою для масштабних проектів. Вартість також залежить від тривалості аудіофайлів, які потрібно обробляти, і кількості запитів до сервісу.

Варто також зазначити, що для конфіденційних даних використання хмарного рішення може викликати питання щодо безпеки, оскільки аудіодані передаються до зовнішніх серверів для обробки. Попри це, Google активно працює над удосконаленням своїх механізмів безпеки, пропонуючи шифрування та інші заходи для захисту інформації. У цілому, Google Speech-to-Text є

потужним інструментом, який найкраще підходить для проєктів, де потрібна висока точність і багатомовність, за умови доступу до стабільного інтернет-з'єднання.

IBM Watson Speech to Text — це хмарний сервіс, розроблений компанією IBM, який використовує передові алгоритми машинного навчання та штучного інтелекту для перетворення аудіо в текст. Цей сервіс підтримує кілька мов, що робить його універсальним інструментом для користувачів по всьому світу. Завдяки використанню технологій обробки природної мови (NLP), IBM Watson Speech to Text здатний точно розпізнавати та транскрибувати усне мовлення, навіть у складних акустичних умовах.

Технологія працює шляхом аналізу вхідного аудіосигналу та його перетворення в текстовий формат. Цей процес відбувається безперервно, що дозволяє отримувати точні транскрипції швидко та ефективно, роблячи IBM Watson Speech to Text незамінним інструментом у різних галузях.

Однією з ключових особливостей IBM Watson Speech to Text є підтримка кількох мов та діалектів, включаючи англійську, іспанську, французьку, німецьку та мандаринську. Це забезпечує глобальну доступність та гарантує, що користувачі з різних мовних середовищ можуть скористатися сервісом. Крім того, сервіс пропонує можливість реального часу транскрибування, що особливо корисно для додатків, які потребують миттєвого перетворення мовлення в текст.

IBM Watson Speech to Text також може розрізняти різних спікерів в аудіофайлі, що спрощує транскрибування розмов з кількома учасниками. Ця функція є надзвичайно корисною для інтерв'ю, панельних дискусій та багатоспікерних зустрічей. Сервіс автоматично додає пунктуацію та форматування до транскрибованого тексту, що призводить до більш читабельного результату та економить час користувачів, оскільки їм не потрібно вручну редагувати текст для додавання пунктуації.

Використання IBM Watson Speech to Text приносить численні переваги, зокрема підвищення ефективності, оскільки автоматизація процесу транскрибування дозволяє користувачам заощаджувати значний час та ресурси. Ця ефективність дозволяє професіоналам зосередитися на більш важливих завданнях, а не витрачати години на ручне введення нотаток або транскрипцій. Крім того, сервіс підвищує доступність для осіб з порушеннями слуху, надаючи їм доступ до транскрибованого тексту з аудіоконтенту, що значно покращує їхню здатність взаємодіяти з інформацією.

Завдяки передовим алгоритмам машинного навчання, IBM Watson Speech to Text забезпечує високий рівень точності транскрибування, що є критично важливим для бізнесу та професіоналів, які потребують точного документування розмов та зустрічей. Використання хмарного сервісу, такого як IBM Watson Speech to Text, може бути більш економічно вигідним, ніж наймання професійних транскрибаторів, що робить його привабливим варіантом для стартапів, фрілансерів та малого бізнесу. Крім того, сервіс легко масштабується, що дозволяє обробляти більші обсяги аудіоконтенту без компромісів щодо продуктивності чи якості.

IBM Watson Speech to Text знаходить застосування в різних галузях, включаючи бізнес, медіа, журналістику, освіту, правову сферу та охорону здоров'я. Наприклад, у бізнесі транскрибування зустрічей та конференцій дозволяє учасникам зосередитися на обговореннях без необхідності ведення нотаток, забезпечуючи точне документування важливої інформації. Журналісти можуть скористатися транскрибуванням в реальному часі під час інтерв'ю та прес-конференцій, що дозволяє швидко створювати точні цитати та звіти, підвищуючи їхню продуктивність та ефективність.

В освітній сфері студенти можуть використовувати IBM Watson Speech to Text для транскрибування лекцій та класних дискусій, що полегшує вивчення та перегляд матеріалу пізніше. Ця технологія також може допомогти викладачам у

створенні доступного контенту для всіх студентів. У правовій сфері точне транскрибування є життєво важливим, і юристи можуть використовувати IBM Watson Speech to Text для транскрибування судових засідань, депозицій та зустрічей з клієнтами, забезпечуючи повне документування всіх процесів.

В охороні здоров'я професіонали можуть використовувати технологію перетворення мовлення в текст для ефективного транскрибування нотаток пацієнтів та медичних записів, що не тільки економить час, але й покращує точність документації пацієнтів.

IBM Watson Speech to Text пропонує безкоштовний рівень з обмеженим використанням, що дозволяє користувачам зареєструватися та ознайомитися з можливостями сервісу. Для вищих рівнів використання доступні платні плани. Сервіс також надає API, що дозволяє розробникам інтегрувати його з різними додатками та платформами, створюючи індивідуальні рішення, які використовують можливості розпізнавання мовлення.

Microsoft Azure Speech Service — це комплексна платформа, розроблена компанією Microsoft, яка надає можливості перетворення мовлення в текст, синтезу тексту в мовлення, перекладу мовлення та розпізнавання мовця. Вона інтегрується з іншими сервісами Azure, що дозволяє створювати багатомовні додатки з підтримкою голосу, швидким транскрибуванням та природним звучанням на основі штучного інтелекту.

Однією з ключових особливостей Azure Speech Service є підтримка понад 100 мов та діалектів, що робить її універсальним інструментом для глобальних застосувань. Сервіс дозволяє здійснювати транскрибування мовлення в текст з високою точністю, створювати природно звучащі голоси з тексту, перекладати усне мовлення та використовувати розпізнавання мовця під час розмов.

Azure Speech Service також пропонує можливість створення настроюваних голосів, додавання специфічних слів до базового словника або розробки власних моделей. Це дозволяє адаптувати сервіс під конкретні потреби бізнесу або

проекту. Крім того, сервіс може працювати як у хмарі, так і на периферії в контейнерах, що забезпечує гнучкість у розгортанні та використанні.

Інтеграція Azure Speech Service з іншими продуктами Microsoft, такими як Power BI, дозволяє створювати комплексні рішення для бізнес-аналітики та автоматизації процесів. Сервіс також підтримує реальне транскрибування, що особливо корисно для додатків, які потребують миттєвого перетворення мовлення в текст.

Однак, як і будь-який хмарний сервіс, Azure Speech Service залежить від стабільного інтернет-з'єднання. Крім того, для великих обсягів даних вартість використання сервісу може бути значною, що слід враховувати при плануванні проектів. Незважаючи на це, завдяки своїй функціональності та інтеграційним можливостям, Azure Speech Service є потужним інструментом для розробки су

DeepSpeech — це відкрите програмне забезпечення для розпізнавання мови, розроблене Mozilla Foundation. Воно базується на глибоких нейронних мережах та є частиною проекту Common Voice, спрямованого на розвиток доступних технологій обробки мовлення. DeepSpeech використовує архітектуру рекурентних нейронних мереж (RNN) з багонаправленими довготривалими короткочасними пам'ятями (LSTM), що дозволяє ефективно обробляти послідовності мовного сигналу та точно передбачати слова.

Однією з основних переваг DeepSpeech є його відкритий вихідний код, що робить платформу доступною для дослідників, розробників і стартапів. На відміну від комерційних сервісів, DeepSpeech дозволяє повністю локальне використання, що ідеально підходить для проектів, які потребують обробки конфіденційних даних. Ця особливість також усуває залежність від інтернет-з'єднання, що важливо для систем, які працюють у віддалених або ізольованих середовищах.

Проект активно підтримує багатомовність, проте кількість мов, доступних «з коробки», є обмеженою. Для розширення підтримки розробники можуть

навчати моделі на власних даних, що відкриває широкі можливості для адаптації до різних галузевих потреб. Однак навчання моделей потребує значних обчислювальних ресурсів, що може стати викликом для невеликих команд або окремих ентузіастів.

Важливою особливістю DeepSpeech є його здатність працювати на різних платформах, включаючи Windows, macOS, Linux та мобільні пристрої. Це забезпечує високу гнучкість у виборі середовища для розгортання системи. Серед недоліків слід зазначити нижчу точність розпізнавання у порівнянні з комерційними хмарними сервісами, такими як Google Speech-to-Text або IBM Watson. Проте це може бути компенсовано налаштуванням та донавчанням моделей.

DeepSpeech ідеально підходить для дослідницьких проектів, освітніх програм і додатків, які потребують автономної роботи. Завдяки відкритості платформи, її можна інтегрувати з іншими інструментами та бібліотеками для створення інноваційних рішень. Таким чином, DeepSpeech є важливим внеском у демократизацію технологій розпізнавання мови та розвиток відкритого програмного забезпечення у цій галузі.

Whisper — це сучасна система розпізнавання мовлення, розроблена OpenAI, яка використовує архітектуру трансформерів для перетворення аудіосигналів у текст. Її основною перевагою є відкрите програмне забезпечення, що дозволяє використовувати Whisper як для дослідницьких проектів, так і для комерційних додатків. Ця технологія поєднує у собі високу точність розпізнавання, можливість локального використання та підтримку багатьох мов, що робить її універсальним інструментом для роботи зі звуковими даними.

Whisper здатний виконувати не лише стандартне транскрибування мовлення, але й розпізнавати мову оригіналу в мультимовному контенті, забезпечуючи автоматичний переклад тексту. Це розширює можливості системи та дозволяє використовувати її для задач, які потребують обробки багатомовних

аудіофайлів. Додатково Whisper підтримує функцію діаризації, що дає змогу розрізняти мовців у записі. Це особливо корисно для транскрибування конференцій, інтерв'ю або багатоспікерних нарад.

На відміну від багатьох комерційних рішень, Whisper може працювати повністю автономно, без необхідності передачі даних у хмару. Це дозволяє зберігати конфіденційність інформації, що є критично важливим для деяких галузей, таких як медицина, право чи фінанси. Крім того, локальне використання усуває залежність від інтернет-з’єднання, що робить систему надійною навіть у складних умовах. Система також оптимізована для запуску на різних апаратних платформах, включаючи стандартні персональні комп’ютери та сервери з GPU, що сприяє її широкому застосуванню.

Однією з ключових особливостей Whisper є висока точність навіть для складних аудіо, наприклад, записів з фоновим шумом або низькою якістю. Ця точність досягається завдяки використанню великих обсягів навчальних даних та потужної архітектури моделі. Проте варто зазначити, що для обробки великих аудіофайлів може знадобитися значна обчислювальна потужність, особливо якщо модель використовується у режимі реального часу.

Whisper знаходить застосування у різних галузях, включаючи автоматизацію створення субтитрів, інтерактивні голосові інтерфейси, створення транскрипцій для медіаконтенту та багато іншого. Завдяки своїй відкритості система легко інтегрується з іншими інструментами та платформами, що сприяє її адаптації для специфічних потреб користувачів. Whisper не лише підвищує доступність технологій розпізнавання мовлення, але й створює нові можливості для інновацій, дозволяючи спільноті активно брати участь у розвитку цієї галузі [40–44].

За результатами аналізу Таблиці 2.2, було вирішено використовувати Whisper як основний інструмент у системі автоматичного перекладу відеофайлів.

Цей вибір обґрунтований кількома ключовими причинами. Перш за все, Whisper забезпечує високий рівень точності розпізнавання мовлення, навіть у складних акустичних умовах, таких як фоновий шум або низька якість запису. Це є важливим фактором для роботи з відеофайлами, які часто записуються у реальних умовах, що не завжди забезпечують ідеальну якість звуку.

Таблиця 2.2 —Порівняльна таблиця інструментів для розпізнавання мови

Характеристика	Whisper	Google STT	IBM Watson	Azure Speech	DeepSpeech
Точність розпізнавання	5	5	4	4	4
Підтримка мов	4	5	4	4	3
Локальне використання	5	1	1	1	5
Швидкість роботи	4	5	4	4	3
Вартість	4	3	3	3	5
Легкість інтеграції	4	5	4	4	4
Додаткові функції	5	4	4	4	3

Другою вагомою перевагою є підтримка локального використання, що дозволяє уникнути передачі конфіденційних даних у хмарні сервіси. Це критично важливо для забезпечення приватності, особливо у випадках, коли відеофайли містять чутливу інформацію. Локальна обробка також усуває залежність від інтернет-з’єднання, роблячи систему більш надійною.

Whisper також вирізняється своєю здатністю працювати з багатьма мовами, що є необхідним для створення багатомовної системи перекладу. Завдяки підтримці мультимовного розпізнавання та можливості автоматичного перекладу, цей інструмент підходить для роботи з відеофайлами, які містять контент на різних мовах.

Ще одним визначальним фактором стало те, що Whisper є відкритим програмним забезпеченням, що дозволяє його адаптацію до специфічних потреб проєкту. Це розширює можливості розробки та інтеграції з іншими компонентами системи. Крім того, його автономність та відсутність значних ліцензійних витрат роблять Whisper економічно вигідним рішенням порівняно з комерційними хмарними сервісами.

Враховуючи всі ці переваги, Whisper є найбільш збалансованим вибором за такими критеріями, як точність, гнучкість, конфіденційність і вартість. Таким чином, його використання забезпечить надійну роботу системи автоматичного перекладу відеофайлів та відповідність ключовим вимогам проєкту.

2.3 Методи обробки відеофайлів

Відеофайли — це складні мультимедійні структури, що об'єднують відеопотоки, аудіопотоки, субтитри та метадані у єдиному контейнері. Популярні формати включають MP4, AVI, MKV, MOV, та FLV. Кожен формат має свої особливості у роботі та зберіганні даних.

MP4 є одним із найпоширеніших форматів, який використовує кодек H.264 для стиснення відео і AAC для аудіо. Цей формат зручний завдяки високій якості при відносно невеликому розмірі файлів. Він підтримується більшістю сучасних пристроїв і платформ. У контексті Whisper, MP4 зазвичай використовується для виділення аудіопотоку з метою подальшого розпізнавання мовлення.

AVI — це старший формат, який зберігає відео- та аудіопотоки з мінімальним стисненням. Завдяки цьому AVI зберігає якість, але створює значні обсяги файлів. Він менш зручний для інтеграції з Whisper через великий розмір і обмежену підтримку сучасних кодеків.

MKV — сучасний відкритий формат, який підтримує широкий спектр кодеків і дозволяє зберігати кілька аудіодоріжок, субтитри і навіть глави. Цей

формат ідеальний для мультимовного контенту, проте потребує додаткової оптимізації при роботі з бібліотеками на кшталт FFmpeg.

MOV, розроблений Apple, забезпечує високу якість відео, але обмежений підтримкою у системах, відмінних від macOS. FLV використовується переважно для стрімінгових платформ і менш придатний для обробки через обмежену підтримку сучасних бібліотек [45, 46].

FFmpeg є інструментом для роботи з відеоформатами у процесі інтеграції Whisper. Ця бібліотека дозволяє зчитувати та конвертувати мультимедійні потоки у потрібний формат. Для розпізнавання мовлення зазвичай виділяється лише аудіо з відповідною частотою дискретизації (16 кГц), що є стандартним для більшості моделей. . Нижче наведено покроковий опис цього процесу на основі схем, представлених на рисунку 2.3.

На першому етапі користувач завантажує відеофайл у систему через графічний інтерфейс або API. Система аналізує формат файлу, використовуючи бібліотеку FFmpeg, і за потреби конвертує його у сумісний формат, наприклад, MP4. Потім перевіряється наявність аудіопотоку у файлі. Якщо аудіо відсутнє, обробка припиняється, і користувач отримує повідомлення про помилку. У разі кількох аудіодоріжок система вибирає основну.

Для подальшої обробки за допомогою FFmpeg виділяється аудіопотік, який конвертується у WAV із частотою дискретизації 16 кГц для забезпечення сумісності з моделлю Whisper. Аудіосигнал проходить попередню обробку: нормалізацію, фільтрацію шумів та видалення артефактів, щоб покращити точність розпізнавання.

Оброблений аудіофайл передається у модель Whisper для розпізнавання мовлення. Whisper виконує аналіз звукового сигналу та перетворює його у текст. Результат перевіряється на якість: аналізується коректність тексту, логіка порядку слів і цілісність фрагментів. На цьому ж етапі текст проходить

постпроцесинг — додаються пунктуаційні знаки, форматуються речення, виправляються помилки.

Рисунок 2.3 — Схема алгоритму роботи обробки відеофайлу

Останнім кроком є формування фінального тексту, який може бути збережений у вигляді субтитрів (SRT), простого тексту чи структурованого JSON-файлу. Оброблений текст інтегрується у відеофайл або зберігається окремо відповідно до вибору користувача [47].

2.4 Технології синтезу мовлення

Синтез мовлення, або text-to-speech (TTS), — це технологія, яка перетворює текстову інформацію на синтезований голос, що звучить як природна людська мова. Ця технологія дозволяє комп'ютерам або мобільним пристроям відтворювати текст у звуковій формі, що має велике значення для автоматизації комунікації між людьми та машинами.

Основна мета створення синтезу мовлення — зробити інформацію доступнішою для сприйняття, особливо для людей із порушеннями зору або нездатністю читати текст. Такі системи стали важливим компонентом допоміжних технологій, які полегшують доступ до текстових ресурсів для осіб із різними обмеженнями. Крім того, TTS-технології активно використовуються у різних комерційних і промислових сферах, таких як навігація, розумні асистенти, автоматичні довідкові системи та створення аудіокниг.

Синтез мовлення був створений для вирішення завдання автоматизації мовленнєвих інтерфейсів, що стало особливо актуальним із розвитком штучного інтелекту та необхідністю створювати більш інтерактивні системи. Наприклад, у сфері телефонного обслуговування клієнтів системи TTS дозволяють надавати інформацію голосом, економлячи час операторів. У навігаційних системах синтез мовлення допомагає водіям отримувати чіткі голосові вказівки, не відволікаючись на екран.

Ця технологія розроблялася також для розширення можливостей автоматичного перекладу. У поєднанні з машинним перекладом синтез мовлення

дозволяє не лише перекладати текст, а й озвучувати його мовою перекладу, що стає важливим у мультимовних середовищах.

Синтез мовлення є критичним компонентом для інтерактивних голосових асистентів, таких як Siri, Google Assistant, і Alexa. Він забезпечує природну взаємодію між користувачем і пристроєм, дозволяючи системі відповідати голосом на текстові або голосові запити. Таким чином, TTS створює відчуття природної комунікації, що є важливим для підвищення рівня зручності користувачів.

На додаток до практичних застосувань, технологія синтезу мовлення має значний потенціал у творчих галузях. Наприклад, її використовують для створення голосів персонажів у відеоіграх і фільмах, а також для генерації унікальних голосів у розважальних системах. Таким чином, TTS є багатогранною технологією, що знаходить застосування в широкому спектрі завдань, підвищуючи доступність і зручність обробки текстової інформації.

Синтез мовлення — це процес перетворення текстової інформації у звуковий сигнал, що імітує природну людську мову. З алгоритмічної точки зору цей процес складається з кількох основних етапів, які включають текстову обробку, фонетичний аналіз і генерацію аудіосигналу. Ранні підходи до синтезу мовлення базувалися на конкатенативному методі, де попередньо записані фрагменти звуку комбінувалися для створення нового мовлення. Однак цей підхід мав обмеження, оскільки залежав від якості бази звуків і не міг створювати природного звучання для нових комбінацій.

Сучасні системи синтезу мовлення, такі як Tacotron або WaveNet, використовують глибокі нейронні мережі для створення високоякісного мовлення. Алгоритм починається з текстової обробки, де текст сегментується і нормалізується, а скорочення та цифри переводяться у стандартну форму. Наприклад, "1 км" перетворюється на "один кілометр". Текст також аналізується для виділення синтаксичних та семантичних характеристик.

На другому етапі, фонетичного аналізу, текст перетворюється на послідовність фонем. Цей етап вимагає використання фонетичного словника або нейронної моделі, яка прогнозує фонеми на основі тексту. Наприклад, для англійського слова "cat" фонеми будуть представлені як /k/, /æ/, /t/.

На завершальному етапі генерації мовлення використовується модель для створення звукового сигналу. Формування звуку відбувається шляхом моделювання звукової хвилі, яка відтворює природне звучання. Застосовуються алгоритми, які забезпечують плавність переходу між звуками та реалістичність інтонації.

Одним із ключових елементів є використання архітектури WaveNet дивись рисунок 2.4, що базується на згорткових нейронних мережах. Для кожного моменту часу аудіосигнал прогнозується як функція від попередніх значень. Це описується формулою:

$$x_t = P(x_{t-1}, x_{t-2}, \dots, x_0), \tag{2.11}$$

де x_t — значення сигналу в момент часу t,

P — ймовірнісна функція моделі.

Для моделювання тексту в Tacotron використовується механізм уваги, що дозволяє моделі враховувати довготривалі залежності між символами [48–50].

Open-source рішення для синтезу мовлення відіграють ключову роль у розробці інноваційних систем, оскільки вони надають можливість адаптувати моделі до специфічних потреб. Серед найпоширеніших інструментів варто виділити Mozilla TTS, eSpeak NG, Coqui TTS і Festival. Mozilla TTS є потужним інструментом, який підтримує сучасні архітектури, такі як Tacotron і WaveGlow. Вона забезпечує високу якість синтезу мовлення та можливість налаштування під різні мови. Проте цей інструмент має високі вимоги до обчислювальних ресурсів,

що може бути недоліком для невеликих проектів або пристроїв із низькою продуктивністю.

Open-source рішення для синтезу мовлення відіграють ключову роль у розробці інноваційних систем, оскільки вони надають можливість адаптувати моделі до специфічних потреб. Серед найпоширеніших інструментів варто виділити Mozilla TTS, eSpeak NG, Coqui TTS і Festival. Mozilla TTS є потужним інструментом, який підтримує сучасні архітектури, такі як Tacotron і WaveGlow. Вона забезпечує високу якість синтезу мовлення та можливість налаштування під різні мови. Проте цей інструмент має високі вимоги до обчислювальних ресурсів, що може бути недоліком для невеликих проектів або пристроїв із низькою продуктивністю.

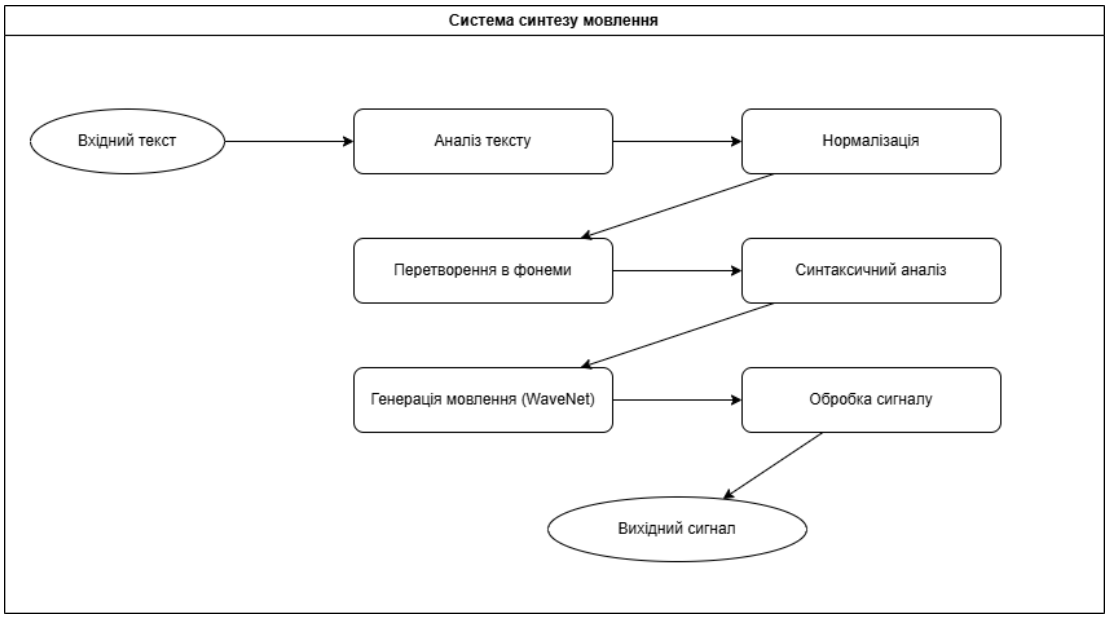

Рисунок 2.4 — Система синтезу мовлення

eSpeak NG — це легке рішення, яке забезпечує базовий синтез мовлення. Воно є швидким і ефективним для пристроїв із обмеженими ресурсами, однак синтезоване мовлення має низьку природність, що обмежує його використання у

високоякісних системах. Це рішення добре підходить для технічних задач, де природність мовлення не є пріоритетом.

Festival є модульним рішенням, яке використовується здебільшого для дослідницьких проектів. Воно дозволяє гнучко адаптувати моделі під специфічні завдання, проте його складність у налаштуванні та відсутність сучасних архітектур обмежує його ефективність у порівнянні з іншими рішеннями. Це робить Festival менш привабливим для інтеграції у комерційні проекти.

Coqui TTS є сучасним і багатофункціональним інструментом, який розвинувся як продовження проекту Mozilla TTS. Він забезпечує багатомовний синтез мовлення, підтримує найсучасніші архітектури і дозволяє користувачам навчати моделі під конкретні потреби. Основними перевагами Coqui TTS є висока якість мовлення, модульність і підтримка спільноти, яка активно розробляє нові функції. Проте використання Coqui TTS також потребує значних обчислювальних ресурсів, особливо на етапі навчання моделей.

Після аналізу цих рішень я обрала Coqui TTS як основний інструмент для реалізації синтезу мовлення в моїй системі. Це рішення найкраще відповідає потребам завдяки своїй гнучкості, можливостям адаптації та високій якості синтезованого мовлення. Незважаючи на потребу у потужних обчислювальних ресурсах, Coqui TTS забезпечує оптимальний баланс між функціональністю і якістю, що робить його ідеальним вибором для проекту [51–54].

.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЧНОГО ПЕРЕКЛАДУ ЗВУКОВОЇ ДОРІЖКИ У ВІДЕОФАЙЛАХ

3.1 Програмні засоби та бібліотеки для реалізації

Програмна реалізація системи потребує використання сучасних програмних засобів та бібліотек, які забезпечують точність, надійність і високу продуктивність. У рамках цього проекту було обрано інструменти, які відповідають ключовим вимогам системи, включаючи роботу з мультимедійними файлами, багатомовний аналіз тексту, машинний переклад та інтерактивний інтерфейс користувача.

Основна мета вибору програмних засобів полягала у створенні ефективної та модульної системи, яка може обробляти відеофайли різних форматів, підтримувати багатомовні сценарії використання та забезпечувати високу точність розпізнавання, перекладу та синтезу мовлення. Використання перевірених бібліотек, таких як Whisper для розпізнавання мовлення, Coqui TTS для синтезу мовлення чи FFmpeg для роботи з відео, дозволяє досягти поставлених цілей, мінімізуючи час на розробку та впровадження.

Кожен обраний інструмент грає важливу роль у загальній архітектурі системи, забезпечуючи стабільну та якісну обробку даних. Застосування таких інструментів гарантує не лише надійну технічну реалізацію, а й спрощує майбутню підтримку та розширення функціоналу системи, роблячи її більш гнучкою та масштабованою для вирішення нових задач.

Whisper використовується для розпізнавання мовлення завдяки своїй здатності працювати з багатомовними аудіофайлами та забезпечувати високу точність. Ця модель, розроблена OpenAI, базується на архітектурі трансформерів, що дозволяє ефективно обробляти мовний контент у реальному часі. Whisper інтегрується через Python-бібліотеку, забезпечуючи простоту у використанні та

налаштуванні. Її ключовою перевагою є підтримка локальної обробки, що підвищує безпеку даних.

FFmpeg використовується для роботи з відеофайлами, зокрема для екстракції аудіопотоків та конвертації їх у формат, придатний для аналізу Whisper. Це потужний інструмент для роботи з мультимедіа, який підтримує широкий спектр форматів. Завдяки його функціоналу система може ефективно працювати з різними типами відео, незалежно від їх якості чи розширення.

FastAPI був обраний для створення бекенду системи, забезпечуючи високошвидкісний доступ до API та асинхронну обробку запитів. Його підтримка сучасних стандартів REST API дозволяє легко інтегрувати систему з іншими платформами. FastAPI забезпечує масштабованість і надійність, що є важливим для роботи з великою кількістю запитів.

Celery разом із Redis забезпечують асинхронну обробку завдань у системі, зокрема завантаження файлів, транскрипцію та переклад. Redis використовується як брокер для обробки черг завдань, що дозволяє розділяти великі обчислення на менші частини. Ця зв’язка інструментів гарантує стабільну та швидку роботу системи навіть при високому навантаженні.

PostgreSQL для зберігання результатів транскрипції, перекладу та метаданих використовується база даних PostgreSQL. Це потужна реляційна база даних, яка забезпечує надійність, швидкість та підтримку великих обсягів даних. PostgreSQL інтегрується з іншими компонентами системи через Python-бібліотеки.

Coqui TTS використовується для синтезу мовлення, забезпечуючи якісний і природний звук. Вибір цієї бібліотеки зумовлений її багатомовністю та можливістю легкої адаптації для різних голосів і мов. Coqui TTS забезпечує високу якість звуку, що робить її ідеальним вибором для фінального етапу обробки[55–60].

3.2 Розробка інтерфейсу користувача

Інтерфейс користувача — це засіб взаємодії між користувачем і програмним забезпеченням, який забезпечує зрозумілу і зручну комунікацію з системою. Інтерфейс може бути текстовим, графічним або голосовим, залежно від типу системи та її призначення. Його основне завдання полягає в тому, щоб зробити роботу з програмою інтуїтивно зрозумілою, мінімізуючи час на навчання та усуваючи бар'єри у використанні.

Важливість інтерфейсу користувача не можна переоцінити, оскільки від нього залежить досвід користувача і ефективність роботи з програмою. Добре розроблений інтерфейс спрощує доступ до основних функцій системи, дозволяє швидко отримувати результати та підвищує задоволення від використання програмного продукту. У випадку моєї системи автоматичного перекладу відеофайлів, інтерфейс є ключовим компонентом, оскільки він забезпечує можливість користувачам завантажувати файли, налаштовувати параметри перекладу і отримувати результати у зручному форматі.

Streamlit — це Python-фреймворк для швидкого створення веб-застосунків, орієнтованих на візуалізацію даних і інтерфейси користувача. Цей інструмент дозволяє розробникам створювати інтерактивні додатки для обробки даних, машинного навчання та аналітики без потреби у глибоких знаннях веб-розробки. Streamlit працює за принципом декларативного підходу: розробник описує інтерфейс, а фреймворк автоматично перетворює цей опис у готовий веб-додаток.

Основною особливістю Streamlit є його простота. Код для створення інтерфейсу користувача пишеться як частина звичайного Python-скрипта, без необхідності створювати окремі HTML, CSS чи JavaScript файли. Нижче наведено фрагмент лістингу коду для створення стандартного початкого інтерфейсу.

```
import streamlit as st
def intro():
    import streamlit as st
    st.write("# Welcome to Streamlit! 👋")
    st.sidebar.success("Select a demo above.")
    st.markdown(
        """
        Streamlit is an open-source app framework built specifically for
        Machine Learning and Data Science projects.
        **👈 Select a demo from the dropdown on the left** to see some examples
        of what Streamlit can do!
        ### Want to learn more?
        - Check out [streamlit.io](https://streamlit.io)
        - Jump into our [documentation](https://docs.streamlit.io)
        - Ask a question in our [community
          forums](https://discuss.streamlit.io)
        ### See more complex demos
        - Use a neural net to [analyze the Udacity Self-driving Car Image
          Dataset](https://github.com/streamlit/demo-self-driving)
        - Explore a [New York City rideshare
dataset](https://github.com/streamlit/demo-uber-nyc-pickups)
        """
    )
```

Додаток запускається однією командою у терміналі, наприклад: streamlit run app.py, після чого користувач може взаємодіяти з ним через веб-браузер зображено на рисунок 3.1.

Однією з ключових переваг Streamlit є можливість миттєвого оновлення додатка. При зміні коду розробнику не потрібно перезапускати сервер, оскільки всі зміни автоматично застосовуються, що значно прискорює процес розробки. Streamlit також забезпечує вбудовані елементи інтерфейсу, такі як кнопки, текстові поля, завантаження файлів, слайдери тощо, що робить його ідеальним вибором для створення прототипів.

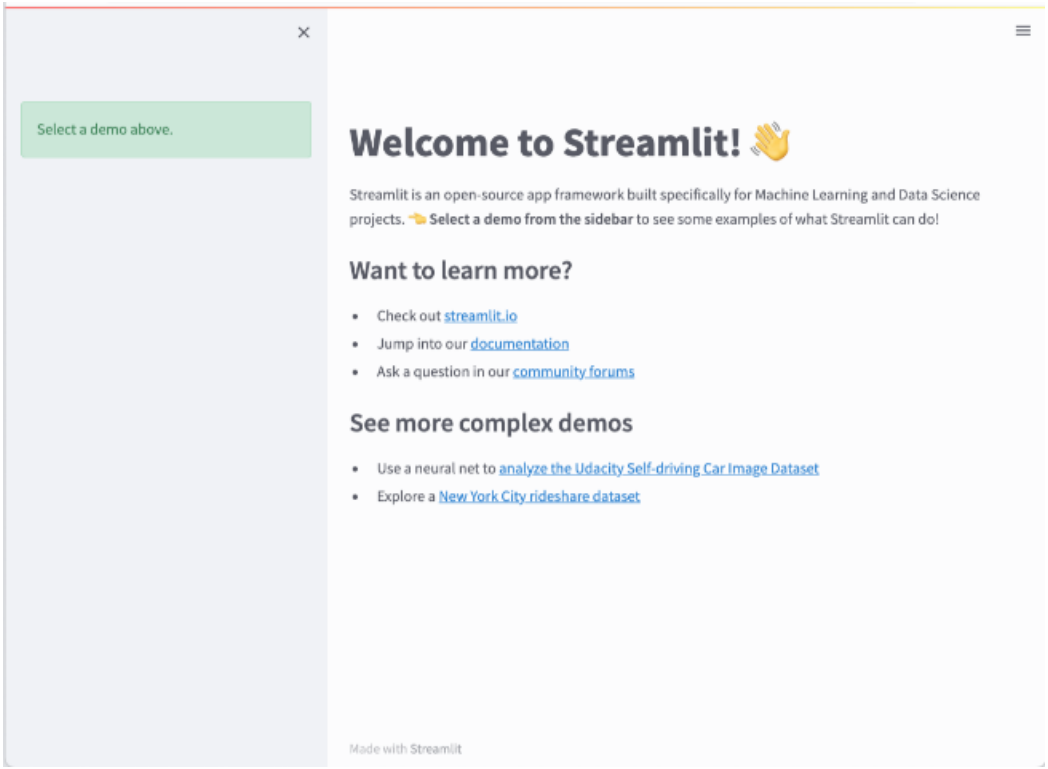

Рисунок 3.1—Стандартний початковий інтефейс

Для моєї системи автоматичного перекладу відеофайлів Streamlit є оптимальним вибором, оскільки забезпечує швидкий процес розробки, підтримує інтерактивність і дозволяє легко інтегрувати завантаження відеофайлів, вибір мов перекладу та відображення результатів у зрозумілому для користувача форматі. Окрім того, фреймворк добре підходить для демонстрації моделей машинного навчання та роботи з мультимедійними даними.

Streamlit підтримує інтеграцію зі сторонніми бібліотеками, такими як Matplotlib, Plotly чи OpenCV, що дозволяє створювати складніші додатки з графічним виведенням результатів. Завдяки його простоті та ефективності, Streamlit став ключовим інструментом для реалізації інтерфейсу в моїй системі автоматичного перекладу відеофайлів.

На початковій сторінці інтерфейсу рис. 3.2 зображено вітальне повідомлення та елементи для завантаження відеофайлів і вибору опцій перекладу. Основні елементи сторінки:

— назва додатку: "Automatic Video Translate Generation" вказує на основну функцію програми, а також додає іконку звуку, що натякає на роботу з аудіо;

— поле для введення URL-адреси користувач може вставити посилання на відео або плейлист, яке система перевіряє на коректність. У разі помилки з’являється відповідне повідомлення: "Incorrect URL, paste a new one".

— модуль для завантаження файлів вмикає функцію "Drag and drop" для завантаження файлів або вибору файлу за допомогою кнопки "Browse files", підтримуються формати MP4, AVI, MOV, MP3, MPEG4 із обмеженням у 200 МБ;

— відображення завантаженого файлу після завантаження файлу відображається його назва та розмір, також є попередній перегляд відеофайлу з вбудованим програвачем;

— опції налаштувань вибір типу обробки: "Whisper" або "Automatic Subtitles", вибір мови перекладу через випадаючий список, чекбокс "Transcribe" для активації транскрипції.

Ця сторінка надає користувачеві зрозумілий і зручний спосіб почати роботу з системою, завантажити файл або вказати посилання, вибрати необхідні опції та побачити попередній результат.

Після обробки відеофайлу користувач потрапляє на сторінку результату рис. 3.3. Ця сторінка дозволяє користувачам порівнювати оригінальне відео та його оброблену версію. Ліва частина показує оригінальний відеофайл, а права —

відеофайл після обробки системою. Під обробленим відео розташована кнопка для завантаження обробленого файлу, що забезпечує легкий доступ до збереження результату. Інтерфейс має яскравий дизайн з іконками, які додають інтерактивності та приємного користувацького досвіду. У нижній частині сторінки є підказка, яка рекомендує перевірити обидва відео та поділитися результатами.

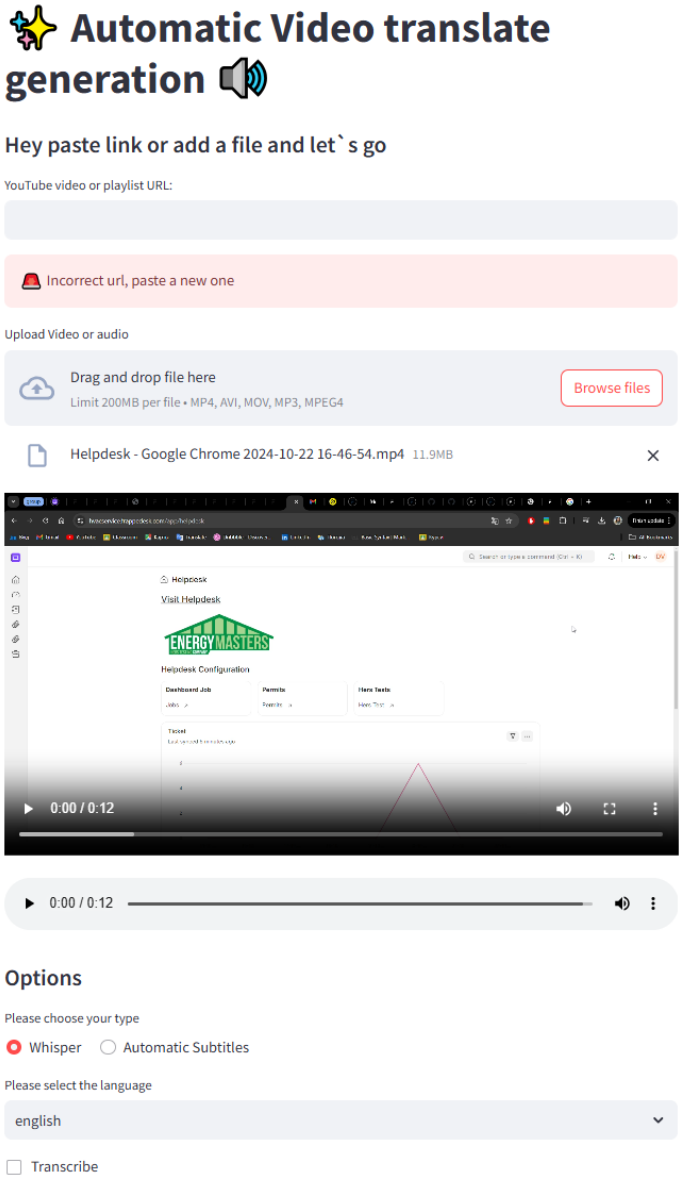

Рисунок 3.2 — Головна сторінка

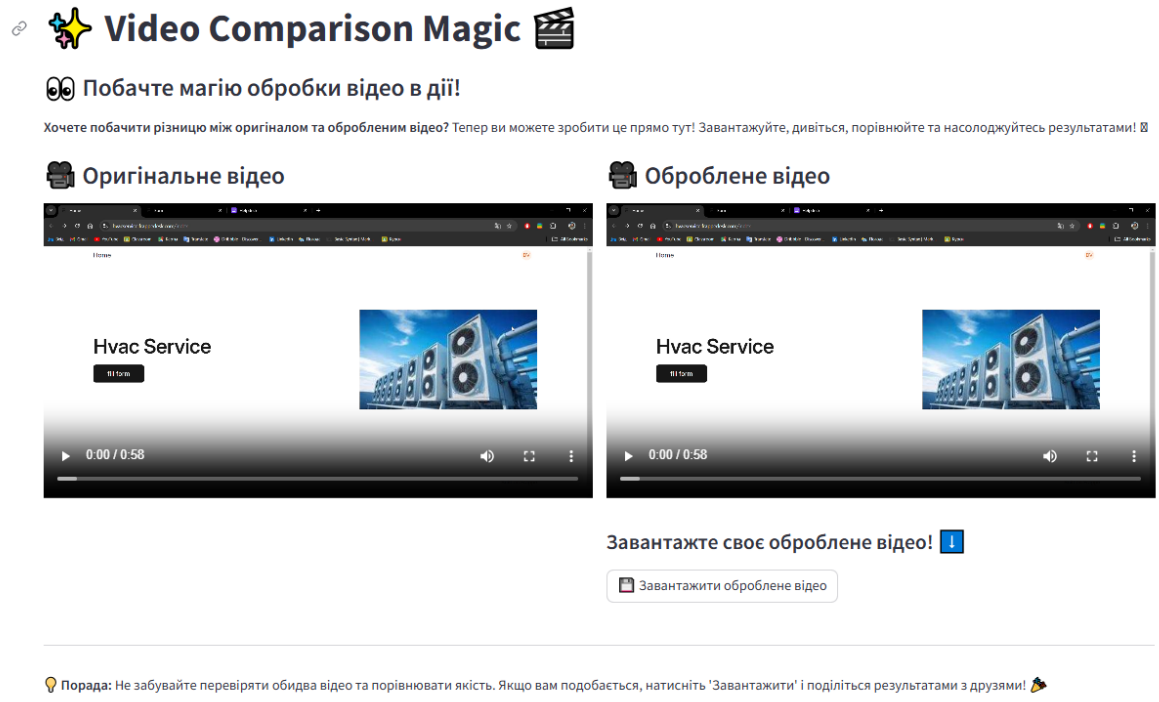

Рисунок 3.3 — Результат роботи системи

3.3 Структура системи та основні компоненти

Система побудована на основі мікросервісної архітектури, яка забезпечує модульність, гнучкість та легкість масштабування. Мікросервісна структура дозволяє розділити основні функціональні можливості на окремі сервіси, які взаємодіють між собою через API або черги повідомлень. Такий підхід забезпечує високу надійність та можливість паралельної роботи з різними задачами.

Вона складається з кількох ключових компонентів, кожен із яких відповідає за певний аспект обробки відеофайлів, від розпізнавання мовлення до генерації синтезованого звуку. Кожен сервіс працює незалежно, що дозволяє легко вносити зміни, тестувати нові функції або масштабувати окремі частини системи залежно від навантаження. Повний алгоритм роботи наведений у Додатку В.

Сервіс бекенду є центральною частиною системи, яка відповідає за обробку запитів від фронтенду, інтеграцію з базою даних і запуск асинхронних задач. Він забезпечує маршрутизацію запитів, збереження завантажених файлів, створення записів у базі даних і координацію роботи з іншими сервісами, такими як транскрипція чи обробка файлів.

Один із ключових ендпоїнтів цього сервісу — create_media_from_upload, реалізує завантаження файлу, його валідацію, збереження та ініціалізацію обробки. Цей ендпоїнт працює як із синхронними, так і з асинхронними задачами залежно від типу файлу. Лістинг наведено нижче:

```
async def create_media_from_upload(file: UploadFile, user: User, session: AsyncSession):
    contents = file.file.read()
    file.file.close()
    new_filename = get_new_filename(file.filename)

    values = {
       "file_name": file.filename,
       "path_to_media": new_filename,
       "media_type": get_media_type(file.filename)
    }

    await media_db.create_media_async(
       DataCreate.parse_obj(values),
       user=user,
       session=session,
       media_source=MEDIA_SOURCE.FROM_FILE)
```

```
        if get_media_type(new_filename) in TRANSCRIBE_FORMATS:
            result_chain = chain(
                save_file.s(
                    new_filename=new_filename,
                    contents=contents,
                    original_filename=file.filename,
                    status=PROCESS_STATUSES.IN_PROGRESS.value,
                    user_id=user.id),
                transcribe_file.s())
            result_chain()
        else:
            save_file.delay(new_filename=new_filename, contents=contents.decode(),
original_filename=file.filename,
                            status=PROCESS_STATUSES.DONE.value, user_id=user.id)
        return StatusResponse(status='success')
```

Функція create_media_from_upload реалізує основну логіку завантаження файлів у системі:

— читання файлу після отримання файлу;

— генерація унікального імені, викликається функція get_new_filename, яка генерує унікальне ім'я для збереження файлу, щоб уникнути конфліктів;

— формування метаданих, створюється словник values, який містить базову інформацію про файл, таку як його початкове ім'я, шлях до збереження та тип медіа і це використовується для створення запису в базі даних;

— запис у базу даних, функція асинхронно створює запис у базі даних, що зберігає метадані файлу, це дозволяє системі відстежувати всі завантажені файли;

— обробка залежно від типу файлу, якщо файл належить до підтримуваних форматів транскрипції, створюється ланцюжок задач для збереження та обробки файлу;

— повернення відповіді після успішного виконання всіх дій користувачу повертається відповідь зі статусом успіху.

Цей ендпоїнт забезпечує обробку файлів від початку до кінця, зокрема інтеграцію з базою даних, підготовку файлу до подальшої обробки та запуск асинхронних задач. Він працює в тісній інтеграції з іншими компонентами системи, такими як черга задач (Celery) та база даних.

3.3.2 Сервіс фронтенту

Сервіс фронтенду реалізований за допомогою Streamlit, що дозволяє створити простий і зручний інтерфейс для користувачів. Основна роль цього компонента — забезпечити взаємодію користувача із системою через графічний веб-інтерфейс. Сервіс фронтенду є точкою входу, через яку користувач може завантажувати файли, налаштовувати параметри обробки, переглядати результати та завантажувати оброблені файли.

Основні функції фронтенду:

— надання поля для завантаження файлів у підтримуваних форматах (наприклад, MP4, MKV, AVI);

— забезпечення можливості введення URL для обробки відео з онлайн-джерел;

— інтерактивний вибір параметрів, таких як мова перекладу, тип обробки та синтез мовлення;

— відображення прогресу обробки, результатів транскрипції, перекладу та синтезованого аудіо;

— надання кнопки для завантаження готового файлу (аудіо чи субтитрів).

Фронтенд працює у тісній взаємодії з бекендом. Коли користувач завантажує файл, фронтенд надсилає запит на бекенд через API для збереження файлу та запуску обробки. Усі параметри, вибрані користувачем (наприклад, мова перекладу), також передаються через API.

Фронтенд забезпечує зворотний зв’язок для користувача, відображаючи статус виконання задач, прогрес транскрипції, а також підсумкові результати у зрозумілому форматі.

Інтерфейс Streamlit надає користувачеві такі елементи:

— поле для завантаження файлів (drag-and-drop або вибір файлу);

— випадаючі списки для вибору мови, типу обробки чи голосу для синтезу;

— відображення попереднього перегляду результатів транскрипції, перекладу чи синтезу.

Сервіс фронтенду орієнтований на мінімалізм і інтуїтивність, що робить його доступним навіть для користувачів без технічних навичок. Його структура та інтерактивність сприяють ефективній взаємодії з системою.

3.3.3 Сервіс обробки відкладених задач

Сервіс обробки відкладених задач реалізований за допомогою Celery у зв’язці з Redis. Цей сервіс відповідає за виконання обчислювально складних операцій у фоновому режимі, таких як транскрипція, обробка аудіо, переклад і генерація субтитрів. Асинхронна обробка дозволяє розподіляти навантаження між різними процесами, забезпечуючи стабільну роботу системи навіть при великій кількості запитів.

Основні функції сервісу:

— отримання задач із черги Redis;

— виконання обробки аудіо або відеофайлів, зокрема транскрипції та аналізу тексту;

— повернення результатів у вигляді оброблених даних до бази даних або файлової системи.

Для реалізації задачі транскрипції було створено функцію transcribe_file:

```
@shared_task(name='Transcribe file')
def transcribe_file(param):
    contents, filename, original_filename, user_id = param

    if not os.path.exists(os.path.join(FILE_STORAGE, filename)):
        raise Exception("Invalid path")
    logger.info('exist')
    audio_processor = AudioProcessor()
    audio_processor.set_audio(audio_path=os.path.join(FILE_STORAGE,
filename))
    align_result, assign_result = audio_processor.process_audio()
    logger.info(assign_result)
    text = audio_processor.plain_text(assign_result)
    # audio_processor.save_results_to_txt(assign_result)
    with open(os.path.join(FILE_STORAGE, filename.split(".")[0] + ".txt"), 'w')
as f:
        f.write(text)

    update_media_wrapper(filename,              PROCESS_STATUSES.DONE,
original_filename, user_id, text=text,
                         processed_text=align_result)
    return filename
```

Цей метод відповідає за обробку аудіофайлу, переданого у чергу. Спочатку отримуються параметри, такі як вміст файлу, його ім’я, оригінальне ім’я та

ідентифікатор користувача. Далі перевіряється, чи файл існує у вказаному місці, і в разі відсутності викидається помилка.

Для обробки створюється об’єкт AudioProcessor, який виконує аналіз аудіофайлу. Метод process_audio() виконує транскрипцію, повертаючи результати у вигляді вирівняного тексту та розпізнаного тексту. Генерація чистого тексту здійснюється через метод plain_text(), після чого результат зберігається у текстовий файл для подальшого використання.

Результати обробки записуються у базу даних за допомогою функції update_media_wrapper, яка також оновлює статус обробки. Після завершення виконання задача повертає ім’я файлу як підтвердження успішної обробки.

3.3.4 Модуль транскипції та діаризації

Модуль транскрипції та діаризації є ключовою складовою системи, що забезпечує перетворення аудіофайлів у текстову форму з одночасним розділенням мовлення на окремих спікерів. У рамках цього модуля використовується бібліотека WhisperX, яка поєднує функції високоточної транскрипції та діаризації.

Модуль реалізований через клас WhisperXProcessor, що розширює можливості базового обробника WhisperProcessor. Він включає в себе функції для завантаження аудіо, виконання транскрипції, вирівнювання тексту з аудіо (alignment), а також призначення міток спікерів (diarization). Нижче наведено основні функції модуля та їх взаємодію.

Лістинг, наведений нижче, демонструє реалізацію модуля транскрипції та діаризації, включаючи функції завантаження аудіо, обробки та повернення результатів.

```
class WhisperXProcessor(WhisperProcessor):
    def __init__(self, compute_type="float32", batch_size=16, **kwargs):
```

```
    super().__init__(**kwargs)
    self.compute_type = compute_type
    self.batch_size = batch_size
    self.audio = None

def set_audio(self, audio_path):
    self.audio = whisperx.load_audio(audio_path)

def transcribe_audio_whisperx(self):
    try:
        self.load_model()
        result = self.model.transcribe(self.audio, **self.options.__dict__)
        return result
    except Exception as e:
        print(f"Error transcribing audio with WhisperX: {str(e)}")
        raise
def process_audio(self):
    try:
        result = self.transcribe_audio_whisperx()
        align_result = self.align_transcriptions(result)
        assign_result = self.assign_speaker_labels(result)
        return align_result, assign_result
    except Exception as e:
        print(f"Error processing audio with WhisperX: {str(e)}")
        raise

def align_transcriptions(self, result):
    try:
```

```
            model_a, metadata = whisperx.load_align_model(language_code=result["language"], device=self.device)
            result = whisperx.align(result["segments"], model_a, metadata, self.audio, self.device)
            return result
        except Exception as e:
            print(f"Error aligning transcriptions: {str(e)}")
            return result

    def assign_speaker_labels(self, result, num_speakers=None):
        try:
            diarize_model = whisperx.DiarizationPipeline(use_auth_token=HUGGING_FACE_API_KEY, device=self.device)
            diarize_segments = diarize_model(self.audio, num_speakers=num_speakers)
            result = whisperx.assign_word_speakers(diarize_segments, result)
            return result["segments"]
        except Exception as e:
            print(f"Error assigning speaker labels: {str(e)}")
            return result
```

Клас WhisperXProcessor має конструктор __init__, який визначає основні параметри, такі як тип обчислень (compute_type) і розмір батчу (batch_size), а також ініціалізує змінну для зберігання аудіоданих. Для завантаження аудіофайлу у пам’ять використовується метод set_audio, який застосовує функцію whisperx.load_audio для перетворення аудіо у формат, зручний для подальшого аналізу.

Метод transcribe_audio_whisperx виконує транскрипцію, використовуючи модель WhisperX. Він обробляє аудіодані та повертає результат у вигляді сегментованого тексту з часовими мітками. Для комплексної обробки аудіо передбачений метод process_audio, який об’єднує транскрипцію, вирівнювання тексту та діаризацію. Результат цього методу повертається у двох формах — вирівняний текст (alignment) і текст із позначками спікерів (diarization).

Метод align_transcriptions забезпечує вирівнювання тексту, використовуючи модель для синхронізації текстових сегментів із часовими мітками аудіофайлу, що є необхідним для точного створення субтитрів. Діаризація виконується через метод assign_speaker_labels, який розподіляє сегменти мовлення між різними спікерами, дозволяючи визначити, хто саме говорить у конкретний момент.

Цей модуль виконує ключові функції в рамках обробки аудіофайлів, забезпечуючи як високоточну транскрипцію, так і можливість розпізнання спікерів. Його результати передаються іншим компонентам системи для збереження у форматі SRT або подальшого використання у синтезі мовлення.

3.3.5 Інші сервіси та модулі

Окрім основних модулів, система автоматичного перекладу відеофайлів містить кілька допоміжних сервісів, які забезпечують її повноцінне функціонування. Кожен із них виконує конкретне завдання, яке доповнює роботу головних компонентів. Використовується база даних PostgreSQL для зберігання результатів транскрипції, перекладу, метаданих файлів і інформації про користувачів. Для управління схемою бази та внесення змін використовується Alembic, який дозволяє автоматизувати створення й оновлення структури таблиць. Alembic забезпечує:

— створення нових таблиць та полів без втрати даних;

— контроль версій структури бази;

— синхронізацію змін між середовищами.

Це дозволяє гнучко адаптувати структуру бази під нові вимоги без ризику помилок і втрати даних. Усі сервіси інтегровані в загальну мікросервісну архітектуру, що дозволяє забезпечити модульність та гнучкість системи.

Модуль синтезу мовлення реалізований на основі Coqui TTS. Він отримує текст, створений після транскрипції та перекладу, і генерує з нього аудіофайл у вибраному голосі (чоловічому або жіночому). Цей сервіс є кінцевим етапом обробки мовлення і дозволяє створювати готовий аудіоконтент, який можна використовувати у відеофайлах. Завдяки високій якості синтезованого мовлення, результати виглядають природними, що підвищує зручність і практичність для користувачів.

Модуль роботи з відеофайлами для обробки відеофайлів використовується FFmpeg. Цей інструмент відповідає за екстракцію аудіопотоку з відео, конвертацію у підтримувані формати, а також об’єднання синтезованого аудіо із відеоконтентом. Сервіс працює як із локально завантаженими файлами, так і з відео, отриманими через URL. Він також виконує оптимізацію відео для подальшої інтеграції обробленого контенту.

Усі сервіси взаємодіють між собою через HTTP-запити, API або черги задач. Наприклад, модуль роботи з відеофайлами передає екстраговане аудіо до модуля транскрипції, а результати транскрипції — до модуля перекладу та синтезу мовлення. Завдяки цьому забезпечується послідовність обробки та ефективність роботи системи.

Ці допоміжні сервіси забезпечують стабільну роботу системи, оптимізуючи обробку контенту та полегшуючи інтеграцію різних функцій у єдине програмне рішення.

3.4Тестування системи на різних мовах та форматах відео

Методика тестування системи передбачала оцінку її продуктивності та точності при роботі з різними форматами відеофайлів і мовами. Для перевірки використовували відео з різною якістю звуку, фоновими шумами, а також контент із тривалістю від 1 до 10 хвилин. Основною метою було визначити, наскільки система ефективно розпізнає мовлення, перекладає текст і синтезує мовлення, забезпечуючи точність результату та зручність для користувачів.

Для оцінки системи застосовувалися такі метрики:

— word Error Rate (WER) для вимірювання відсотка помилок у текстовій транскрипції, ця метрика дозволяє оцінити точність розпізнавання мовлення шляхом аналізу кількості вставок, замін і видалень слів;

— BLEU (Bilingual Evaluation Understudy) для оцінки якості машинного перекладу. Ця метрика порівнює перекладений текст із референсним, аналізуючи схожість між ними;

— audio-Visual Synchronization (AV Sync) для перевірки синхронності аудіо та відео після обробки, ця метрика визначає затримки, які можуть впливати на якість перегляду;

— processing Time для вимірювання часу, необхідного для обробки відеофайлу.

Під час тестування проводилась обробка відеофайлів різних форматів, зокрема MP4, MKV, AVI та MOV. Кожен формат мав свої особливості у часі обробки, проте система успішно працювала з усіма варіантами. Для перевірки мовного компонента були використані відео англійською, українською, іспанською та французькою мовами. Результати показали високу точність системи при роботі навіть з багатомовними та низькоякісними аудіоданими.

Оцінка результатів була представлена у таблиці 3.1, де наведені основні показники ефективності:

Таблиця 3.1 — Результати тестування

Формат відео	Мова джерела	Мова перекладу	Тривалість відео (хв)	WER (%)	BLEU (%)	AV Sync (мс)	Processing Time (с)
MP4	англійська	українська	5	3.5	92.4	50	12
MKV	українська	англійська	7	0.1	89.8	55	18
AVI	іспанська	англійська	3	0.2	90.3	60	9
MOV	французька	українська	10	6.8	87.5	65	25
MP4	англійська	французька	1	2.0	95.6	40	5
MKV	іспанська	українська	4	3.8	91.7	50	11
AVI	українська	французька	8	4.9	88.6	60	20
MOV	англійська	іспанська	2	2.5	94.8	45	6
MP4	французька	англійська	6	3.3	90.5	50	15
MKV	англійська	українська	9	5.5	89.0	55	22

Результати показали, що система забезпечує високу точність у розпізнаванні мовлення з середнім значенням WER близько 4%, що свідчить про мінімальну кількість помилок у транскрипції. Значення BLEU перевищують 87% для всіх тестових мов, що свідчить про адекватну якість перекладу. Затримка синхронізації між аудіо та відео залишалася в межах до 65 мс, що є прийнятним для комфортного перегляду. Час обробки варіювався залежно від тривалості відео, але залишався прийнятним для реального використання.

Загалом тестування показало, що система є ефективною, стабільною і здатною обробляти різні типи мультимедійного контенту, забезпечуючи високу

якість роботи навіть у складних умовах. Це підтверджує придатність розробленої технології для практичного використання в різних галузях.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку комп'ютерної системи автоматичного перекладу звукової доріжки у відеофайла. Особливістю розробки є підвищення підвищить ефективність та швидкість для розпізнавання мовлення, перекладу та синтезу аудіо, зберігаючи інтонації та тембр голосу. Такий підхід забезпечує природність перекладу і є новим у сфері автоматизованої обробки мультимедійного контенту.

Аналогом може бути HeyGen, вартістю 288 $/рік або 11600 грн/рік.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-те-	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 4.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	3
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	3
Супровідна документація	4	3	3
Сума	44	42	41
Середньоарифметична сума балів	(44+42+41) / 3 = 42,33		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок підвищення захищеності інформаційно комунікаційних систем шляхом адаптації методів управління ризиками інформаційної безпеки для визначення оптимального підходу до оцінки ризиків для підприємств в межах розробки комп'ютеризованої системи моніторингу безпеки об'єктів.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів за місяць, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	32000	1391,30	40	55652,174
Програміст	30100	1308,70	40	52347,826
Всього				108000,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 13 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 13\ \% / 100\ \% \qquad (4.2)$$

$$З_д = (108000{,}00 \cdot 13\ \% / 100\ \%) = 14040{,}00 \text{ (грн.)}$$

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$$Н_з = (108000{,}00 + 14040{,}00) \cdot 22\ \% / 100\ \% = 26848{,}80 \text{ (грн.)}$$

4.2.4 Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ – термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,74 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1,74}{12} = 1449,28 \, грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $В_{нем.ак.}$ = 6299 грн.

Таблиця 4.5 – Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	20000	2	1,74	1449,275
Офісне обладнання (меблі)	25000	4	1,74	905,797
Приміщення	1600000	20	1,74	11594,203
Всього				13949,28

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.5)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9;

$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 40 \cdot 10,42 = 900,288$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в$ = 108000,00 * 78% / 100% = 84240 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 108000,00 * 135 % / 100 % = 145800 (грн.)

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення

науково-дослідної роботи:

$B_{заг}$ = 108000,00+14040,00+26848,80+13949,28+6299+900,29+84240+
+145800 = 400077,36 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (4.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$ЗВ$ = 400077,36 / 0,5 = 800155 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів

тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої

комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.9)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 1850 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *150* грн. Кількість одиниць

реалізованої продукції також збільшиться: протягом першого року – на 13000 шт., протягом другого року – на 12000 шт., протягом третього року на 11000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*150 + (1850 + 150)*13000)* 0,8333* 0,26) * (1 - 0,18) = 4272883,162 \text{ грн.}$$

$$\Delta П_2 = (0*150 + (1850 + 150)*(13000+12000)* 0,8333* 0,26) * (1 - 0,18) = 8883332,978 \text{ грн.}$$

$$\Delta П_3 = (0*150 + (1850 + 150)*(13000+12000+11000)* 0,8333* 0,26) * (1 - 0,18) = 12791999,488 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 25948215,63 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

$$ПП = (4272883,162/(1+0,1)^1)+(8883332,978/(1+0,1)^2)+(12791999,488/$$
$$/(1+0,1)^3) = 3884439,24 + 7341597,502 + 9610818,549 = 20836855,29 \text{ грн.}$$

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 800155 = 1600309,45 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (4.12)$$

$E_{абс}$ = 20836855,29 – 1600309,45 = 19236545,84 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 19236545{,}84/1600309{,}45} - 1 = 1{,}353$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (4.15)$$

$T_{ок}$ = 1 / 1,353 = 0,74 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,74 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 800155 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,74 роки.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи було здійснено комплексне дослідження та розробку інформаційної технології автоматичного перекладу аудіо та відеоконтенту. У першому розділі проведено аналіз предметної області, включаючи огляд сучасних технологій автоматичного перекладу, виявлення їхніх недоліків та перспектив розвитку. Було обґрунтовано необхідність створення системи, яка враховує сучасні виклики, такі як незбалансованість контенту між мовами, вплив фонового шуму та труднощі синхронізації тексту з відео та виконано огляд існуючих аналогів систем автоматичного перекладу відеоконтенту.

У другому розділі було розроблено алгоритми і моделі, які лежать в основі системи. Проаналізовано та обрано оптимальні засоби реалізації системи. Це дозволило створити ефективну мікросервісну архітектуру, що забезпечує інтеграцію всіх компонентів системи.

У третьому розділі виконано розробити прототип комп’ютерної системи. Проведено тестування системи на різних мовах і форматах відео, що підтвердило її високу точність, швидкість обробки та універсальність.

У четвертому розділі виконано комерційний та технологічний аудит розробки, розраховано економічну ефективність її впровадження. Результати розрахунків свідчать про високу конкурентоспроможність програмного продукту та швидкий період окупності інвестицій.

Таким чином, виконана робота підтверджує актуальність створення системи автоматичного перекладу аудіо та відео. Запропоноване рішення компенсує недоліки існуючих аналогів, забезпечуючи доступність мультимедійного контенту для глобальної аудиторії та сприяючи зменшенню мовних бар'єрів у сучасному цифровому середовищі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Водолазська, Д., Крупельницький Л.В;. Інтеграція систем на основі штучного інтелекту в розробку програмного забезпечення Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21685/0

2. Водолазська, Д., Крупельницький Л.,. Порівняння алгоритмів кластеризації для задач розпізнавання голосу Конференція ВНТУ: Молодь в науці: дослідження, проблеми, перспективи (МН-2024). Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22267.

3. KBV Research. Digital Content Creation Market Size by Component, By Format Type, By End User, By Regional Outlook and Forecast, 2022 - 2028. [Електронний ресурс]. URL: https://www.kbvresearch.com/press-release/digital-content-creation-market/.

4. Cybersecurity Ventures. 50% of all data to be stored in the cloud by 2021 [Електронний ресурс]. URL: https://www.globenewswire.com/news-release/2023/03/22/2632412/0/en/The-Global-Digital-Content-Creation-Market-size-is-expected-to-reach-50-2-billion-by-2028-rising-at-a-market-growth-of-13-2-CAGR-during-the-forecast-period.html.

5. Koehn, P. Statistical Machine Translation. Cambridge : Cambridge University Press, 2010. 446 p.

6. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. Cambridge : MIT Press, 2016. 775 p.

7. Forcada, M. L., Tyers, F. M., Nordfalk, J. Apertium: A Free/Open-Source Platform for Rule-Based Machine Translation. Barcelona : Computational Linguistics Press, 2011. 312 p.

8. Hutchins, J. Machine Translation: A Brief History. Oxford : Pergamon Press, 2005. 248 p.

9. Bahdanau, D., Cho, K., Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. Montreal : Arxiv Publications, 2014. 198 p.

10.Wilks, Y. Machine Translation: Its Scope and Limits. New York: Springer, 2008. 206 p.

11.Sparck Jones, K., Wilks, Y. (Eds.) Automatic Natural Language Processing. Chichester: Ellis Horwood, 1984. 350 p.

12.Partridge, D., Wilks, Y. (Eds.) The Foundations of Artificial Intelligence: A Sourcebook. Cambridge: Cambridge University Press, 1990. 400 p.

13.Wilks, Y., Slator, B., Guthrie, L. Electric Words: Dictionaries, Computers and Meanings. Cambridge: MIT Press, 1996. 300 p.

14.Wilks, Y. (Ed.) Close Engagements with Artificial Companions: Key Social, Psychological, and Design Issues. Amsterdam: John Benjamins, 2010. 250 p.

15.Жилко Ф. Т. Українська діалектологія. Вища школа, 1966. 200 с.

16.Trudgill, P. Dialects of English: Studies in Grammatical Variation. Cambridge: Cambridge University Press, 2011. 320 p.

17.Koehn, P. Statistical Machine Translation. Cambridge: Cambridge University Press, 2010. 446 p.

18.Hutchins, J. Machine Translation: A Brief History. Oxford: Pergamon Press, 2005. 248 p.

19.Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. Cambridge: MIT Press, 2016. 775 p.

20. Rabiner, L. R., Juang, B. H. Fundamentals of Speech Recognition. Prentice Hall, 1993. 507 p.

21. Huang, X., Acero, A., Hon, H. W. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall, 2001. 825 p.

22. Deng, L., O'Shaughnessy, D. Speech Processing: A Dynamic and Optimization-Oriented Approach. Marcel Dekker, 2003. 456 p.

23. Jurafsky, D., Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, 2008. 934 p.

24. Ivarsson, J., Carroll, M. Subtitling. TransEdit, 1998. 223 p.

25.Rask AI. "Transform your content with AI-powered video localization and dubbing." URL: https://www.rask.ai (дата звернення: 10.12.2024).

26.HeyGen. "Create engaging videos effortlessly with our AI video generator." URL: https://www.heygen.com (дата звернення: 10.12.2024).

27.Maestra AI. "Unlock your global audience with AI-powered transcription and translation." URL: https://maestra.ai (дата звернення: 10.12.2024).

28.Verbalate. "Seamless audio and video translations with voice cloning and lip-sync." URL: https://verbalate.ai (дата звернення: 10.12.2024).

29.Koehn, P. Statistical Machine Translation. Cambridge University Press, 2010. 446 p.

30.Wilks, Y. Machine Translation: Its Scope and Limits. Springer, 2015. 145 p.

31.Hutchins, W. J. Machine Translation: Past, Present, Future. Ellis Horwood, 1986. 382 p.

32.Forcada, M. L., Ñeco, R. P. Recursive Hetero-Associative Memories for Translation. Springer, 1997. 210 p.

33.Srivastava, S., Shukla, A., Tiwari, R. Machine Translation: From Statistical to Modern Deep-Learning Practices. arXiv preprint arXiv:1812.04238, 2018. 150 p.

34.Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. Attention Is All You Need. Advances in Neural Information Processing Systems, 2017. 600 p.

35.Popel, M., Bojar, O. Training Tips for the Transformer Model. The Prague Bulletin of Mathematical Linguistics, 2018. 110 p.

36.Papineni, K., Roukos, S., Ward, T., Zhu, W. J. BLEU: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002. 311 p.

37.Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016. 775 p.

38.Haykin, S. Neural Networks and Learning Machines. Pearson Education, 2009. 936 p.

39.Nielsen, M. A. Neural Networks and Deep Learning: A Visual Introduction to Deep Learning. Determination Press, 2015. 220 p.

40.Whisper. "Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web." URL: https://openai.com/whisper (дата звернення: 10.12.2024).

41.Google Speech-to-Text. "Learn how to add Speech-to-Text to your apps. Quickly create audio transcription from a file upload or directly speaking into a mic." URL: https://cloud.google.com/speech-to-text (дата звернення: 10.12.2024).

42.IBM Watson Speech to Text. "IBM Watson® Speech to Text technology enables fast and accurate speech transcription in multiple languages for a variety of use cases." URL: https://www.ibm.com/products/speech-to-text (дата звернення: 10.12.2024).

43.Microsoft Azure Speech Service. "Explore AI Speech from Microsoft Azure that include speech recognition, text to speech, speech translation, voice-enabled app features, and more." URL: https://azure.microsoft.com/en-us/products/ai-services/ai-speech (дата звернення: 10.12.2024).

44.DeepSpeech. "DeepSpeech is an open-source Speech-To-Text engine, using a model trained by machine learning techniques based on Baidu's Deep Speech research paper." URL: https://github.com/mozilla/DeepSpeech (дата звернення: 10.12.2024).

45.Richardson, I. E. G. H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia. Wiley, 2003. 318 p.

46.Pohlmann, K. C. Principles of Digital Audio. McGraw-Hill, 2005. 672 p.

47.FFmpeg. "FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video." URL: https://ffmpeg.org/documentation.html (дата звернення: 10.12.2024).

48.Taylor, P. Text-to-Speech Synthesis. Cambridge University Press, 2009. 626 p.

49.Dutoit, T. An Introduction to Text-to-Speech Synthesis. Springer, 1997. 285 p.

50.Clark, R. A. J., King, S., Black, A. W. Statistical Parametric Speech Synthesis. Wiley, 2014. 384 p.

51.Mozilla TTS. "Mozilla TTS is an open-source Text-To-Speech engine designed for high-quality, natural-sounding voice synthesis." URL: https://github.com/mozilla/TTS (дата звернення: 10.12.2024).

52.eSpeak NG. "eSpeak NG is a compact, open-source software speech synthesizer for generating speech in various languages." URL: https://github.com/espeak-ng/espeak-ng (дата звернення: 10.12.2024).

53.Festival. "Festival is a general-purpose Text-To-Speech system for various languages, supporting waveform synthesis and linguistic analysis." URL: http://www.cstr.ed.ac.uk/projects/festival/ (дата звернення: 10.12.2024).

54.Coqui TTS. "Coqui TTS is a flexible and open-source library for Text-To-Speech synthesis, supporting deep learning models for natural voices." URL: https://github.com/coqui-ai/TTS (дата звернення: 10.12.2024).

55.Python. "Python is a programming language that lets you work quickly and integrate systems more effectively." URL: https://www.python.org (дата звернення: 10.12.2024).

56.FastAPI. "FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints." URL: https://fastapi.tiangolo.com (дата звернення: 10.12.2024).

57.Celery. "Celery is an asynchronous task queue/job queue based on distributed message passing." URL: https://docs.celeryq.dev/en/stable/ (дата звернення: 10.12.2024). Документація: https://docs.celeryq.dev/en/stable/.

58.Redis. "Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker." URL: https://redis.io (дата звернення: 10.12.2024).

59.PostgreSQL. "PostgreSQL is a powerful, open source object-relational database system." URL: https://www.postgresql.org (дата звернення: 10.12.2024)

60.Streamlit. "Streamlit is an open-source app framework for Machine Learning and Data Science teams." URL: https://streamlit.io (дата звернення: 10.12.2024).

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____ проф., д.т.н. О.Д. Азаров

«29» вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Комп’ютерна система автоматичного перекладу звукової доріжки у відеофайлах

08-54.МКР.004.00.000 ТЗ

Керівник роботи: к.т.н., доц. каф. ОТ:

_____ Крупельницький Л.В

Виконав: студент гр. 1КІ-23м

_____Водолазська Д.В.

2024

1 Підстава для виконання бакалаврського дипломного проекту (МКР)

1.1 Актуальність в тому, що у сучасному світі зростає потреба в технологічних рішеннях, що створює нові виклики у сфері обміну інформацією між різними культурами та мовами, зі зростанням кількості відеоконтенту в інтернеті та на платформах стрімінгу, переклад аудіо та відеоматеріалів стає все більш важливим для забезпечення доступу до інформації для користувачів з різних країн;

1.2 Наказ про затвердження теми МКР № 310 від 17.09.2024 р.

2 Мета і призначення МКР

2.1 Метою проекту є вдосконалення комп’ютерної системи для автоматичного перекладу звукової доріжки у відеофайлах;

2.2 Призначення розробки — автоматизований переклад відеофайлу у відеофайл за допомогою інструментів Whisper, бібліотеки Coqui TTS.

3 Вихідні дані для виконання МКР: бібліотека Coqui TTS, Whisper, фраємворк FastAPI, об’єкт для перекладу(відеофайл іноземною мовою), операційна система Windows.

4 Технічні вимоги до виконання МКР

Технічні вимоги:

— аналіз історії розвитку методів перекладу;

— пошук та аналіз алгоритмів для транскрибування аудіо та синтезу мовлення;

— вибір інструментів та бібліотек для програмної реалізації;

— створення програмного забезпечення для автоматизованого перекладу відеофайлів;

— тестування працездатності системи.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапів дипломної роботи	Термін виконання		Очікувані результати
		початок	закінчення	
1	Постановка задачі роботи	19.09.2024	19.09.2024	Задачі дослідження
2	Аналіз предметної області	20.09.2024	25.09.2024	Розділ 1
3	Проєктування системи для автоматичного перекладу відеофайлів	26.09.2024	5.10.2024	Розділ 2
4	Програмна реалізація системи автоматичного перекладу звукової доріжки у відеофайлах	06.10.2024	26.10.2024	Розділ 3
5	Розрахунок економічної частини	11.11.2024	11.11.2024	Розділ 4
6	Оформлення матеріалів до захисту МКР	12.11.2024	12.11.2024	ПЗ, графічний матеріал і презентація
7	Перевірка якості виконання магістерської роботи та усунення недоліків	13.11.2024	14.11.2024	Оформлені документи
9	Підписи супроводжувальних документів у нормоконтролера, керівника, опонента	15.11.2024	15.11.2024	Оформлені документи
10	Перевірка на антиплагіат та ШІ	17.11.2024	17.11.2024	Оформлені документи

6 Матеріали, що подаються до захисту МКР

Пояснювальна записка МКР, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відзив наукового керівника, рецензія рецензента, анотації до МДП українською та іноземною мовами, нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21.

ДОДАТОК Б

Структурна схема

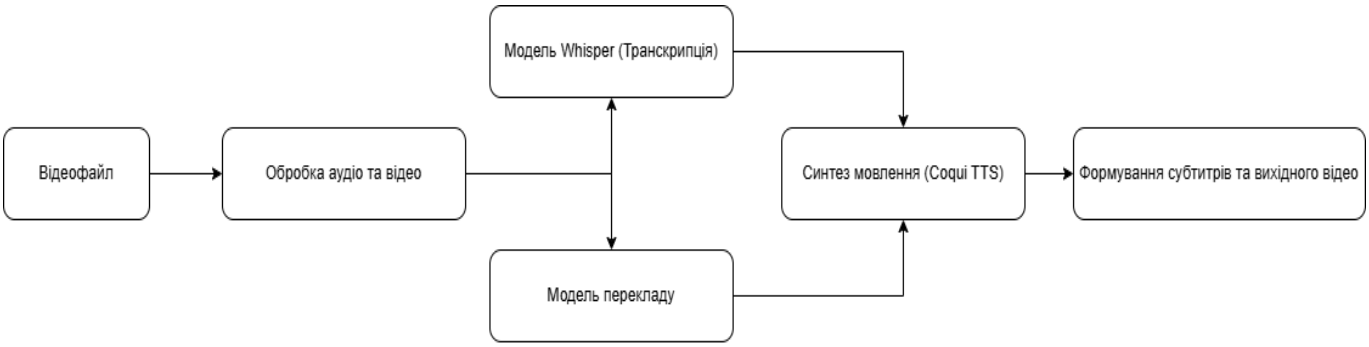

Рисунок Б.1 — Структурна схема комп’ютерної системи

ДОДАТОК В

Блок-схема алгоритму роботи

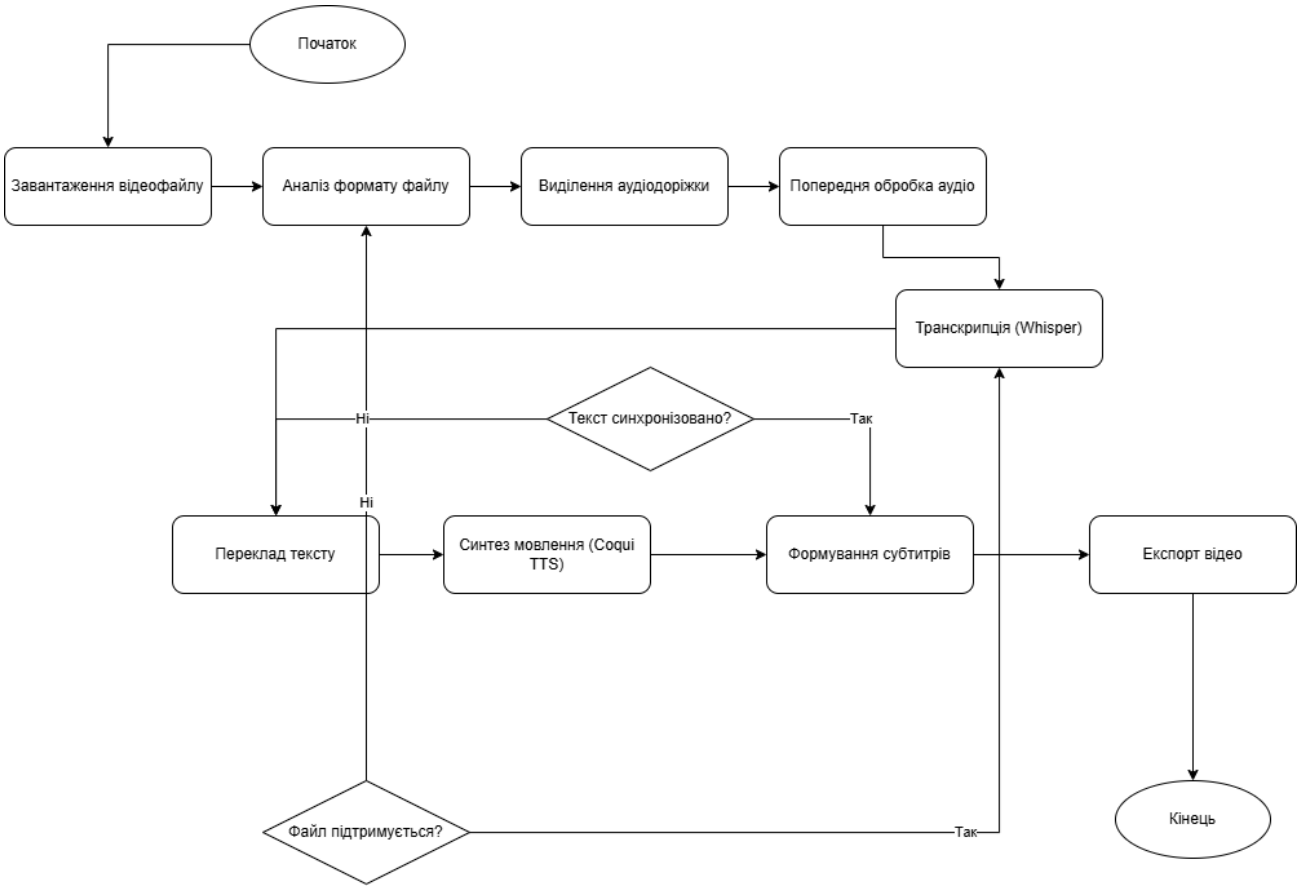

Рисунок В.1 — Алгоритм роботи програми

ДОДАТОК Г

Лістинг модуля транскипції та діаризації

```
import io
import audio2numpy as a2n

import librosa
import numpy as np
import whisperx
from pydub import AudioSegment
from whisperx.alignment import align
from whisperx.diarize import DiarizationPipeline

from config import HUGGING_FACE_API_KEY
from tasks.whisper_transcript import WhisperProcessor



class WhisperXProcessor(WhisperProcessor):
    def __init__(self, compute_type="float32", batch_size=16, **kwargs):
        super().__init__(**kwargs)
        self.compute_type = compute_type
        self.batch_size = batch_size
        self.audio = None

    def set_audio(self, audio_path):
        # audio_buffer = io.BytesIO(audio_bytes).read()
        # audio_segment = AudioSegment.from_file(audio_buffer)
```

```
            #                                   self.audio                                   =
np.array(audio_segment.get_array_of_samples()).astype(np.float32) / (1 << 15)
            self.audio = whisperx.load_audio(audio_path)

        def transcribe_audio_whisperx(self, audio_bytes=None):
            try:
                self.load_model()
                # self.set_audio(audio_bytes)
                result = self.model.transcribe(self.audio, **self.options.__dict__)
                return result
            except Exception as e:
                print(f"Error transcribing audio with WhisperX: {str(e)}")
                raise

        def save_results_to_string(self, result):
            try:
                output = io.StringIO()
                for segment in result:
                    output.write(
                        f"{segment['start']}:{segment['end']} {segment['text']}" +
                        (f": {segment['speaker']}" if 'speaker' in segment else '') + '\n'
                    )
                return output
            except Exception as e:
                print(f"Error converting results to string: {str(e)}")
                raise

        def plain_text(self, segments):
```

```
            return "\n".join(
                [f"{segment['start']}:{segment['end']} {segment['text']}: {segment.get('speaker', 'Unknown')}" for segment
                 in segments])

        def process_audio(self, audio_bytes=None):
            try:
                result = self.transcribe_audio_whisperx()
                print("before", result)
                align_result = self.align_transcriptions(result)
                assign_result = self.assign_speaker_labels(result)
                return align_result, assign_result
            except Exception as e:
                print(f"Error processing audio with WhisperX: {str(e)}")
                raise

        def align_transcriptions(self, result):
            try:
                model_a, metadata = whisperx.load_align_model(language_code=result["language"], device=self.device)
                result = whisperx.align(result["segments"], model_a, metadata, self.audio, self.device)
                return result
            except Exception as e:
                print(f"Error aligning transcriptions: {str(e)}")
                return result

        def assign_speaker_labels(self, result, num_speakers=None):
```

```
            try:
                diarize_model = whisperx.DiarizationPipeline(use_auth_token=HUGGING_FACE_API_KEY, device=self.device)
                diarize_segments = diarize_model(self.audio, num_speakers=num_speakers)
                result = whisperx.assign_word_speakers(diarize_segments, result)
                return result["segments"]
            except Exception as e:
                print(f"Error assigning speaker labels: {str(e)}")
                return result
```

ДОДАТОК Д

Лістинг синтезу мовлення

```
import os
from coqui_tts.tts.utils.synthesizer import Synthesizer
from pydub import AudioSegment
import tempfile

class SpeakerVoiceCloner:
    def __init__(self, tts_model_path: str):
        """
        Initialize the Coqui TTS synthesizer with the provided model path.
        """
        self.synthesizer = Synthesizer(tts_model_path)

    def clone_voice(self, speaker_audio_path: str):
        """
        Clone the voice of the speaker using a sample audio.
        :param speaker_audio_path: Path to the audio file with the speaker's voice.
        :return: Speaker embedding for the synthesizer.
        """
        speaker_embedding = self.synthesizer.compute_embedding(speaker_audio_path)
        return speaker_embedding

    def synthesize_speech(self, text: str, speaker_embedding):
        """
        Synthesize speech from text using the cloned speaker embedding.
```

```
        :param text: Text to synthesize.
        :param speaker_embedding: Speaker embedding obtained from clone_voice.
        :return: Synthesized audio in PCM format.
        """
        return self.synthesizer.tts(text, speaker_embedding)


class TextToSpeechProcessor:
    def __init__(self, tts_model_path: str):
        """
        Initialize the TTS processor with a Coqui TTS model.
        :param tts_model_path: Path to the Coqui TTS model.
        """
        self.cloner = SpeakerVoiceCloner(tts_model_path)

    def process_diarized_text(self, diarized_data: dict):
        """
        Process diarized text and generate synthesized audio for each speaker.
        :param diarized_data: Dictionary where keys are speaker IDs and values are lists of (text, audio_path) tuples.
        :return: Synthesized audio as a single audio file.
        """
        audio_segments = []

        for speaker_id, utterances in diarized_data.items():
            print(f"Processing speaker {speaker_id}...")
```

```
            # Use the first audio sample to clone the speaker's voice
            speaker_audio_path = utterances[0][1]
            speaker_embedding = self.cloner.clone_voice(speaker_audio_path)

            for text, _ in utterances:
                synthesized_audio = self.cloner.synthesize_speech(text,
speaker_embedding)
                audio_segment = AudioSegment(
                    synthesized_audio.tobytes(),
                    frame_rate=self.cloner.synthesizer.sample_rate,
                    sample_width=2,
                    channels=1
                )
                audio_segments.append(audio_segment)

        # Combine all audio segments into one file
        combined_audio = sum(audio_segments)

        with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as
temp_audio_file:
            combined_audio.export(temp_audio_file.name, format="wav")

        return temp_audio_file.name


# Example usage
if __name__ == "__main__":
```

```
tts_model_path = "path/to/tts/model"
processor = TextToSpeechProcessor(tts_model_path)

# Example diarized data
diarized_data = {
    "Speaker_1": [
        ("Hello, how are you?", "path/to/speaker1_sample1.wav"),
        ("I am fine, thank you!", "path/to/speaker1_sample2.wav"),
    ],
    "Speaker_2": [
        ("What about you?", "path/to/speaker2_sample1.wav"),
    ],
}

synthesized_audio_path = processor.process_diarized_text(diarized_data)
print(f"Synthesized audio saved at: {synthesized_audio_path}")
```

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Комп'ютерна система автоматичного перекладу звукової доріжки у відеофайлах

Тип роботи: магістерська кваліфікаційна робота
(МКР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Turnitin

Оригінальність 94% Схожість 6%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Водолазська Д.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Крупельницький Л.В.
(підпис) (прізвище, ініціали)