Вінницький національний технічний університет
Факультет інтелектуальних інформаційних систем та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС ПЕРЕТВОРЕННЯ «УСНЕ УКРАЇНСЬКЕ МОВЛЕННЯ – ТЕКСТ»»

Виконала: студентка 2 курсу, групи 2АКІТ-21м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Юлія Нестюк

Керівник: д.т.н., доцент, зав. каф. КСУ
В'ячеслав Ковтун

«10» грудня 2022 р.

Рецензент: к.т.н., доцент каф. АІІТ
Юрій Іванов

«15» грудня 2022 р.

Допущено до захисту
Зав. кафедри КСУ
В'ячеслав Ковтун
«14» грудня 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ

В'ячеслав КОВТУН

"03" жовтня 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Нестюк Юлії Юріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Людино-машинний інтерфейс перетворення «усне українське мовлення –текст»
керівник роботи д.т.н., доцент, зав. кафедри КСУ Ковтун В'ячеслав Васильович
затверджені наказом ВНТУ від "14" вересня 2022 року №203
2. Термін подання студентом роботи "12" грудня 2022 року
3. Вихідні дані до роботи: підтримка ОС – Windows; мова програмування – Python, мова введення звукової інформації – українська.
4. Зміст текстової частини: вступ, аналіз існуючих методів обробки природної мови та огляд варіантів їх програмної реалізації, дослідження програм-аналогів, розробка та тестування програмного забезпечення, розрахунок економічних характеристик розробки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): приклади роботи досліджуваних аналогів, UML-діаграма варіантів використання, UML-діаграма активності, UML-діаграма розгортання, основні екранні форми програмного забезпечення.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1	Ковтун В.В., д.т.н., доц., зав каф. КСУ		
2	Ковтун В.В., д.т.н., доц., зав каф. КСУ		
3	Ковтун В.В., д.т.н., доц., зав каф. КСУ		
4	Небава М. І., к.е.н., професор кафедри ЕПВМ, декан факультету МІБ		

2. Дата видачі завдання "03" жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз методів обробки природної мови та огляд варіантів їх програмної реалізації	03.10.2022	15.10.2022	
2	Дослідження програм-аналогів та визначення основних функцій розроблюваного програмного забезпечення	16.10.2022	27.10.2022	
3	Проектування структури програмного забезпечення системи	28.10.2022	01.11.2022	
4	Розробка UML-діаграм системи	02.11.2022	04.11.2022	
5	Розробка програмного забезпечення	05.11.2022	20.11.2022	
6	Тестування програмного забезпечення	21.11.2022	28.11.2022	
7	Розрахунок економічних характеристик розробки	29.11.2022	04.12.2022	
8	Оформлення пояснювальної записки, графічного матеріалу і презентації	05.12.2022	12.12.2022	
9	Захист МКР	20.12.2022	21.12.2022	

Студент _____ Юлія Нестюк
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ В'ячеслав Ковтун
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.514, 004.522

Нестюк Ю. Ю. Людино-машинний інтерфейс перетворення «усне українське мовлення – текст». Магістерська кваліфікаційна робота зі спеціальності 151 – автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2022. 127 с.

На укр. мові. Бібліогр.: 59 назв; рис.: 33; табл. 5.

У магістерській кваліфікаційній роботі розроблено людино-машинний інтерфейс, що дозволяє розпізнавати усне українське мовлення та перетворювати його у текст. Даний програмний продукт пропонує до використання алгоритм розпізнавання мовлення від Google, що демонструє вищій показник точності, порівняно з програмами-аналогами. У першому розділі загальної частини роботи розглянуто особливості розробки подібних інтерфейсів і методи обробки природної мови. В другому розділі досліджено існуючі програми-аналоги, сформульовано основні функціональні вимоги до розроблюваного програмного забезпечення та виконано його проектування. В третьому розділі магістерської кваліфікаційної роботи розроблено програмне забезпечення, наведено результати його тестування та здійснено оцінювання точності розпізнавання. У економічній частині проведено розрахунки планової собівартість розробленого програмного продукту та термін його окупності. Ілюстративна частина складається з 12 плакатів із результатами досліджень та тестуванням системи розпізнавання.

Ключові слова: людино-машинний інтерфейс, усне українське мовлення, розпізнавання мовлення, Speech to Text.

ABSTRACT

UDC 004.514, 004.522

Nestyuk Y. Y. Human-machine interface of conversion «oral Ukrainian speech – text». Master's thesis on specialty 151 – automation and computer-integrated technologies, educational program – Intelligent computer systems. Vinnytsia: VNTU, 2022. 112 p.

In Ukrainian language. Bibliography: 59 titles; fig.: 33; table 5.

In the master's thesis, a human-machine interface was developed, which allows recognizing spoken Ukrainian speech and converting it into text. This software product offers a speech recognition algorithm from Google for use, which demonstrates a higher accuracy rate compared to similar programs. In the first section of the general part of the work, the peculiarities of the development of such interfaces and natural language processing methods are considered. In the second chapter, the existing analog programs were studied, the main functional requirements for the developed software were formulated, and its design was carried out. In the third section of the master's qualification work, the software was developed, the results of its testing were given, and recognition accuracy was evaluated. In the economic part, the planned cost of the developed software product and its payback period were calculated. The illustrative part consists of 12 posters with the results of research and testing of the recognition system.

Keywords: human-machine interface, oral Ukrainian speech, speech recognition, Speech to Text.

Відгук
керівника магістерської кваліфікаційної роботи

студентки Нестюк Юлії Юріївни група 2АКІТ-21м
на тему: Людино-машинний інтерфейс перетворення «усне українське мовлення – текст».

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані фахові статті.

Представлені в магістерській кваліфікаційній роботі рішення обґрунтовані послідовним ланцюгом дій, які включають пошук інформації про проблему, її узагальнення, формулювання прикладних задач для її вирішення, проектування відповідного програмного засобу для вирішення поставлених задач, його тестування і формулювання висновків. Раціональність та ефективність прийнятих рішень доведені результатами тестування.

Дипломниця показала хороший рівень спеціальних знань і «м'яких» навичок. Дипломниця продемонструвала вміння: - вирішувати поставлені керівником завдання самостійно, згідно власноруч розробленої схеми заходів; - здійснювати пошук і узагальнення інформації; - комунікативні навички. Доведенням ерудиції та креативності дипломниці є вчасно представлена магістерська кваліфікаційна робота.

Основні результати, представлені в роботі отримані дипломницею самостійно. Матеріалу роботи властивий високий ступінь оригінальності, що доведено результатами перевірки на наявність запозичень.

Дипломниця працювала ритмічно, без суттєвих відхилень від затвердженого графіку. Втрати зв'язку з керівником не було.

Результати багатоваріантного аналізу бажано було звести в таблицю. Автор не акцентує увагу на тім, які саме з отриманих результатів опубліковані у вигляді фахових статей. Бажано було аналізувати свіжіші літературні джерела. Не всі рисунки достатньо якісні.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку **А**, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Керівник магістерської кваліфікаційної роботи

В.о. зав. каф. КСУ Ковтун В.В.

Відгук
опонента на магістерську кваліфікаційну роботу

студентки Нестюк Юлії Юріївни група 2АКІТ-21м
на тему: Людино-машинний інтерфейс перетворення «усне українське мовлення – текст».

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані чотири фахові статті.

Перший розділ магістерської кваліфікаційної роботи повністю присвячений огляду літературних та інформаційних джерел за обраною темою. Проаналізовано не менш ніж три аналоги створюваної системи. Функціональність та форм-фактор створеної системи цілком визначені на основі результатів критичного огляду літератури.

Прийняті рішення обґрунтовані результатами огляду літератури, результатами проектування та втілені в функціонуючу програмну систему. Результати її тестування доводять правильність прийнятих рішень.

Експериментальні дослідження продумані і повні. Тести охоплюють як функції інтерфейсу створеної системи, так і доводять якість та повноту виконання нею функціонального призначення, обґрунтованого на етапі проектування.

Вміст графічної частина повною мірою репрезентує всі отримані в магістерській кваліфікаційній роботі результати. Якість рисунків в графічній частині прийнятна.

Недоліки: Процес обчислень метрики якісних показників описаний занадто детально. В роботі зустрічається забагато великих таблиць. Вміст більшості з них можна було представити у вигляді діаграм.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку **A**, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Опонент на магістерську кваліфікаційну роботу

Доцент каф. АІІТ Іванов Ю.Ю.

ЗМІСТ

ВСТУП

Комунікація людей між собою може здійснюватися у різних формах: усній, письмовій та візуальній. Але не всі ці методи можна використати для взаємодії з комп'ютером. Принаймні, не в повній мірі. І якщо «спілкування» з комп'ютером за допомогою письмових команд більше не викликає складностей, то з усним мовленням все не так просто.

Для полегшення голосової взаємодії між людиною та комп'ютером розробляються спеціальні методи розпізнавання усного мовлення. Додатки, що використовують прийоми обробки природної мови для аналізу текстових і аудіо даних, чим далі, тим більше стають невід'ємною частиною нашого життя. Від нашого імені вони переглядають величезні об'єми інформації в мережі та пропонують нові і персоналізовані механізми взаємодії людини з комп'ютером. Ці додатки настільки поширені, що ми звикли до широкого спектру закулісних інструментів, таких як пошукові системи, котрі ведуть нас прямо туди, куди ми хочемо потрапити; чи віртуальні помічники, що завжди готові вислухати та відповісти [1].

Проте, не зважаючи на всю різноманітність існуючих програмних засобів, об'ємна множина питань залишається не повністю вирішеною. Складність аналізу та обробки інформації саме українською мовою також полягає у її структурних відмінностях, порівняно з англійською.

Об'єкт дослідження – процес перетворення усного українського мовлення в текст.

Предмет дослідження – методи та засоби перетворення усного мовлення в текст.

Мета роботи – створення людино-машинного інтерфейсу, що здійснюватиме перетворення усного українського мовлення в текст.

Головні завдання роботи:

- дослідження методів та засоби перетворення усного мовлення в текст, що дозволять підвищити ефективність розроблюваного програмного продукту;

- проектування та розробка програми;
- підтвердження функціонування та ефективності перетворення усного мовлення в текст, шляхом тестування розробленого програмного забезпечення.

Новизна одержаних результатів полягає у використанні алгоритму розпізнавання мовлення від Google, що дозволяє підвищити точність розпізнавання усного українського мовлення.

Публікації:

1. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Моделювання сценаріїв розвитку інфокомунікаційного процесу в бездротовому централізованому мережевому кластері», Вісник ВПІ, № 6, 2021; с. 100-113. DOI: https://doi.org/10.31649/1997-9266-2021-159-6-100-113.
2. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Визначення параметричного простору показників для оцінювання доступності інфокомунікаційного процесу в бездротовому централізованому мережевому кластері», Вісник ВПІ, № 1, 2022; с. 50-64. DOI: https://doi.org/10.31649/1997-9266-2022-160-1-50-64.
3. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Дослідження бездротового централізованого мережевого кластера з реалізацією сеансів інфокомунікаційної взаємодії в незалежних віртуальних сегментах», Вісник ВПІ, № 2, 2022; с. 68-80. https://doi.org/10.31649/1997-9266-2022-161-2-68-80.
4. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Елементи методології прецизійного фонетичного аналізу фонограм усного мовлення», Вісник ВПІ, № 2, 2022; с. 36-51. https://doi.org/10.31649/1997-9266-2022-162-3-36-51.

У першому розділі магістерської кваліфікаційної роботи представлено основні поняття предметної області, розглянуто сучасні методи обробки

природної мови та засоби їх програмної реалізації. У другому розділі досліджено існуючі програмні застосунки перетворення усного мовлення в текст, на їх основі сформульовано основні функціональні вимоги до розроблюваного програмного забезпечення та виконано його проектування. У третьому розділі здійснено розробку програмного продукту за допомогою обраної мови програмування, подано результати його тестування та оцінено ефективність роботи програми. В останньому розділі було проведено розрахунки економічних характеристик розробки.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОБРОБКИ ПРИРОДНОЇ МОВИ ТА ОГЛЯД ВАРІАНТІВ ЇХ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

1.1 Інтерфейс як засіб взаємодії з користувачем

Людино-машинний інтерфейс (Human machine interface, HMI) – термін, що охоплює інженерні рішення, що забезпечують взаємодію оператора з машинами, якими він керує.

Під "машиною" в цьому випадку розуміється система з усіх технічних засобів, що бере участь у процесі вимірювання, контролю, сигналізації та управлінні, а під «людиною» – оператор-технолог, який бере безпосередню участь в процесі управління.

Створення систем людино-машинного інтерфейсу тісно пов'язане з ергономікою, але не тотожно їй. Проектування людино-машинного інтерфейсу, охоплює: створення робочого місця (крісла, столу, або пульта керування), розміщення приладів і органів керування, освітлення робочого місця, а, можливо, і мікроклімат. Далі розглядаються дії оператора з органами керування: їх доступність і потрібні зусилля, узгодженість (несуперечність) і «захист від дурня», розташування дисплеїв і розміри написів на них [2].

Загалом, поняття людино-машинної взаємодії трактується наступним чином.

Людино-комп'ютерна взаємодія (human–computer interaction, HCI), досліджує проектування та використання комп'ютерних технологій, на межі розподілу між людьми (користувачами) та комп'ютерами. Дослідники в області HCI вивчають способи, якими люди взаємодіють з комп'ютерами, та технології проектування, котрі дозволяють людям взаємодіяти з комп'ютерами новими способами.

Галузь досліджень взаємодії людини з комп'ютером, розташовано на перетині інформатики, поведінкових наук, дизайну, медіа досліджень, а також кількох інших областей вивчення. Цей термін було введено Стюартом К. Кардом, Алленом Ньюелом і Томасом П. Мораном у їх основоположній книзі

1983 року, «Психологія людино-машинної взаємодії», хоча автори вперше використали це визначення ще 1980 року, а перше відоме його застосування, відбулося 1975 року. Цей термін означає, що, на відміну від інших інструментів лише з обмеженим використанням (наприклад, молоток насамперед, корисний для забивання цвяхів), комп'ютер має безліч застосувань, і це відбувається в межах відкритого діалогу між користувачем і комп'ютером. Сутність взаємодії людини з комп'ютером, уподібнюється до взаємин людини з людиною – аналогії, яка має вирішальне значення для теоретичних міркувань у цій області.

Люди взаємодіють з комп'ютерами багатьма способами, а інтерфейс між людьми та комп'ютерами, що вони використовують, має вирішальне значення задля полегшення цієї взаємодії .

Інтерфейс «людина-комп'ютер» може бути змальовано, як сутність зв'язку між людиною-користувачем і комп'ютером. Потік даних між людиною та комп'ютером, визначається як коло взаємодії. Ця співдія має кілька аспектів, у тому числі:

Зорова основа: Візуальна людино-комп'ютерна взаємодія, ймовірно, є найбільш поширеною областю, у дослідженнях HCI (співпраці людини з комп'ютером).

Звукова основа: Заснована на аудіо-взаємодії між комп'ютером і людиною, і є ще однією важливою складовою у системах HCI. Ця галузь стосується інформації, отриманої за допомогою різних звукових сигналів.

Цільове середовище: Умови і цілі, які встановлено для користувача.

Машинне середовище: Середовище, у якому комп'ютер приєднано до мережі.

Області інтерфейсу: Не накладені одна на одну області, містять процеси між людиною та комп'ютером, котрі не стосуються їх взаємодії. У той же час, накладені області, стосуються лише процесів, пов'язаних з їх взаємодією.

Вхідний потік: Потік даних, який починається у середовищі завдань, коли користувач має якесь завдання, що вимагає, використання власного

комп'ютера.

Вихід: Потік інформації, яка бере свій початок у середовищі машини.

Зворотний зв'язок: Цикли крізь інтерфейс, що оцінюють, модерують та підтверджують процеси, коли вони переходять інтерфейсом від людини, до комп'ютера і назад.

Придатність: Це відповідність між комп'ютерним дизайном, користувачем і завданням оптимізації людських ресурсів, потрібних для виконання цього завдання [3].

Інтерфейс (interface – поверхня розділу, перегородка) – це:

- сукупність засобів, методів і правил, що забезпечують взаємодію (керування, контроль, тощо) між елементами системи, такими як комп'ютери, периферійні пристрої, пристрої введення/виведення, комп'ютерних програм тощо;
- сукупність описів і узгоджень щодо процедури передачі керування в підпрограму та повернення до вихідної програми.

Ці терміни використовують у багатьох галузях науки й техніки, а їх значення належать до будь-якої сполуки взаємозалежних сутностей (як природничих, так апаратних і людино-машинних). Окрім того, під інтерфейсом розуміють не тільки пристрої, але й протоколи та правила взаємодії цих пристроїв.

Інтерфейси є основою взаємодії в сучасних інформаційних системах. А їх стабільність та стандартизованість дає можливість модифікації самого об’єкту, не змінюючи його принципи взаємодії з іншими об'єктами. Це відкриває функції масштабування інтерфейсу, або створення інших пристроїв, що будуть використовувати той самий спосіб взаємодії.

В інформаційних або обчислювальних системах взаємодія може здійснюватися на користувацькому, програмному й апаратному рівнях. Відповідно до цього виділяють наступні варіанти існування інтерфейсів:

- як взаємодія фізичних пристроїв (з'єднання кабелів, мікросхем);

- як взаємодія віртуальних пристроїв (власне елементи інтерфейсу програми: меню, кнопки, випадаючі списки);
- як взаємодія людини з машиною (окремі периферійні пристрої, такі як миша, клавіатура чи провідна рукавичка).

Інтерфейс користувача (ІК, user interface, UI) – сукупність засобів зручної взаємодії користувача з інформаційною системою, що обробляють та відображають інформацію, таким чином, щоб бути якнайбільше інтуїтивно-зрозумілим; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їх розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Інший термін для інтерфейсу користувача – це інтерфейс «людина-машина», коли відповідним пристроєм, є комп'ютерний інтерфейс «людина-комп'ютер».

Існує різниця між інтерфейсом користувача й інтерфейсом оператора або людино-машинною взаємодією (HMI):

- термін «інтерфейс користувача» часто застосовується стосовно персональних комп'ютерних систем та електронних пристроїв;
- інтерфейс оператора – це спосіб взаємодії, за допомогою якого, можна отримати доступ або керувати багатьма механізмами, що пов'язані із системою контролю Host;
- людино-машинним інтерфейс вважається якщо для відбиття інформації мережа обладнання або комп'ютерів, взаємопов'язана через систему MES (Manufacturing Execution System) або Host;
- інтерфейс «людина-машина» (HMI) зазвичай, є окремим для однієї машини чи частини обладнання та є способом взаємодії між людиною та обладнанням/машиною;

- інтерфейс користувача механічної системи, транспортного засобу або промислової установки, іноді називають інтерфейсом «людина-машина» (HMI). Інші терміни, що використовуються, – консоль інтерфейсу або інтерфейсний термінал оператора. Однак ці терміни стосуються «шару», що відокремлює людину, яка керує машиною, із самим об’єктом керування;
- система може мати кілька інтерфейсів для обслуговування різних користувачів (наприклад, один для людей, що займають керівні посади – обмежений набір функцій, прилаштований для простоти використання, а інший для звичайних працівників – широкий набір можливостей, пристосованих для продуктивності роботи).

Розрізняють такі види інтерфейсів:

1. Інтерфейс прямої обробки, що належить до загального класу інтерфейсів користувача і дозволяє користувачам маніпулювати наданими їм об'єктами, з використанням дій, котрі відповідають фізичному світу.
2. Графічний інтерфейс користувача (GUI), що приймає вхідні дані за допомогою таких пристроїв, як комп'ютерна клавіатура та миша, й забезпечує графічний вивід інформації на моніторі комп'ютера. У GUI-дизайні, широко застосовуються щонайменше, два різні принципи: об'єктно-орієнтовані інтерфейси користувача (OOUI) й інтерфейси, спрямовані на додатки.
3. Вебінтерфейс користувача (WUI), який приймає вхідні дані та забезпечує виведення, створенням вебсторінок, котрі передаються Інтернетом і проглядаються користувачем за допомогою програми веб-браузера. Для їх реалізації використовуються: PHP, Java, JavaScript, AJAX, Apache Flex, NET Framework, або подібні технології для забезпечення керування у дійсному часі в окремій програмі, що усуває потребу відновлення традиційного веб-браузера на основі HTML. Адміністративні веб-інтерфейси для

вебсерверів і мережевих комп'ютерів часто називаються панелями керування.

4. Сенсорний екран – дисплей, який приймає введення дотиком пальців або стилусом. Такі екрани знайшли реалізацію в технологіях створення мобільних пристроїв, багатьох інформаційних засобах точок продажу, промислових процесах і машинах, пристроях самообслуговування тощо.
5. Сенсорний інтерфейс користувача – це графічний інтерфейс, що використовує сенсорну панель або сенсорний екран як об'єднаний пристрій введення і виведення. Він доповнює або замінює інші способи виведення тактичними методами зворотного зв'язку та використовується, наприклад, в комп'ютерних тренажерах.
6. Інтерфейс командного рядка, де користувач надає вхідний сигнал, введенням рядка, що містить команду, за допомогою комп'ютерної клавіатури, а система забезпечує виведення, шляхом відбиття тексту на моніторі комп'ютера. Використовується програмістами та системними адміністраторами в інженерних і наукових середовищах, а також технічно-просунутими користувачами персональних комп'ютерів.
7. Уважний інтерфейс користувача – такий інтерфейс, що керує увагою користувача шляхом вирішення миті переривання користувача, різними видами застережень та рівнем докладності повідомлень, наданих оператору.
8. Апаратний інтерфейс – фізичний, просторовий інтерфейс, присутній на виробах у повсякденному житті (таких як тостер, приладова панель автомобіля чи навіть кабіна літака). Такий інтерфейс, переважно, є поєднанням ручок, кнопок, слайдерів, перемикачів, джойстиків і сенсорних екранів.
9. Розмовний інтерфейс, що часто наслідує людські розмови та дозволяє користувачам керувати комп'ютером за допомогою

звичайного допису (за допомогою текстових повідомлень або чатів), чи голосових команд, замість графічних елементів.

10. Пакетний інтерфейс, є неінтерактивним інтерфейсом користувача, де останній заздалегідь задає всі деталі пакетного завдання для пакетної обробки і отримує результат тільки тоді, коли всю обробку виконано. На протязі виконання завдання взаємодія комп'ютера з користувачем не передбачається.
11. Агент діалогового інтерфейсу. Цей підхід намагається персоналізувати комп'ютерний інтерфейс у вигляді анімованої особи, робота або іншого персонажа (наприклад, Microsoft Clippy the paperclip) і представляє взаємодії в діалоговій формі.
12. Інтерфейс жестів – графічний інтерфейс користувача, що приймає вхідні дані у вигляді різних порухів, накреслених за допомогою комп'ютерної миші або стилуса.
13. Перехресний інтерфейс – графічний інтерфейс користувача, в якому основне завдання полягає у перетині меж, а не зосередженні на вказівках.
14. Голографічний інтерфейс користувача – інтерфейс, що забезпечує вхід для електронних або електромеханічних пристроїв, передаванням пальця через відтворені голографічні зображення того, що в іншому випадку, було б тактильним керуванням цими пристроями, які вільно плавають у повітрі, виявленим джерелом хвилі та без доторку.
15. Інтерфейс відстеження руху, що визначає порухи тіла користувача і втілює їх у команди. Ця технологія розробляється Apple ще з 2010-х років.
16. Інтелектуальний інтерфейс користувача – це людино-машинний інтерфейс, який спрямовано на підвищення якості, продуктивності і природності взаємодії між людиною та машиною, за допомогою уявлення, обґрунтування і впливу на моделі користувача, домену,

завдання, дискурсу і середовища (наприклад, графіки, природна мова, жест).

17. Багатоекранний інтерфейс, що використовує кілька дисплеїв для забезпечення більш гнучкої взаємодії. Часто застосовується у взаємодії з комп'ютерними іграми як у комерційних аркадах, так і останнім часом, на портативних ринках.
18. Некомандний інтерфейс користувача, який спостерігає за користувачем, аби вивести його потреби і наміри, та не вимагає, щоб той проводив явні команди.
19. Об'єктно-орієнтований інтерфейс користувача (OOUI), заснований на метафорах об'єктно-орієнтованого програмування, що дозволяє користувачам керувати імітованими об'єктами та їхніми властивостями.
20. Рефлексивний інтерфейс користувача, в якому користувачі контролюють і перевизначають всю систему лише через призначений для них інтерфейс (наприклад, для зміни власних командних дієслів). Але такі функції можливі, переважно, у дуже дорогих графічних інтерфейсів.
21. Інтерфейс пошуку – це відбиття вікна пошуку на сайті, а також зорове представлення підсумків пошуку.
22. Матеріальний інтерфейс, що більше зосереджений на сенсорному та фізичному середовищі, або його елементах.
23. Інтерфейс, орієнтований на завдання. Являє собою інтерфейс користувача, що усуває проблему інформаційного перевантаження метафори робочого столу та використовує не файли, а завдання, основною одиницею взаємодії.
24. Голосовий інтерфейс користувача, який приймають введення і забезпечує виведення, створенням голосових підказок. Введення користувачем, здійснюється натисканням клавіш або кнопок чи виконується словесно.

25. Інтерфейс на природній мові – застосовується для пошукових систем і на вебсторінках. Користувач вводить запитання і чекає на відповідь.
26. Текстовий інтерфейс користувача є інтерфейсом, що виводить допис. Він можуть містити інтерфейс командного рядка, або текстове WIMP-середовище.
27. Інтерфейс з нульовим входом, який отримує вхідні дані з набору давачів, а не через запитання користувача у діалоговому вікні.
28. Масштабований інтерфейс користувача – це графічний інтерфейс, в якому об'єкти представлені на різних рівнях деталювання, а користувач може змінювати масштаб області перегляду [4].

1.2 Розпізнавання мовлення як головний компонент інтерфейсу перетворення

Розпізнавання мовлення (Speech recognition) або мовлення-у-текст (Speech to text (STT)) – процес перетворення мовленнєвого сигналу в текстовий потік. Дане поняття відмінне від «розпізнавання мови», оскільки останнє означає лише дати відповідь на питання, до якої мови належить фрагмент мовленнєвого сигналу. Часто використовується в технологіях управління комп’ютером через голосові команди, а також голосовому введенні інформації, диктуванні, транскрибуванні (стенографуванні) фонограм) [5].

Саме поняття транскрибації також має своє визначення.

Транскрибація – це розшифрування інформації з аудіо або відео в текстову форму. Її актуальність підтверджена не тільки для людей, що мають вади слуху, а й для тих, хто надає перевагу споживанню текстового контенту.

Мета транскрибації – зрозуміле і якісно перетворення усного мовлення в текстовий формат.

Її завдання:

- повний і точний розбір всього сказаного спікерами;

- розбиття мовлення на окремі речення;
- видалення слів-паразитів і пауз.

Розшифровування використовується в субтитрах до відео, у статтях та постах, а також для аналізу та досліджень. Далі наведено таблицю 1.1, у якій вказано, де використовується транскрибація [6].

Таблиця 1.1 – Види та особливості транскрибації

Вид транскрибації	Особливості
Диктування	Автор читає підготовлений текст – повільно, виразно, дотримуючись пауз за розділовими знаками. Текст може використовуватися під час підготовки статей, дописів до соцмереж тощо.
Лекція	Студенти записують промову викладача на диктофон, щоби надалі перевести аудіофайли в текст, якщо там трапляються складні терміни та незнайомі імена. Використовується, коли лектор швидко розмовляє й записати матеріал у такій ситуації дуже складно.
Підкаст	Формат радіо, де проблеми з вимовою трапляються рідко. Підкаст може бути монологом чи діалогом. Під час розшифровування обов’язково вкажіть ім’я людини. Розшифровування потрібно для субтитрів.
Інтерв’ю	Формат, у якому бере участь двоє та більше осіб. Транскрибація необхідна для субтитрів у відео, написання статті чи посту. При чому, головною ціллю є передати емоції та головну думку інтерв’юйованого, а не притримуватися суворої дослівності.
Синхрон	Схожий на інтерв’ю. У кадрі експерт коментує якусь тему, камера записує не тільки голос, а і звукове тло, що супроводжує запис.

Вебінар	Мова одного чи кількох експертів вебінару. Розшифровування схоже на формат підкасту – потрібно вказати ім’я спікера. Транскрибацію використовують для субтитрів, написання статті чи книги.
Фокус-група	Фокус-група бере участь у маркетингових дослідженнях та соціологічних опитуваннях. Спочатку всі люди можуть говорити спокійно, та під кінець багато хто починає перебивати один одного. Щоб уникнути мішанини, запитуйте кожного учасника по черзі. Розшифровування потрібно для звітів, аналізу та досліджень.
Конференція	У конференції є ведучий, він же модератор, і спікери. Тут також важливо вказати ім’я того, хто говорить. Розшифровування використовується для субтитрів, написання статей та постів.
Телефонні розмови	Розшифровування використовується для аналізу клієнтів, збору даних про них – інтереси, потреби, болі та проблеми. Також це допомагає покращити якість роботи менеджерів. Транскрибація дзвінків використовується не тільки для бізнесу, а і для вирішення специфічніших завдань – наприклад, силовики можуть прослуховувати розмови потенційних злочинців.
Субтитри	Використовуються у відео. Текст має точно повторювати мову того, хто говорить. Щоб розбити її на підтеми використовуються тайм-коди.

Монтажний лист	Застосовується в кіно та кліпах. Це таблиця, де вказано кадри з репліками героїв та технічними коментарями. Тут потрібно не тільки розписати промову персонажа, а й зафіксувати все, що зараз перебуває в кадрі.

В загальному випадку розпізнавання мовлення виконується наступним чином:

1. Першим кроком є оцінка якості мовленнєвого сигналу. Тут визначається рівень спотворень та перешкод.
2. Отриманий результат подається в модуль акустичної адаптації, що керує, необхідними для розпізнавання, розрахунками параметрів мовлення.
3. Далі відбувається виокремлення ділянок, які містять мовлення, для подальшої оцінки його параметрів. Вона складається з виділення фонетичних та просодичних ймовірнісних характеристик для синтаксичного, прагматичного та семантичного аналізу.
4. Розглянуті параметри потрапляють в основний блок системи – декодер. Цей елемент проводить зіставлення вхідного мовленнєвого потоку з інформацією, яка зберігається в мовних і акустичних моделях та визначає найбільш ймовірну послідовність слів. Вона і є кінцевим результатом.

Класифікація систем розпізнавання мовлення відбувається за наступними критеріями:

- розмір словника (словник великого розміру, обмежений набір слів);
- залежність від диктора (дикторозалежні й дикторонезалежні);
- тип мовлення (роздільне або злите);
- призначення (командні системи, системи диктування);
- використовуваний алгоритм (динамічне програмування, приховані марковські моделі, нейронні мережі);

- тип структурних одиниць (слова, фрази, фонеми, алофони, дифони);
- принцип виділення структурної одиниці (лексичні елементи, розпізнавання по шаблону).

Алгоритми та методи, які застосовуються для розпізнавання мовлення в системах управління можна класифікувати наступним чином:

На основі порівняння з еталоном – тимчасові динамічні алгоритми (динамічне програмування, dynamic time warping).

Контекстно-залежна класифікація, при реалізації якої з мовленнєвого потоку виокремлюються лексичні елементи – фонеми та алофони, що потім з'єднуються в склади та морфеми:

- нейронні мережі (neural networks);
- приховані марковські моделі (hidden Markov model);
- методи дискримінантного аналізу, що базуються на Байесівській дискримінації (Bayesian discrimination) [7].

Основний підхід до розпізнавання мовлення для використання в комп'ютерно-інтегрованих інтерфейсах засновується на генеративній моделі аналізу та розпізнавання образів. Аналіз, проведений через синтез, виявився найбільш продуктивним, у порівнянні з дискримінативною моделлю, для багатьох задач розпізнавання мовленнєвого сигналу. Теорія генеративної моделі успішно використовується в обробці та розпізнаванні різних об'єктів, сигналів, полів і зображень. Зокрема, в біоінформатиці й радіофізиці.

Під час розпізнавання вхідного сигналу він перетворюється у послідовність векторів-ознак, що надходять в проміжку, наприклад, раз на десять мілісекунд. В такий спосіб формується відлік сигналу, із яким в подальшому працює декодер, який реалізує алгоритм певної схеми розпізнавання. Будь-який розпізнавач працює з базою даних і базою знань.

В найбільш поширеній схемі розпізнавання, процесі перетворення мовлення в текст, формуються:

- абетка базових елементів – фонем, для кожної з яких методами навчання-самонавчання за мовленнєвим корпусом завчасно створюється акустична модель;
- словник, для слів, що на початку розпізнавання буде перетворений у композитні акустичні моделі за фонемною транскрипцією;
- лінгвістична модель, що обмежує допустимі послідовності слів. Завчасно будується за текстовим корпусом та може мати ймовірнісний або детермінований характер [8].

Проте, існує ряд критеріїв, які перешкоджають реалізації якісних систем розпізнавання мовлення. Нижче наведено деякі з них.

Вимовляючи одне і те ж слово чи фразу в різні моменти часу, під впливом великої кількості факторів, диктор генерує спектрально-часові розподіли енергії, що відчутно різняться. Це явище набуває особливого посилення якщо порівняти спектрограми однієї і тієї ж фрази, вимовленої різними людьми.

Темпи мовлення, що варіюються в широких межах, породжують непропорційне стискання та розтягування звуків.

Ці два фактори також є причиною коартикуляційної нестаціонарності, що означає зміну впливу сусідніх звуків один на одного.

В неперервному мовленнєвому потоці розпізнавання мовленнєвих одиниць ускладнюється також через неточне визначення меж [9].

В загальному, метою такого розділу наукової дисципліни розпізнавання образів, як розпізнавання мовлення є отримання детальної інформації на основі вхідного голосового (мовленнєвого) сигналу. Проблеми, що вирішуються в цій області наступні:

- диктувальна машина, введення інформації голосом;
- автоматичне перетворення мовленнєвого сигналу в текст;
- пошук ключових мовленнєвих одиниць в потоці мовлення;
- верифікація та ідентифікація диктора;
- формулювання сенсу голосових повідомлень;

- розпізнавання мови диктора, його акценту;
- адаптація системи до акустичного каналу та голосу мовця;
- розпізнавання фізичного та емоційного стану диктора;
- усний переклад з однієї мови на іншу [10].

Завдяки таким широким можливостям системи розпізнавання стали поширені в різних галузях використання, зокрема:

- в телефонії, для створення голосових систем самообслуговування, що автоматизують обробку дзвінків та надають довідкову інформацію, консультують, проводять опитування тощо;
- в системах типу «Розумний будинок» для створення голосових інтерфейсів управління самими будинками та побутовою технікою;
- в автомобілях, для голосового управління, наприклад в системі навігації.

Окрім того, в десктопах та ноутбуках для голосового вводу в комп'ютерних іграх, додатках і месенджерах. Слід виокремити також соціальні сервіси для людей з обмеженими можливостями [7].

1.2.1 Обробка природної мови як більш загальний напрям розпізнавання

Явище розпізнавання мовлення є одним із завдань більш широкого поняття – обробки природної мови.

Обробка природної мови (Natural Language Processing, NLP) – спільний напрямок штучного інтелекту та комп'ютерної лінгвістики, що вивчає проблеми комп'ютерного аналізу і синтезу текстів природною мовою. Стосовно штучного інтелекту аналіз означає розуміння мови, а синтез – генерацію грамотного тексту.

Головними завданнями цього напряму є:

- видобуття даних: їх вивчення, пошук зв'язків та закономірностей між ними;

- синтез мовлення: озвучення (прочитання, відтворення) тексту голосом, що максимально наближений до природного;
- генерування природної мови: конвертування комп'ютерних даних у природну мову людини;
- розпізнавання мови: виведення/розпізнавання тексту з картинок, відсканованих документів або файлів у PDF форматі; а також розпізнавання мовлення, здійснене людським голосом;
- машинний переклад: автоматичний переклад з однієї людської мови на іншу. Складність даного завдання полягає у різних підходах до «розуміння» тексту у людини та машини;
- питально-відповідальні системи: відповіді на питання, поставлені людською мовою. Складність даного завдання полягає у тому, що не всі питання завжди мають чітку відповідь, а деякі з них є «філософським»;
- визначення тематики: поділ тексту на частини з подальшим розпізнаванням провідної теми для кожної з них;
- інформаційний пошук: пошук, розпізнавання та видобування інформації;
- добуття семантичних даних: отримання семантичної інформації з тексту;
- спрощення тексту: зміна, розширення або інша обробка інформації для спрощення структури або граматики тексту зі збереженням основної думки;
- отримання зв'язків: визначення відносин між об'єктами у певній частині тексту;
- розв'язання лексичної багатоманітності: формування списку можливих значень конкретного багатозначного слова, серед яких можна вибрати найбільш доречне у деякому контексті;
- розпізнавання абревіатур та заголовків;

– детектування окремих лінгвістичних одиниць;
– морфологічна декомпозиція: перетворення окремих термінів та визначень, що належать до тієї чи іншої галузі (наприклад, медичних або технічних), у форму зрозумілу звичайному користувачу [11].

Дещо узагальнено ці завдання подано на рисунку 1.1 [12].

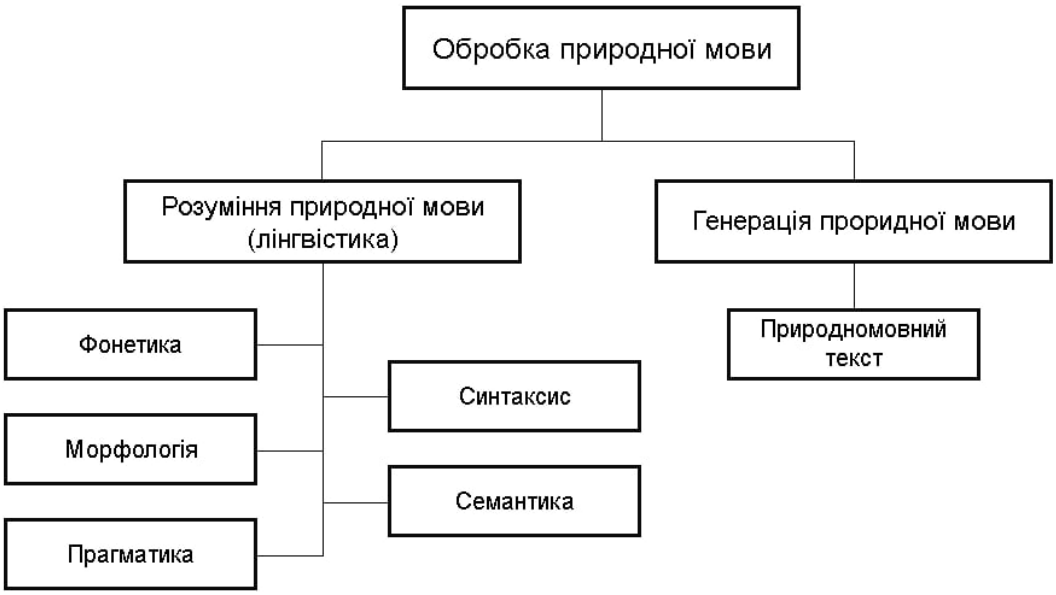


Рисунок 1.1 – Структура напряму «Обробка природної мови»

Існують спеціальні підходи до виконання вище наведених завдань:

Статистичний підхід, в основі якого лежить припущення, що найуживаніші слова в тексті можуть пояснювати його зміст. І визначення кількості повторень таких слів або їх сполучень є основним завданням даного підходу. Латентно-семантичний підхід є різновидом статистичного методу та базується на ідеї, що сукупність усіх контекстів, у яких зустрічається або не зустрічається дане слово, визначає множину взаємних обмежень для виявлення схожостей у значеннях слів. Основною проблемою, що питає на ляху статистичних підходів є те, що будь-який текст розглядається виключно як набір слів (без смислового зв'язку).

Символічний підхід, який базується на явному представленні знань і

здійснюється за допомогою використання добре досліджених схем та алгоритмів, що працюють з ними. Його метою є здійснення глибинного аналізу лінгвістичних явищ, джерелом знань про мову в яких виступають словники, правила та формули, розроблені людьми.

Лінгвістичний підхід. Він складається з чотирьох рівнів:

- графематичний, що полягає у виділенні окремих елементів тексту, таких як розділи, абзаци, речення і т. д.;
- морфологічний, який полягає у визначенні морфологічних характеристик окремого слова;
- синтаксичний, що відповідає за визначення синтаксичної залежності слів у реченнях;
- семантичний, який пов'язаний зі смисловим розумінням тексту, що включає розробки у сфері штучного інтелекту.

Обмежена кількість дослідницьких досягнень у цій сфері зумовлена складністю людської мови.

Конективістський підхід, що відповідає за обробку загальних моделей з використанням конкретних прикладів мовних явищ. Найбільш значуща відмінність даного методу обробки природної мови від інших статистичних методів полягає у поєднанні статистичних знань та різних теорій уявлень, які дозволяють трансформувати логічні формули та працювати з логічними висновками.

Прихована марковська модель. Зазвичай вона зображується у вигляді графічної системи, кожна вершина якої – випадкова змінна, яка з певними ймовірностями може набувати будь-якого значення. Переходячи з одноо стану в інший вона кожен раз породжує один з декількох можливих вихідних символів. Це обумовлює великий розмір вихідної множини всіх можливих станів та унікальних символів. Вихідні дані можуть бути видимі, на відміну від початкових станів системи, що завжди є прихованими.

N-грамні моделі, назва яких походить від їх побудови, що складається з послідовності з n елементів (речень, слів, букв, звуків тощо). Вони дозволяють

розраховувати ймовірність появи будь-якого елемента за відомих ймовірностей появи його попередників. Результати діяльності таких моделей зводяться до скінченної множини ймовірностей, над кожною з яких, після обчислення повторюваності відповідних n-грам, можна буде провести оцінку.

Існують також і деякі методи машинного навчання, що базуються на диференційних (роздільних) моделях:

- метод допоміжних векторів, який побудований на певній множині властивостей і допомагає провести класифікацію слів за категоріями;
- метод умовних випадкових полів, який формує логістичну регресію для послідовності даних. На основі спостережень за деякою змінною вона може передбачити її подальший стан [13].

Окрім того, існують рівні обробки природної мови, що базуються на рівнях мовної структури. Всього їх шість:

1. Фонологічний аналіз, який досліджує організацію та інтерпретацію звуків мовлення у мові, за допомогою базових правилами фонологічного аналізу (фонетичних, фонемних та просодичних).
2. Морфологічний аналіз можна визначити як аспект дослідження, що полягає у ідентифікації, аналізі та описі структури або форм слів у мові.
3. Лексичний аналіз, що виконує поділ тексту на розділи, абзаци, речення та/або слова.
4. Синтаксичний аналіз, метою якого є аналіз слів у реченнях, що допомагає розумінню їх граматичної структури. На цьому рівні обробки природної мови слова перетворюються в структури, що демонструють, який зв'язок між ними існує. Варто зазначити, що деякі словосполучення можуть бути використані порушуючи граматичні правила або правил комбінування слів у мові.
5. Семантичний аналіз, головним завданням якого є пошук та визначення значеннь слів, фраз та речень у мові. Він сприяє

дослідженню можливих смислових навантажень речення у тому чи іншому контексті.

6. Прагматичний аналіз є аспектом дослідження, що допомагає зрозуміти, механізми комбінування речень з різними контекстами для їх подальшого формування в абзаци, тексти або діалоги. Прагматичний аналіз полягає в інтерпретації окремих речень у відповідних для них контекстах [11].

Сьогоденне використання методів обробки природної мови зводиться до вирішення задач розпізнавання мовлення, автоматичного або напівавтоматичного перекладу, виконання пошукових запитів (письмових чи усних), підбору контекстної онлайн-реклами, аналізу настроїв для задач маркетингу, чат-ботів та голосових помічників [14].

1.2.2 Морфологічна класифікація природних мов

Розглядаючи питання обробки природної мови, варто також звернути увагу на морфологічну класифікацію мов.

Морфологічна класифікація мов – типологічна класифікація мов світу, що побудована на основі принципів морфологічної будови слів.

Відповідно до цієї класифікації усі мови поділяються на:

- кореневі (ізоляційні, аморфні);
- аглютинативні;
- флективні (фузійні);
- полісинтетичні (інкорпоруючі, багатоскладові) [15].

Кореневі (ізоляційні) мови – мови, які не мають афіксів і граматичні значення виражають способом прилягання одних слів до інших або за допомогою службових слів. До ізоляційних мов належать більшість мов Південно-Східної Азії, китайська, в'єтнамська та ін.

Кореневими вважаються мови, в яких слово зазвичай дорівнює кореню, а відносини між словами передаються перш за все синтаксично (порядком слів, службовими словами, ритмом, інтонацією); в них немає афіксів

формоутворення, також відсутні і граматичні зміни слів, пов'язаних з такими афіксами.

В кореневих мовах виділяють такі основні ознаки:

- відсутність форм словозміни: незмінність слів за відмінками, числом, особою, часом тощо;
- вираження синтаксичних відношень зазвичай за допомогою порядку слів.

До супутніх ознак належать:

- переважно односкладові корені слів;
- накладення значних обмежень на структуру складу;
- всі слова неможливо розподілити за граматичними класами (типу частин мови), тобто одне слово може виступати в різних граматичних функціях, і т. п.;
- наявність складових музичних тонів [16].

Аглютинативні мови (лат. agglutino – приклеюю) – мови, в яких граматичні форми й похідні слова утворюються додаванням однозначних афіксів до незмінюваних основ слів. Афікси розташовані переважно після основи або кореня слова, вони і характеризують окремі відмінки іменників або час, спосіб, особу дієслова. До аглютинативних мов належать тюркські, фінно-угорські, монгольські, іберійсько-кавказькі та окремі мови народів Азії, Африки, Океанії [17].

Флективні мови – це такі мови, в яких у вираженні граматичних значень провідну роль відіграє закінчення (флексія). До флективних мов належать індоєвропейські та семіто-хатські.

У флективних мовах закінчення є багатозначними та нестандартними, вони приєднується до основи, що зазвичай не вживається без флексії, і органічно зливаються з нею, утворюючи єдиний сплав (внаслідок цього на стику морфем можуть відбуватися різні зміни), на відміну від аглютинативних мов, де афікси є однозначними, стандартними і механічно приєднуються до повних слів. Друга назва флективних мов – фузійні, походить від самого

поняття фузії, що означає формальне взаємопроникнення контактуючих морфем, яке призводить до стирання меж між ними [18].

Флективні мови поділяють на:

- синтетичні – мови, у яких взаємозв'язки між словами виражаються завдяки їх формам. Оскільки словоформи в такому різновиді мов передають як лексичне, так і граматичне значення, зокрема значення відмінків, форми яких відтворюють граматичні відношення слів. До синтетичних мов належать: українська, російська, польська, литовська, а також деякі мертві мови: давньогрецька, готська, латинська.
- аналітичні – різновид мов, у якому відношення між словами у реченні виражаються порядком розташування повнозначних слів та за допомогою службових частин мови. В таких мовах відмінкові форми слабко виражені або їх зовсім немає. До аналітичних мов належать англійська, французька, болгарська і деякі інші індоєвропейські мови [15].

Полісинтетичні (інкорпоративні) мови – це такі мови, в яких різні частини висловлення у вигляді аморфних словооснов об'єднані в єдині складні комплекси, сукупності яких оформлюються за допомогою службових елементів. Також інкорпоративні мови можна охарактеризувати як різновид синтетичних мов, де всі граматичні значення передаються в складі конкретного слова, для якого характерне послідовне приєднання відповідних морфем. До полісинтетичних мов належать абхазько-адигські, ескімосько-алеутські, чукотсько-камчатські та багато індіанських мов [19].

В залежності від категорії, до якої належить мова, що має бути розпізнана, обирають відповідні методи її обробки.

1.2.3 Глибинне навчання та нейронні мережі в обробці природної мови

Переважна частина технологій обробки природної мови працює завдяки глибинному навчанню (deep Learning) – галузі машинного навчання, основою

якого є набір алгоритмів, що намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома шарами обробки, які побудовано в результаті декількох лінійних або нелінійних перетворень [20].

Одним із методів функціонування штучного інтелекту є так зване машинне навчання, практичною реалізацією можливостей якого є створення алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальше використання для самонавчання.

Останнє – головна особливість і пріоритетне завдання: не вирішити одну конкретну задачу напряму, а навчитися в процесі застосування рішень виконувати інші подібні завдання. Для цього використовуються математичний та статистичний аналізи, оптимізація та інші техніки опрацювання даних.

Загальну мету machine learning можна охарактеризувати як автоматизацію та оптимізацію рішень складних рутинних задач у найрізноманітніших сферах, від метеорології до комунікацій (про котрі піде мова нижче). Сучасний фунціонал інструментів машинного навчання справді вражає: додатки вже здатні розпізнавати мову, жести та образи, проводити медичну та технічну діагностики, біржовий і фінансовий аналізи, систематизувати документацію та виявляти спам. Тотальна діджиталізація призводить до накопичення величезних обсягів інформації у різних галузях, що, в свою чергу, розширює сферу застосування machine learning.

Технології машинного навчання активно використовуються і в Україні, зокрема у сфері комунікацій. Нещодавно кейс OMD Media Direction медійного дивізіону комунікаційної групи AGAMA Communications здобув бронзу щорічної премії WARC Media Awards. В основі кейсу – медіа-мікс на базі технологій machine learning для банку ПУМБ. Вони застосували інструмент власної розробки Brand Metrics для аналізу та подальшої оптимізації роботи одного з каналів продажів кредитних продуктів банку кол-центру: збільшення вхідного потоку дзвінків і віддачі інвестованих коштів [21].

Існує багато підходів до реалізації машинного навчання.

Глибинне навчання (deep learning) – варіант реалізації машинного навчання, який складається з кількох прихованих шарів штучної нейронної мережі. Падіння цін на апаратне забезпечення та розвиток графічних процесорів для особистого використання протягом останніх кількох років зробили свій внесок у розвиток поняття глибинного навчання, що ним людський мозок обробляє світло для зору та звук для слуху. Успішною реалізацією цього напрямку є технології комп'ютерного зору та розпізнавання мовлення.

Штучні нейронні мережі (artificial neural network) – це алгоритм навчання, що створений за прикладом біологічних нейронних мереж. Сучасні НМ є нелінійними статистичними інструментами моделювання даних, де інформація, оброблена за допомогою конективістського підходу до обчислень, структурується в термінах взаємозв'язаних груп штучних нейронів. Даний алгоритм застосовується у задачах пошуку закономірностей даних, для моделювання складних взаємозв'язків входів і виходів системи а також у виявленні статистичних структур в невідомому спільному розподілі ймовірностей спостережуваних величин.

Навчання дерев рішень (decision tree learning) – метод, що використовує дерево рішень як передбачувану модель. Її задачею є відображення спостережень про предмет на висновки про його цільове значення.

Індуктивне логічне програмування (inductive logic programming) – універсальне представлення вхідних даних, зворотного розповсюдження та гіпотез, що лежить в основі однойменного підходу машинного навчання правил. Основне знання, в закодованому вигляді, та набір прикладів, представлених логічною базою даних фактів, формують навчальну вибірку для системи ІЛП, метою якої є виведення гіпотетичної логічної програми, що має наслідками лише позитивні приклади. Індуктивне програмування є пов'язаною областю, що для представлення гіпотез розглядає будь-які види мов програмування, такі як процедурні програми.

Навчання асоціативних правил (association rule learning) – метод

відкриття цікавих залежностей та зв'язків між деякими величинами у великих базах даних.

Баєсові мережі (bayesian networks, інші назви: мережа переконань – belief network, спрямована ациклічна графова модель – directed acyclic graphical model) – це ймовірнісна модель, що представляє набір випадкових величин та їх умовних незалежностей у вигляді спрямованого ациклічного графа. Одним із варіантів використання таких мереж є встановлення ймовірнісних взаємозв'язків між хворобами та їх симптомами, а також використовуючи останні висувати передбачення наявності перших. Для ґрунтовного аргументування винесення рішень та здійснення навчання Баєсової мережі існує велика кількість ефективних алгоритмів.

Кластерний аналіз (cluster analysis) – це деякий набір спостережень, який розподілено на підмножини, що мають назву кластерів. Розподіл відбувається таким чином, щоб спостереження в межах одного кластеру відповідали деякому наперед встановленому критерію подібності, в той час, як варіанти спостережень, взяті з різних кластерів, є різними. Різні методики кластеризації припускають різні варіанти структури даних, що зазвичай визначені деякими мірами подібності, і оцінювані, наприклад, внутрішньою компактністю (подібністю членів одного й того ж кластеру) та відокремленістю між різними кластерами. Інші методи ґрунтуються на оцінюваній густині та зв'язності графа. Методи кластеризації відносяться до машинного навчання без учителя і є поширеним застосуванням статистичного аналізу даних.

Метод опорних векторів (support vector machines) – це набір пов'язаних між собою методів навчання з учителем, що використовуються для регресії та класифікації. За існування набору тренувальних даних, що кожні з них належать до відповідної категорії, про яку заздалегідь відомо, алгоритм тренування даного методу повинен будувати модель для передбачення належності нових даних до однієї з уже існуючих категорій.

Навчання з підкріпленням (reinforcement learning) займається дослідженням поведінки агента в тому чи іншому середовищі, який має діяти

таким чином, щоб деяке уявлення про довготривалу винагороду було максимальним. Дані алгоритми намагаються віднайти таку політику, що відображає залежність станів світу та дій, які агент повинен виконувати в цих станах. Різниця між навчанням з підкріпленням і навчання з учителем полягає в тому, що першому ніколи не представляються пари правильних входів/виходів, і ніколи явно не виправляються дії, які є не зовсім оптимальними.

Деякі алгоритми навчання (зазвичай ті, що пропонують навчання без учителя) мають на меті відкриття кращих вхідних представлень, що надаються протягом тренування. Класичні приклади включають кластерний аналіз та метод головних компонент. Алгоритми навчання представлень (representation learning) схильні до попередньої обробки даних, а саме перетворення вхідної інформації, що подавалась би у найбільш зручному форматі, для її подальшої класифікації або виконання на її основі прогнозування. Це дає можливість відбудови початкових пунктів входу, що йдуть з невідомого розподілу та породжують, не завжди точні для малоймовірних, за такого розподілу, конфігурацій, дані.

Нижче представлено деякі алгоритми та їх цілі:

- навчання многовидів – створення представлень низької розмірності;
- розрідженого кодування – створення розріджених представлень (таких, що містять багато нулів);
- навчання полілінійного підпростору – представленя низької розмірності, що утворені з тензорних представлень багатовимірних даних (без перетворення їх на багатовимірні вектори);
- глибинного навчання – створення ієрархії ознак, що в ній відслідковується структурна залежність між ознаками різних рівнів.

Існує думка, що розумною машиною можна вважати таку з них, яка

навчається представлення; що здатна розплутати чинники, які лежать в основі варіацій опису спостережуваних даних.

Навчання розріджених словників – це метод, в якому дані подаються у вигляді лінійної комбінації базисних функцій. Передбачається, що коефіцієнти цих функцій є розрідженими (звідки й походить назва методу). Нехай x є d-вимірними даними, а D є матрицею d на n, кожен стовпчик якої представляє базисну функцію. r є коефіцієнтом для представлення x за допомогою D. З математичної точки зору, навчання розрідженого словника (sparse dictionary learning) означає розв'язання $x \approx Dr$, де r є розрідженим. Взагалі кажучи, передбачається, що n є більшим за d, щоби дати свободу для розрідженого представлення.

Навчання словника разом із розрідженим представленням є строго NP-складним, і є також складним і для наближеного розв'язання. Популярним евристичним методом навчання розріджених словників є K-CPM.

Навчання розріджених словників використовується в кількох контекстах.

Головною метою задач класифікації є ідентифікування нових, тобто раніше небачених, даних. І використовуючи припущення, про завершене представлення словників кожного з класів, рішення знаходиться в тому з них, де ці дані представлені в найкращому розрідженому вигляді.

Також ця технологія знайшла своє місце у задачах обробки зображень, а саме, їх знешумлюванні. Де на відміну від шуму, розрідженим словником може бути представлено лише чисту частину зображення.

Навчання подібностей та мір – задача, в якій машині, що навчається, надаються до розгляду пари прикладів, що мають різні ступені подібності. Метою такого методу є навчання машини будуванню функцій подібності (мір відстані), які можуть робити передбачення відповідно до різних об’єктів. Цей алгоритм інколи використовується в рекомендаційних системах.

Машинне навчання на основі правил (rule-based machine learning) – це загальний термін для будь-якого методу машинного навчання, що ідентифікує,

навчається або виводить «правила» для зберігання, маніпулювання або застосування знань. Визначальною рисою систем, що використовують даний метод машинного навчання є ідентифікація та використання набору реляційних правил, що в сукупності представляють знання, отримані системою. На відміну від інших систем машинного навчання, що зазвичай ідентифікують одиничну модель, яку для отримання передбачення можливо універсально застосовувати до будь-якого випадку.

Представниками машинного навчання на основі правил є такі системи:

- навчання класифікації;
- навчання асоціативних правил;
- штучні імунні системи.

Генетичні алгоритми (genetic algorithms) – це евристичні алгоритми пошуку, що імітують процес природного добору та використовують методи мутації та схрещування для винайдення нового генотипу, сподіваючись на знаходження добрих розв'язків поданої задачі. В 1980-90-х роках генетичні алгоритми знаходили деяке застосування в машинному навчанні. І навпаки, методики машинного навчання використовувалися для покращення продуктивності генетичних та еволюційних алгоритмів.

Системи навчання класифікації (LCS – learning classifier systems) – це сімейство алгоритмів машинного навчання на основі правил, які поєднують у собі дві складові: відкривальну, що зазвичай реалізується у вигляді генетичного алгоритму; та навчальну, яка може виконувати деякі види навчання (кероване, з підкріпленням, або спонтанне). Такі системи мають на меті ідентифікацію набору контекстно-залежних правил, що їх сукупне зберігання та кускове застосування знань дає підґрунтя для винесення передбачень [22].

Класифікація методів машинного навчання за типом задач наведена на рисунку 1.2 [23].

Рисунок 1.2 – Класифікація методів машинного навчання

Машинне навчання за допомогою нейронних мереж застосовується у багатьох напрямках та способах обробки природної мови, таких як: обробка за допомогою векторного представлення, машинний переклад, голосові помічники, системи типу «питання-відповідь», стислий переказ тексту та ін. [14].

1.2.4 Векторне представлення текстової інформації

У традиційній обробці природної мови слово розглядається як набір дискретних символів, що в подальшому представляється у вигляді одноразових векторів. Але відсутність можливості встановлення схожості одноразових векторів є суттєвим недоліком такого підходу. І варіантом вирішення даної проблеми є тільки кодування схожості у самих векторах.

Метод векторного представлення полягає у поданні рядків у вигляді векторів зі значеннями, які присвоюються таким чином, щоб контекстно-близькі слова мали відповідний ступінь подібності векторів. Такий спосіб представлення є стартовою точкою для більшості завдань NLP та підвищує ефективність застосування глибинного навчання на малих вибірках даних [24].

Word2Vec – технологія, що використовується для різних задач аналізу та має в основі подання слів у вигляді векторів певної розмірності. Вона розміщує слова за ступенем схожості понять, що вони позначають. Так вектори слів «кіт» та «собака» матимуть між собою значно меншу відстань, ніж «кіт» та «літак». Ця особливість дозволяє подавати дані в більш гнучкому вигляді та потім легше використовувати їх в різних видах навчання.

Для створення бази відповідностей «слово – вектор», алгоритм спочатку переглядає весь наданий йому текст, створюючи «словник», який в подальших ітераціях роботи буде використаний для визначення відповідних векторів [25].

На рисунку 1.3 представлено принцип роботи даної технології: кожне слово позначається певним вектором, а контекстно-схожі поняття мають значне співпадіння параметрів (в даному випадку кольорів) [26].

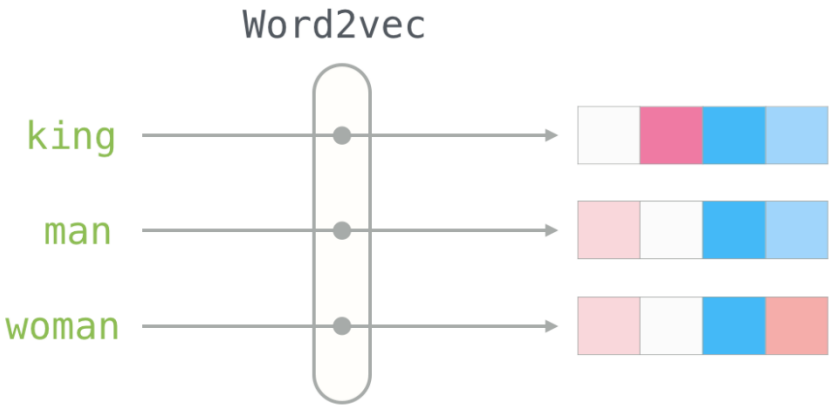


Рисунок 1.3 – Приклад роботи технології Word2Vec

1.3 Програмні застосунки для мовної комунікації з людино-машинним інтерфейсом

До найпоширеніших мов програмування, що використовуються для обробки природної мови належать Java та Python. Можливим, також є використання API хмарних сервісів для задач NLP, але в цьому випадку не варто залишати без уваги специфіку їх тарифікації. На противагу останнім існує велика кількість бібліотек з відкритим вихідним кодом та безкоштовних бібліотек, що надають значний перелік переваг розробнику, власнику та

кінцевому споживачу таких систем. А ключовий фактор при виборі таких бібліотек – це підтримка української мови [27].

У таблиці 1.2 наведено кілька бібліотек обробки природної мови та їх короткі характеристики.

Таблиця 1.2 – Бібліотеки обробки природної мови

Назва	Ліцензія	Підтримувана мова програмування	API для інших мов програмування	Мови що підтримуються
OpenNLP	Apache Lic.v2.0	Java	Java	english
Stanford NLP	GNU General Public Licens	Java	Javascript, Python (or Jython), Ruby, Perl, F#, and other .NET and JVM language	english, germany, french, italian, ukrainian, russian
LingPipe	Alias-i	Java	Java	english, germany, french, italian
GATE	GNU General Public License	Java	Json, java	english, germany, french, italian, russian
LanguageTool	LGPL 2.1	Java	Java	english, germany, french, italian, ukrainian, russian
NLTK	Apache 2.0	Python	Python	english, germany, french, italian, ukrainian, russian

FreeLing	Affero GPL	Lex, C++, C	Python	english, germany, french, italian, russian
Apache Solr	Apache 2.0	Java	Java, JavaScript, Python, Ruby, JSON	english, germany, french, italian, ukrainian, russian
GoogleCloud Natural Language API	Apache 2.0		Java, JavaScript, Python, Ruby, C++, C#, C	english, germany, french, italian, russian

Розглянемо детальніше деякі з них.

1.3.1 Monty Lingua

Monty Lingua – набір бібліотек і програм для символьної та статистичної обробки природньої мови для мов програмування Python і Java. Він збагачений знаннями про повсякденний світ з Open Mind Common Sense. З англійських речень він отримує кортежі суб'єкта/дієслова/об'єкта, а також прикметники, словосполучення з іменниками і дієсловами, імена людей, місця, події, дати і час та іншу семантичну інформацію.

Monty Lingua може виконувати над текстом наступні функції:

- MontyTokenizer – нормалізація розділових знаків, інтервалів та скороченнь;
- MontyTagger – тегування частин мови з використанням набору тегів Penn Treebank з проекту Open Mind Common Sense;
- MontyREChunker – розбиття тексту з тегами на дієслова, іменники та прикметники (VX, NX і AX відповідно);
- MontyExtractor – витягування структур дієслова-аргументу, фраз та іншої семантично-цінної інформації з речень і повернення реченнь у формі «дайджеста»;
- MontyLemmatiser – лематизація, чуттєва до частин мови, що

включає регулярні вислови з morph.lex Хамфріса і Кэрролла та корпус XTAG від UPENN;

– MontyNLGenerator – генерує анотації, речення поверхневої форми, визначає і нумерує дієслова NP і часів, враховуючи тип речення [28].

1.3.2 Natural Language Toolkit

Набір інструментів природної мови, або частіше NLTK, – це набір бібліотек і програм для символьної та статистичної обробки природної мови (NLP) для англійської мови, написаних мовою програмування Python. Його розробили Стівен Берд і Едвард Лопер з кафедри комп'ютерних та інформаційних наук університету Пенсільванії. NLTK містить як набори даних, так і графічні матеріали. До пакету входить книга, яка пояснює основні концепції завдань обробки мови, що підтримуються набором інструментів, а також прикладами застосування пакету.

NLTK призначений для підтримки досліджень і викладання навчальних курсів пов'язаних з NLP та близькоспорідненими областями, включаючи емпіричну лінгвістику, когнітивну науку, штучний інтелект, пошук інформації та машинне навчання. Даний набір бібліотек успішно використовується як навчальний інструмент, а також як платформа для створення прототипів і побудови дослідницьких систем. Він підтримує функціональні можливості класифікації, токенізації, стемінгу, тегів, аналізу та семантичного міркування [29].

В Natural Language Toolkit доступні такі функції:

– класифікація текстів;
– маркування частин мови;
– отримання сутності;
– токенізаація;
– парсинг;
– стемінг;

– семантичне судження.

Текстові та лексичні корпуси бібліотеки містять:

– корпус Penn Treebank;
– відкритий багатомовний Wordnet;
– корпус звітів проблем;
– тезаурус залежностей Ліна [30].

1.3.3 Tensorflow

TensorFlow – відкрита програмна бібліотека, розроблена компанією Google для машинного навчання. Зокрема, для вирішення задач створення та навчання нейронних мереж. Її кінцевою ціллю є автоматизація класифікації образів, що не поступається в якості людському сприйняттю. Система належить до другого покоління та здатна виконувати роботу на кількох паралельних процесорах, використовуючи архітектуру CUDA для підтримання загальних обрахунків на графічних процесорах.

Бібліотека використовується для автоматизованої анотації зображень та є базисом для системи RankBrain, яка збільшує релевантність ранжування пошукової видачі Google. А також добре підходить для аналізу та генерування природної мови [31].

Основою TensorFlow є обчислювальний граф, який складається з деякої множини вузлів. Вузлом представляється операція, кількість входів і виходів якої може коливатися вгору від нуля. А для посилань на такі операції створюється символічний дескриптор (тензор).

З точки зору математики тензор можна розуміти як деяке узагальнення (скалярів, векторів, матриць тощо). Конкретніше скаляр визначається як тензор нульового рангу, вектор – як тензор рангу 1, матрицю – як тензор другого рангу, а матриці, покладені встовпчик в третьому вимірі, – третього. Але в бібліотеці значення зберігаються в масивах [32].

2 ДОСЛІДЖЕННЯ ПРОГРАМ-АНАЛОГІВ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Дослідження програм-аналогів

Програми розпізнавання мовлення з подальшим його перетворенням у текст не така вже й рідкість та зустрічаються у багатьох формах. Вони можуть бути подані на окремих сайтах для онлайн користування, у вигляді ботів або ж самостійними додатками. Розглянемо деякі з них.

2.1.1 Чат-боти

Одним із варіантів вирішення питання розпізнавання мовлення є створення відповідних чат-ботів.

Чат-бот (chatbot) – це програма, що імітує реальну розмову з користувачем. Чат-боти дозволяють спілкуватися за допомогою текстових чи аудіо повідомлень на сайтах, в месенджерах, мобільних додатках чи по телефону [33].

Основними задачами чат-ботів є:

- автоматизація задач. В залежності від його функціонал і системи, чат-бот може самостійно опрацювати замовлення клієнта, зібрати його контактні дані, записати на демо-виклик, вислат статтю в базі знань, виставити рахунок на оплату послуг і т.д.;
- зворотній зв’язок з користувачем. Він забезпечує безперебійну роботу 24/7, без емоційного вигорання, та здатен зібрати відгуки клієнтів, надати вирішення проблеми чи додаткову інформацію;
- продажі. Це сучасне рішення, що розповість все про продукт, допоможе оплатити його чи запропонує супутній товар;
- інтеграція з великою кількістю CRM. Може бути встановлено зв’язок між помічником та інструментами для бухгалтерії та аналітики і т.п. [34].

Як правило, чат-боти використовують машинне навчання для виявлення

моделей спілкування. Завдяки постійній взаємодії з людьми вони навчаються наслідуванню справжніх розмов та реагують на усні чи письмові запити, допомагаючи знайти відповідь. Так як чат-боти використовують технології штучного інтелекту, вони здатні розуміти саме мову, а не просто команди. Але існує й інший метод їх функціонування – на основі запрограмованих сценаріїв із множинним вибором, наприклад, варіант А призводить до варіанту В [33].

Гарним прикладом втілення даної технології для розпізнавання українського мовлення є розробка під назвою Speech Recognition for Ukrainian. Даний чат-бот пропонує до використання одразу три двигуни на вибір:

- VOSK;
- DeepSpeech;
- Silero STT [35].

Але не дивлячись на широкий вибір варіантів розпізнавання, даний чат-бот має два суттєвих недоліки. Перший полягає в тому, що довжина голосового повідомлення, що може прийматись до обробки становить не більше однієї хвилини. Це робить технологію непридатною до використання у більшості випадків необхідності розпізнавання мовлення. Другий недолік – необхідність мережевого з'єднання достатньої якості, що зможе забезпечити оперативне функціонування месенджеру та обмін інформацією (голосовою, з боку користувача, та текстовою, з боку додатку) в режимі реального часу.

2.1.1.1 VOSK

VOSK – це автономний інструмент для розпізнавання мовлення з відкритим програмним кодом. Він дозволяє використовувати моделі для більш ніж 20 мов і діалектів. Моделі VOSK малі (50 Мб) та дозволяють перетворювати мовлення в текст «на льоту». Існують і більш точні моделі. Їх розмір досягає 2 Гб.

Реалізація бібліотеки можлива на багатьох мовах програмування, таких як Python, Java, NodeJS, C#, C++ та інші. А запуск можливий на на ОС Windows, Linux та Android [36].

Окрім того бібліотека має і ряд інших переваг:

1. Працює без доступу до мережі навіть на мобільних пристроях.
2. Для мови програмування Python встановлюється за допомогою всього лише однієї команди pip3 install vosk, без додаткових кроків.
3. Створена для потокової обробки мовлення, що дозволяє реалізувати миттєву реакцію на команди.
4. Дозволяє швидко налаштовувати словник розпізнавання для покращення точності розпізнавання.
5. Дозволяє ідентифікувати мовця [37].

2.1.1.2 DeepSpeech

DeepSpeech – двигун розпізнавання мовлення від компанії Mozilla, щореалізує однойменну архітектуру розпізнавання мовлення, запропоновану дослідниками з компанії Baidu. Реалізація написана на мові програмування Python з використанням платформи машинного навчання TensorFlow і поширюється під вільною ліцензією.

Підтримується робота в Linux, Android, macOS и Windows. Потужності достатньо для використання движка на платах LePotato, Raspberry Pi 3 і Raspberry Pi 4, а також на смартфонах Google Pixel 2, Sony Xperia Z Premium і Nokia 1.3. Для вбудування функції розпізнавання мовлення в програму запропоновані готові до застосування модулі для Python, NodeJS, C++ и .NET.

DeepSpeech складається з двох підсистем – акустичної моделі та декодувальника. Акустична модель використовує методи глибинного машинного навчання для обрахунку ймовірності наявності певних символів в звуці, що подається на вхід. Декодувальник застосовує алгоритм променевого пошуку для перетворення даних про ймовірність символів у текстове представлення. DeepSpeech значно простіша традиційних систем і при цьому забезпечує більш високу якість розпізнавання при наявності шумів. В розробці не використовуються традиційні акустичні моделі та концепція фонем, замість

них застосовується добре оптимізована система машинного навчання на основі нейронних мереж, котра дозволяє уникнути розробки окремих компонентів для моделювання різноманітних відхилень, таких як шум, ехо та особливості мовлення.

Зворотнім боком подібного підходу є те, що для отримання якісного розпізнавання та навчання нейронної мережі движок DeepSpeech потребує великого об'єму різнорідних даних, надиктованих в реальних умовах різними голосами та при наявності природніх шумів. Збиранням подібних даних займається створений в Mozilla проект Common Voice, що надає перевірений набір даних з великою кількістю годин різними мовами [38].

2.1.1.3 Silero STT

Silero STT – це кросплатформне перетворення мовлення в текст в реальному часі на пристроях за допомогою комерційної моделі машинного навчання від Silero. Для запуска моделі машинного навчання цей пакет потребує NatML. Особливості включають в себе:

- продуктивність Bare Metal. Прогноз перетворення мовлення в текст реалізований за допомогою NatML, котрий використовує переваги апаратних прискорювачів машинного навчання, таких як CoreML для iOS і macOS, NNAPI для Android та DirectML для Window;
- легкий у використанні. Предиктор претворення мовлення в текст приймає аудіо дані (AudioClip, NatDevice или любые аудиоданные PCM) та повертає виявлений текст у вигляді простого рядка;
- кросплатформність. Перетворювач мовлення в текст підтримує Android, iOS, macOS і Windows, тому користувач може протестувати його в редакторі, та розгорнути його на пристрої в рамках єдиного робочого процесу;
- полегшений пакет. Він містить сценарії прогнозування, тоді як модель машинного навчання буде завантажена під час виконання

з NatML Hub і кешована на пристрої, що значно зменшить розмір додатку [39].

2.1.2 CyberMova

Одним із перших пропонованих результатів пошуку Гугл на запит «розпізнавання української мови» є CyberMova – VoiceTypist. Це програма розпізнавання усного українського мовлення, яка встановлюється на ПК та працює без інтернет-з'єднання (що можна віднести до значних переваг).

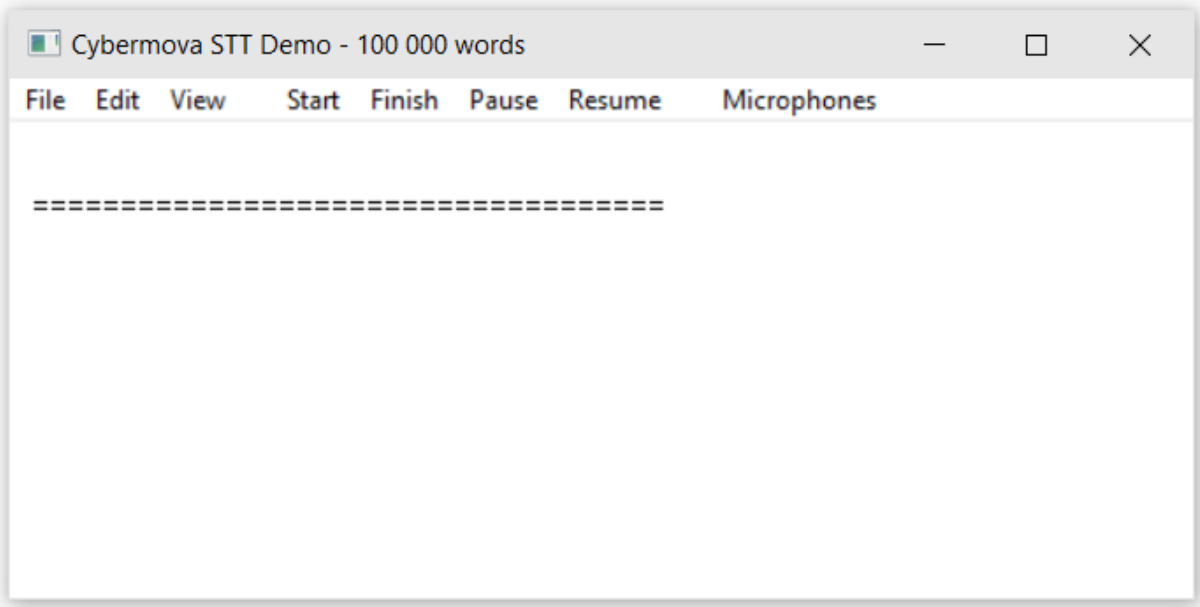


Рисунок 2.1 – Стартове вікно програми CyberMova

На рисунку 2.1 показано стартове вікно програми CyberMova, що з'являється після її інсталяції та запуску. Верхня панель меню пропонує наступні функціональні можливості:

- File: About – коротка інформація про додаток,
  Quit – завершення роботи програми;
- Edit – взаємодія з текстовою областю: Copy, Clear, Undo;
- View – зміна відображення тексту в текстовій області: Larger, Smaller, Normal, Autoscroll;

– Start – початок диктування для розпізнавання;
– Finish – завершення диктування;
– Pause – зупинка диктування;
– Resume – продовження диктування;
– Microphones – обрання мікрофону, за допомогою якого буде здійснюватись звукозапис.

Програма має словник розпізнавання, що складається з 100 тисяч слів, та здатна до розпізнавання таких розділових знаків, як «крапка», «кома», «знак питання» і «знак оклику» з їх подальшою заміною символами [40].

Тестування програми проводилося шляхом надиктовування тексту перших двох абзаців вступу даної роботи. Результат розпізнавання продемонстровано на рисунку 2.2.

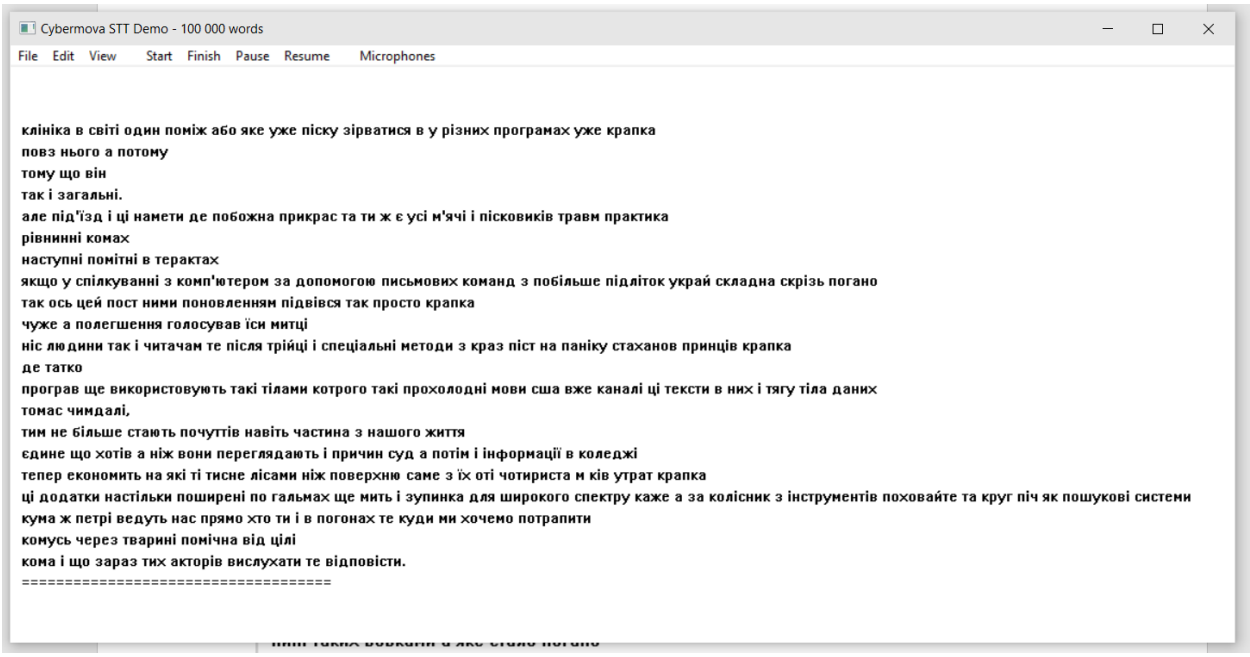


Рисунок 2.2 – Результат розпізнавання українського мовлення в програмі CyberMova

За допомогою онлайн-сервісу [41] було проведено порівняння надиктованого тексту з одержаним, результат якого наведено на рисунку 2.3.

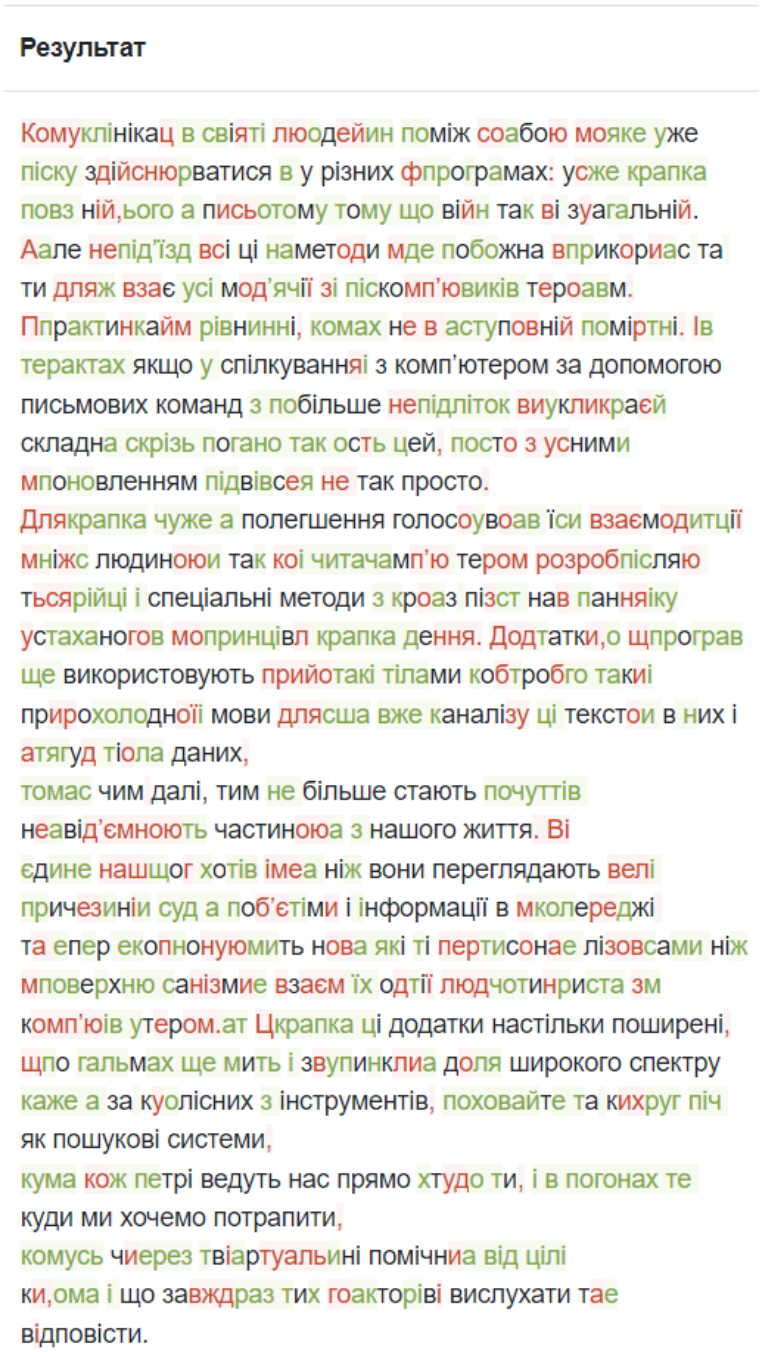


Рисунок 2.3 – Порівняння початкового тексту з одержаним із програми CyberMova

Хоча відсоток точності розпізнавання в даній програмі, розрахований на основі отриманих даних, складає 70,8%, але як видно з рисунка 2.2, реальна якість розпізнавання, що стосується більше інформаційного навантаження тексту, аніж співпадіння символів, є доволі низькою. А текст отриманий в результаті розпізнавання важко сприймається користувачем.

Також варто відзначити, що програма погано розпізнає розділові знаки.

Це видно навіть у першому рядку, де слово «крапка» так і залишилось словом, на відміну від останнього рядка, де його було замінено на відповідний символ.

2.1.3 Онлайн-ресурси для розпізнавання мовлення

Окрім десктопних додатків також існує велика кількість онлайн-ресурсів для розпізнавання мовлення, що мають ряд своїх переваг. Деякі застосунки розглянуті в таблиці 2.1 [6].

Таблиця 2.1 – Порівняння сервісів транскрибації

Назва	Особливість	Український інтерфейс	Вартість
Google Docs	Онлайн-сервіс, є застосунок	Так	Безкоштовно
Dictation.io	Онлайн-сервіс	Ні	Безкоштовно
oTranscribe	Онлайн-сервіс	Так	Безкоштовно
Transcribe by Wreally	Онлайн-сервіс	Ні	Від 20 доларів на рік, є тестовий період 7 днів
Субтитри YouTube	Онлайн-сервіс	Так	Безкоштовно
SpeechText.AI	Онлайн-сервіс	Так	До 10 хвилин запису – безкоштовно, далі тариф від 10 доларів
Google Перекладач	Онлайн-сервіс	Так	Безкоштовно
VoiceIn	Розширення для браузера	Ні	Платний тариф із розширеним функціоналом – 4 долари на місяць

SpeechTexter	Розширення для браузера або онлайн-сервіс	Так	Безкоштовно
Voice to text	Застосунок для Android	Ні	Безкоштовно

Деякі з цих програм підтримують велику кількість мов, можуть розставляти розділові знаки або працюють із аудіо файлами. Але вони мають і свої недоліки, такі як низька якість розпізнавання або тарифи на послуги.

Розглянемо детальніше більш доступний онлайн-сервіс [42] із широким спектром пропонованих можливостей (рисунок 2.4).

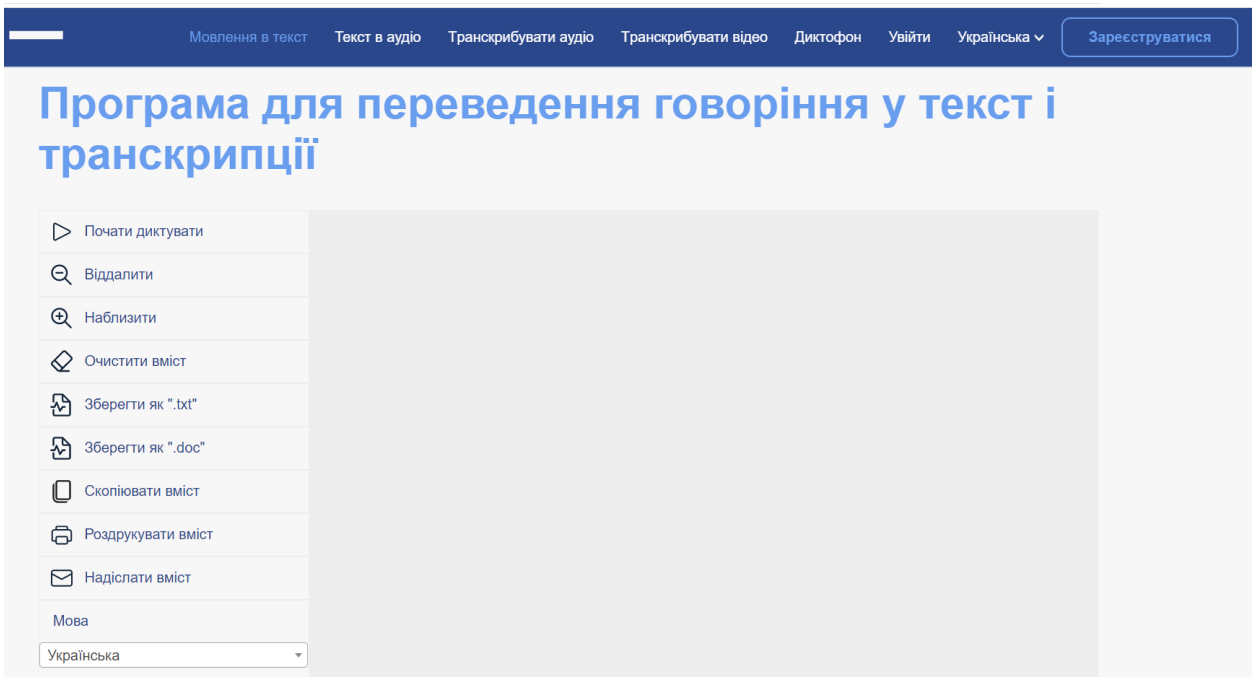


Рисунок 2.4 – Вигляд сайту для розпізнавання мовлення

У верхньому меню сайту можна обрати операцію, що буде виконуватися:

– перетворення мовлення в текст;
– озвучення тексту;
– транскрибація аудіо чи відео;

– запис аудіо файлу.

Варто зазначити, що послуги транскрибування доступні лише після реєстрації.

Обравши програму для переведення говоріння у текст бачимо деякі опції в боковому меню, а саме:

– кнопку початку диктування, що після натиснення змінюється на паузу;
– кнопки зміни розміру тексту в робочій області та видалення вмісту;
– кнопки збереження отриманого тексту в doc. та txt. форматах;
– кнопки копіювання, роздрукування та поширення отриманого тексту.

А також випадаючий список, у якому можна обрати мову розпізнавання. Окрім того, на сторінці наведені деякі теоретичні відомості, що стосуються використання онлайн-сервісу та розпізнавання мовлення в цілому. І зазначено, що програма здатна розпізнати такі розділові знаки як «крапка», «кома», «крапка з комою», «двокрапка», «тире/дефіс», «знак питання», «знак оклику», «відкрита дужка», «закрита дужка», «пропуск/пробіл», «новий рядок» та «новий абзац».

Тестування програми проводилося шляхом надиктовування тексту перших двох абзаців вступу даної роботи. Результат розпізнавання продемонстровано на рисунку 2.5.

За допомогою того ж онлайн-сервісу порівняємо розпізнаний текст з початковим (рисунок 2.6).

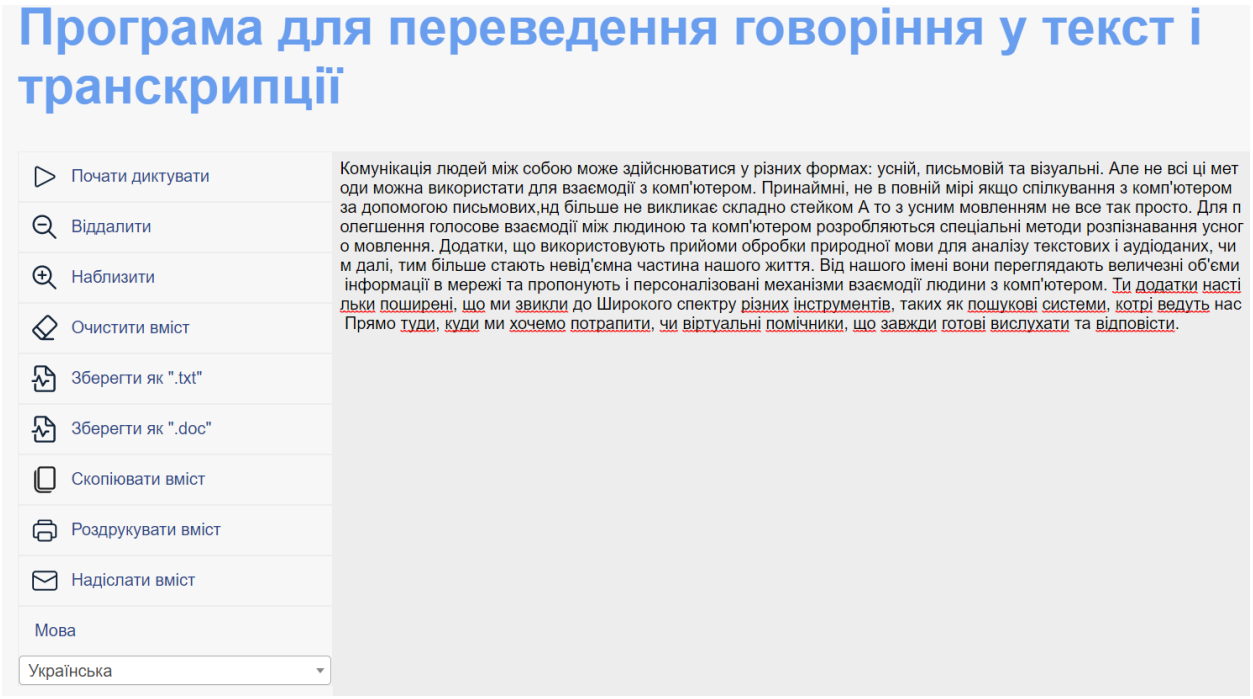


Рисунок 2.5 – Результат розпізнавання українського мовлення в онлайн-сервісі

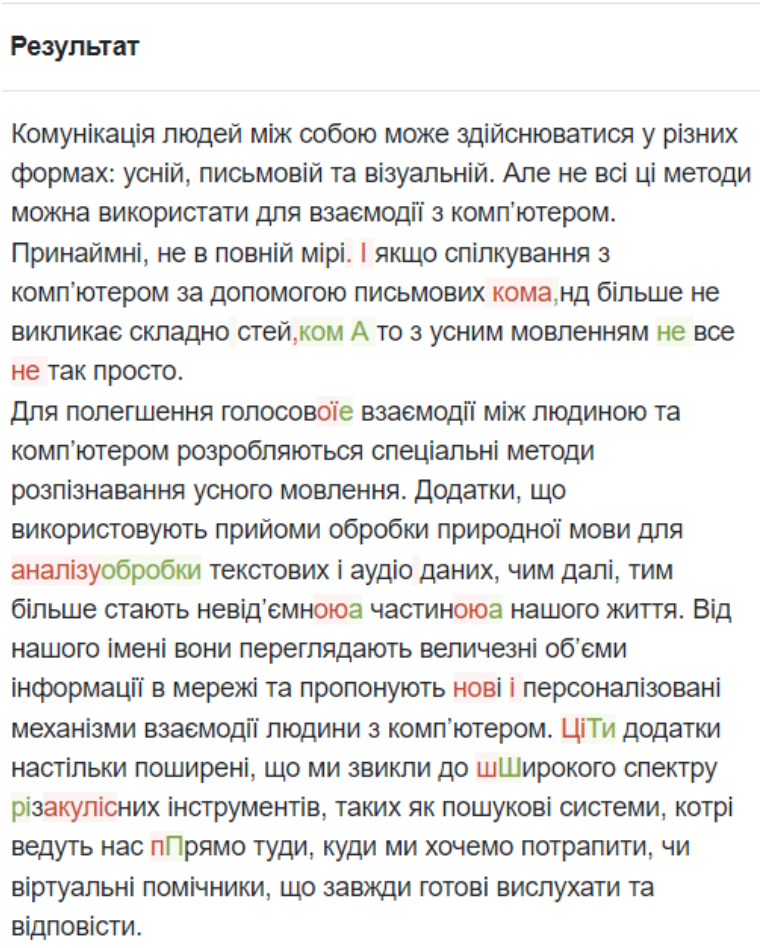


Рисунок 2.6 – Порівняння початкового тексту з одержаним з онлайн-сервісу

Як видно з рисунків, якість розпізнавання мовлення цією програмою значно вища, ніж у попередньої. За отриманими даними вона становить 96,3%. Текст, за невеликим виключенням, залишився інформативним та зручним для читання. Більшість розділових знаків були розпізнані та замінені на відповідні символьні позначення.

2.2 Визначення основних функцій програмного забезпечення

На основі розглянутих вище програм-аналогів сформулюємо основні функціональні вимоги до розроблюваного програмного продукту.

Основною задачею, що ставиться при розробці, є створення програмного інтерфейсу для перетворення звукової інформації в текст. Звук буде отримуватись з аудіо файлів, що можна або одразу завантажити з пам'яті комп'ютера, або спочатку провести запис, а вже потім подати на розпізнавання. Результат перетворення отриманого файлу в текст буде виводитись на екран, звідки його можна буде скопіювати для подальшої роботи.

Для більшої зручності використання звукозапису користувач зможе змінювати його налаштування. А для підвищення «дружелюбності» інтерфейсу буде створено розділ з довідковою інформацією.

2.3 Розробка UML-діаграми варіантів використання

Діаграма варіантів використання (use case diagram) – вихідна концептуальна модель процесів проектування та розробки системи. На ній позначаються відношення між зовнішніми агентами і варіантами використання. Ця діаграма є описом загальних функцій модельованої системи без розгляду її внутрішньої структури [43].

Діаграма прецедентів (ще одна назва) є графом. Її суть полягає в представленні системи у вигляді певної кількості акторів чи сутностей, а взаємодія яких із системою відбувається за допомогою «варіантів використання». Останні використовуються для опису послуг (деякого набору

дій), що система надає актору [44].

UML-діаграма варіантів використання, представлена на рисунку 2.7, пропонує до розгляду систему, що в ній єдиним актором є користувач.

Після запуску програми йому надається можливість виконання наступних дій:

– безпосереднє завантаження аудіофайлу та його розпізнавання;
– попередній запис необхідної інформації та збереження її у форматі звукового файлу (для подальшого використання у першому пункті);
– перегляд та зміна налаштувань системи розпізнавання;
– перегляд довідкової інформації у розділі допомоги користувачу.

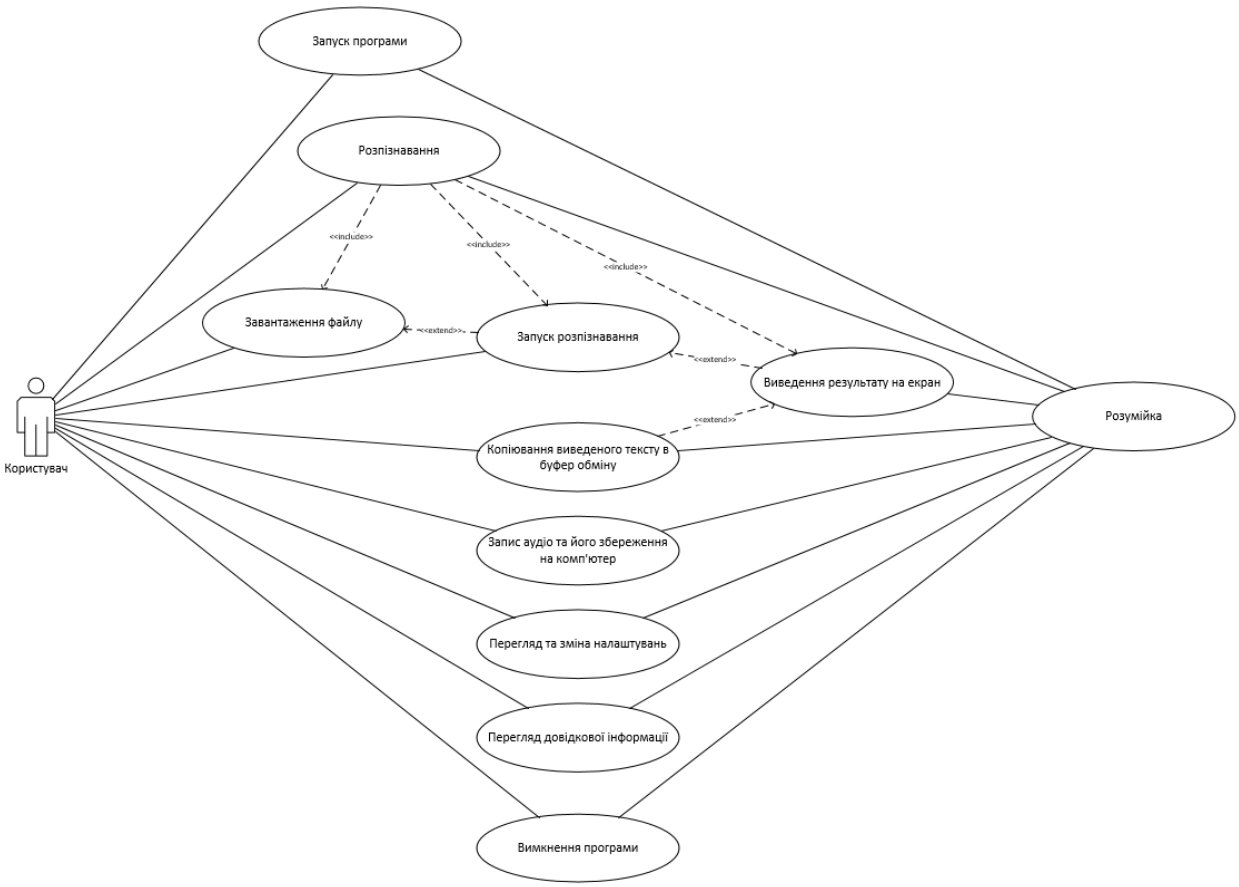


Рисунок 2.7 – UML-діаграма варіантів використання

2.4 Розробка UML-діаграми активності

Діаграма активності (activity diagram) моделює процес виконання операцій і деталізує їх. Особлива увага приділяється послідовності виконання процедур та елементарних операцій, що в результаті призводять до досягнення цілі [43].

Дія (action) – фундаментальна одиниця визначення поведінки в системі. Отримуючи вхідну множину сигналів, вона перетворює їх на вихідну. Варто зазначити, що обидві вони можуть бути порожні. Кожна дія зображена на діаграмі є ілюстрацією реальної дії користувача, що також є вірним і для діяльності в цілому. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати. Виконання деяких дій може бути чітко послідовним, або ж навпаки проводитись паралельно, в залежності від структури системи [45].

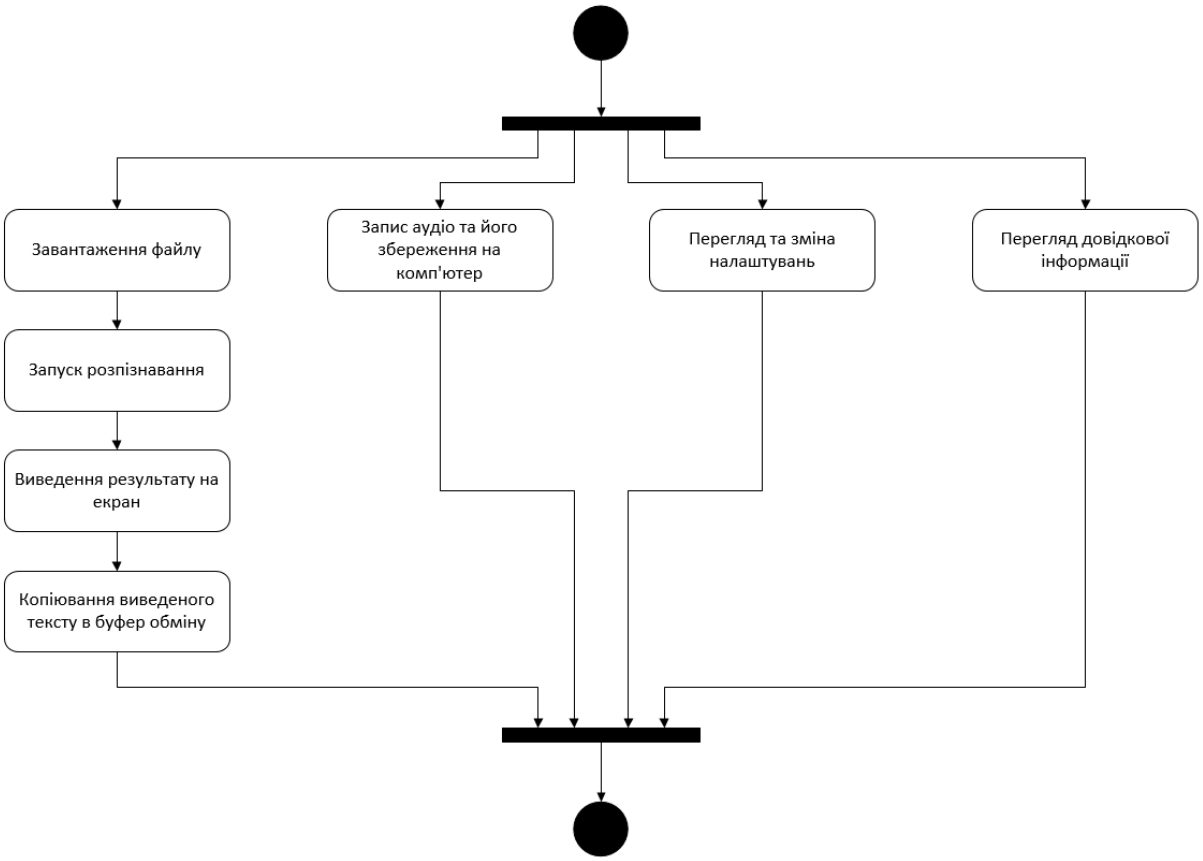


Рисунок 2.8 – UML-діаграма активності

На рисунку 2.8 зображено UML-діаграму активності системи. Подано перелік операцій, що деякі з них можуть виконуватися незалежно одна від одної, а деякі повинні йти у чітко визначеному порядку.

2.5 Розробка UML-діаграми розгортання

Діаграма розгортання (deployment diagram) – діаграма, на якій представлені вузли системи. Ця діаграма представляє загальну структуру і топологію системи та зображує розміщення компонентів в окремих вузлах системи. Крім того, діаграма розгортання демонструє наявність фізичних з'єднань – маршрутів передачі інформації між апаратними засобами, задіяними в реалізації системи. Вона містить графічні зображення процесорів, пристроїв, процесів і зв'язків між ними. Діаграма розгортання, на відміну від діаграм логічного подання, будується для системи в цілому (в єдиному екземплярі), оскільки відображає всі особливості її реалізації [43].

На UML-діаграмі розгортання програми (рисунок 2.9) представлено один із варіантів послідовного задіяння усіх елементів системи. Так користувач може спочатку ознайомитися з розділом «Допомога користувачеві» та отримати уявлення про програму та інструкцію до використання. Далі переглянути налаштування системи та записати власне аудіо, яке завантажить на розпізнавання та отримає текстову інформацію.

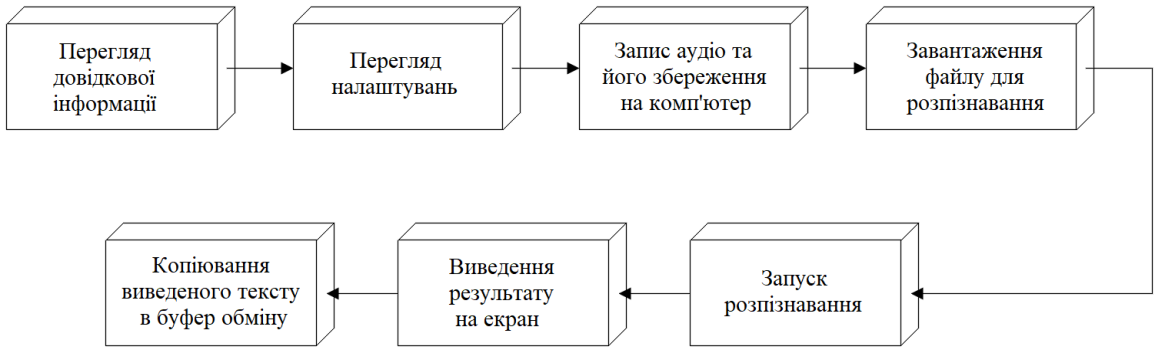


Рисунок 2.9 – UML-діаграма розгортання

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обгрунтування обраної мови програмування

Програму буде реалізовано на мові програмування Python.

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Наявність високорівневих структур даних із динамічними семантикою та зв’язуванням допомагає пришвидшити розробку програм, а підтримка модулів та їх пакетів дозволяє повторне використання коду. Окрім того, Python підтримує чотири парадигми програмування: об'єктно-орієнтовану, функціональну, процедурну та аспектно-орієнтовану.

Однією з переваг даної мови програмування є величезна кількість бібліотек різноманітного призначення. Так, тільки в стандартній бібліотеці є модулі роботи з мережевими протоколами, операційною системою, регулярними виразами, архівами, юніт-тестуванням, криптографічними протоколами, мультимедійними форматами та ін. [46].

Є бібліотеки і для обробки природної мови. Деякі з них вже було розглянуто в попередньому розділі. А в даній роботі використовуватимуться PyAudio та Speech Recognition API. Перша дозволяє отримання звукової інформації через мікрофон комп’ютера, а друга розпізнає мовлення та перетворює його в текст. За допомогою PyAudio програма зможе не тільки завантажувати готові аудіо файли для подальшого розпізнавання, але й дасть користувачу можливість створення необхідних звукових даних безпосередньо в процесі роботи [47].

Speech Recognition API створена для розпізнавання мовлення з підтримкою декількох движків і API. Для перетворення мовлення в текст потрібний лише клас Recognizer, що в залежності від базового API, що використовується має наступні методи:

- recognize_bing(): використовує Microsoft Bing Speech API;
- recognize_google(): використовує Google Speech API;

- recognize_google_cloud(): використовує Google Cloud Speech API;
- recognize_houndify(): використовує Houndify API від
- SoundHound;
- recognize_ibm(): використовує IBM Speech to Text API;
- recognize_sphinx(): використовує PocketSphinx API [48].

З усіх вище перелічених методів розпізнавання лише останній може працювати офлайн, в автономному режимі. Решта шість методів вимагають інтернет-з'єднання. Але recognize_sphinx() має суттєвий недолік для даної розробки – він підтримує лише чотири мови: англійську, французьку, китайську та італійську.

Тому для розпізнавання буде застосовано метод recognize_google(), що підтримує українську мову та є простим у користуванні. Для його безкоштовного використання, що накладає обмеження на довжину оброблюваного звукозапису, поданий користувачем аудіофайл буде розбитий на частини, що задовольнятимуть умовам використання методу.

Для взаємодії з системою буде використана бібліотека os. Цей модуль надає портативний спосіб використання функцій, що залежать від операційної системи. Він дає змогу:

- читати та записувати файли, за допомогою методу open();
- маніпулювати шляхами, за допомогою методу path();
- читати стрічки файлів в командному рядку, за допомогою методу fileinput();
- створювати тимчасові файли та каталоги, за допомогою методу tempfile();
- високорівневої обробки файлів і каталогів, за допомогою методу shutil().

Конструкція всіх вбудованих модулів Python, що залежать від операційної системи така, що поки доступні одні й ті ж функції, вони використовують один і той же інтерфейс; наприклад функція os.stat(path)

повертає статистичну інформацію про шлях в тому ж форматі, що з'явився в інтерфейсі POSIX.

Через модуль також доступні розширення, властиві тій чи іншій операційній системі os, але їх використання є загрозою перенесення. А всі функції, що приймають шлях чи імена файлів, приймають як байтові, так і рядкові об'єкти та приводять їх до об'єкту того ж типу, якщо повертається шлях чи ім'я файлу [49].

Бібліотека json – це вбудований пакет Python, який можна використовувати для взаємодії з файлами відповідного типу. Загалом, JSON – це синтаксис для зберігання та обміну даними. Цей модуль дозволяє перетворювати дані різних типів з формату json в Python і в зворотному напрямку (таблиця 3.1) [50].

Таблиця 3.1 – Перетворення даних з Python в JSON

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

Бібліотека shutil надає користувачеві набір вискорівневих функцій для роботи з файлами, групами файлів і папками. Зокрема, копіювати, переміщувати та видаляти їх [51].

А для роботи із аудіофайлами та звуком будуть підключені такі бібліотеки, як wave, pyaudio та pydub.

Варто також зазначити, що для створення комунікаційного інтерфейсу з користувачем використовуватиметься бібліотека PyQt5. Вона дозволяє застосовувати фреймворк Qt GUI в Python. Хоча сам Qt написаний на C++, ця бібліотека зберігає усю продуктивність додатку і в подібній реалізації.

PyQt5 – набір розширень графічного фреймворка Qt для мови програмування Python.

PyQt розроблений британською компанією Riverbank Computing і працює на всіх платформах, що підтримуються Qt: Linux та інші UNIX-подібні ОС, macOS і Windows. Існує 3 версії: PyQt6, PyQt5 и PyQt4, що підтримують відповідні версії Qt. PyQt поширюється під ліцензіями GPL (2 и 3 версії) та комерційною.

PyQt практично повністю реалізує можливості Qt. Це понад 600 класів, біліше 6000 функцій і методів, включаючи:

- існуючий набір віджетів графічного інтерфейсу;
- стилі віджетів;
- доступ до баз даних за допомогою SQL (MySQL, Oracle, PostgreSQL, ODBC,);
- QScintilla, заснований на Scintilla віджет текстового редактора;
- підтримку інтернаціоналізації (i18n);
- XML-парсер ;
- підтримку SVG;
- інтеграцію с WebKit, движком рендерінгу HTML;
- підтримку відтворення відео й аудіо.

PyQt також включає в себе Qt Designer (Qt Creator) – дизайнер графічного інтерфейсу користувача. Програма pyuic генерує Python код з файлів, створених в Qt Designer. Це робить PyQt дуже корисним інструментом для швидкого прототипування. Крім того, можна додавати нові графічні елементи управління, написані на Python, в Qt Designer.

Раніше PyQt поставлялася разом із середовищем розробки Eric, написаної на PyQt. Eric має вбудоване відлагоджування і може бути

використане для створення консольних програм. Тепер вона доступна в якості окремого проекту.

PyQt має кілька основних модулів:

- QtCore – основні не графічні класи: система сигналів і слотів, платформонезалежні абстракції для Unicode, потоків, роздільної пам'яті, регулярних виразів і т.д.;
- QtGui – компоненти графічного інтерфейсу (елементи управління), що базуються на візуальному представленні;
- QtNetwork – класи для мережевого програмування. Наприклад, клієнтів і серверів через UDP и TCP;
- QtOpenGL – класи, що дозволяють використовувати OpenGL і 3D-графіку в додатках PyQt.
- QtScript – класи, що дозволяють використовувати вбудований в Qt інтерпретатор JavaScript для управління додатком;
- QtSql – класи для інтеграції с базами даних за допомогою SQL;
- QtSvg – класи для відображення векторної графіки в форматі SVG;
- QtXml – класи, що реалізують обробку XML;
- uic – реалізація обробки XML-файлів, створених в Qt Designer, для генерації з них Python-коду графічного інтерфейсу [52].

3.2 Тестування програмного забезпечення

На рисунку 3.1 представлено початковий вигляд вікна програми, на якому можна бачити логотип програми та бокове меню.

Першим пунктом бокового меню є функція розпізнавання мовлення (рисунок 3.2). Щоб розпочати, необхідно обрати аудіофайл, з якого буде проводитись розпізнавання. Це можна зробити натиснувши кнопку «Обрати файл», а потім «Розпізнати текст» для запуску.

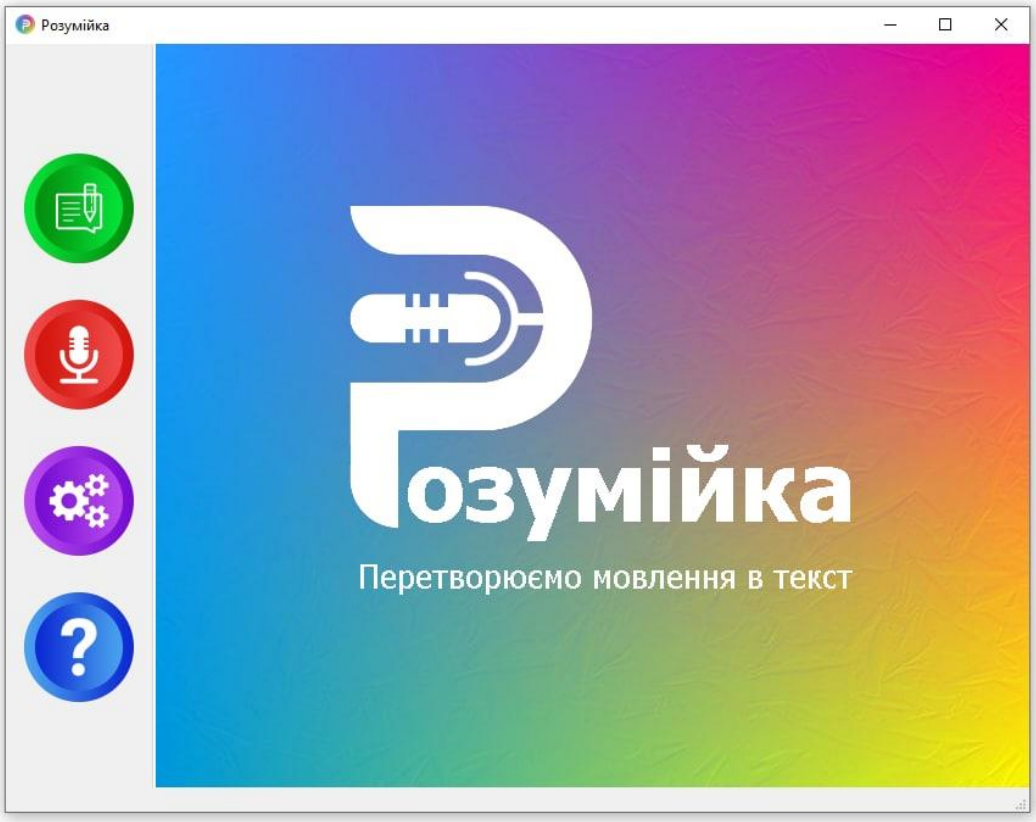


Рисунок 3.1 – Початковий вигляд програми

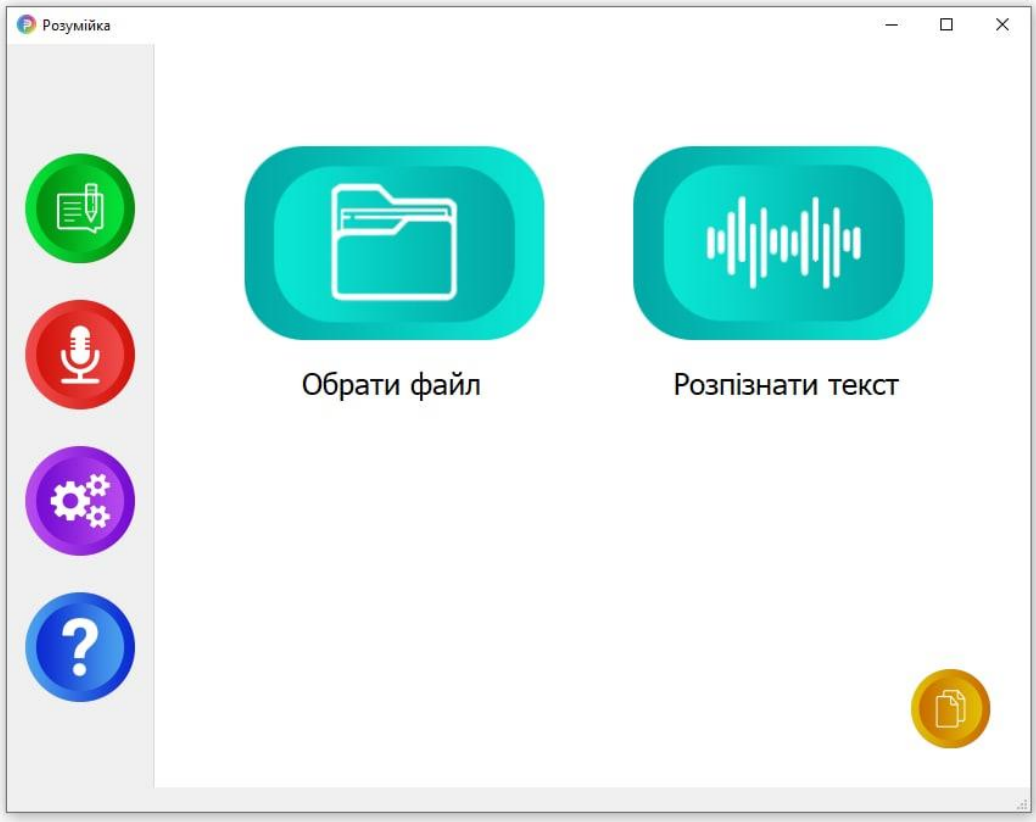


Рисунок 3.2 – Опція розпізнавання мовлення

Наступною на боковій панелі знаходиться функція запису аудіо (рисунок 3.3) та його автоматичне збереження у форматі .wav. Щоб почати запис необхідно натиснути на відповідне зображення. Завершення запису відбувається автоматично після кількох секунд паузи диктора.

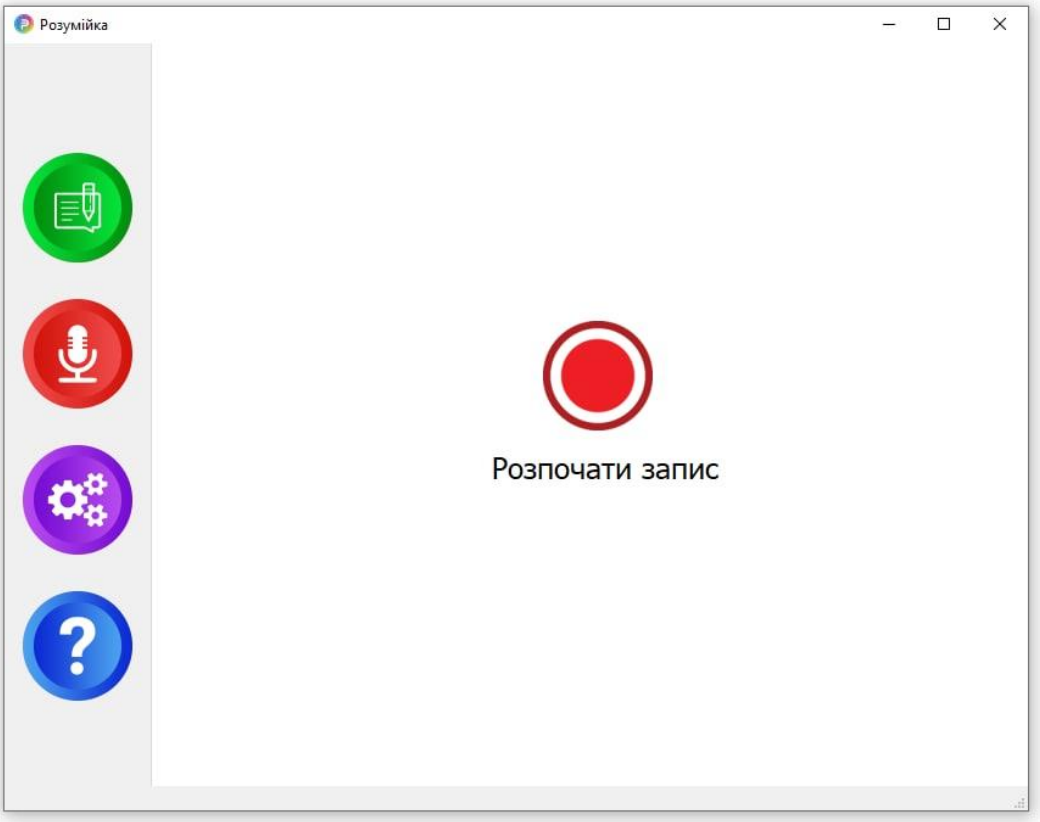


Рисунок 3.3 – Опція запису аудіофайлу

На рисунку 3.4 продемонстровано третій пункт бокового меню – можливість зміни налаштувань системи розпізнавання.

Редагуванню підлягають наступні параметри:

- Довжина сегменту запису (мс) – довжина частинок, в мілісекундах, на які програма розбиває запис;
- Мінімальна пауза – довжина паузи, в секундах, після якої завершується запис аудіо;
- Мова розпізнавання;

– Мінімальний час запису (с) – запис триватиме не менше вказаного числа секунд, навіть якщо користувач буде просто мовчати;
– Частота – частота звукового сигналу під час запису;
– Канал запису.

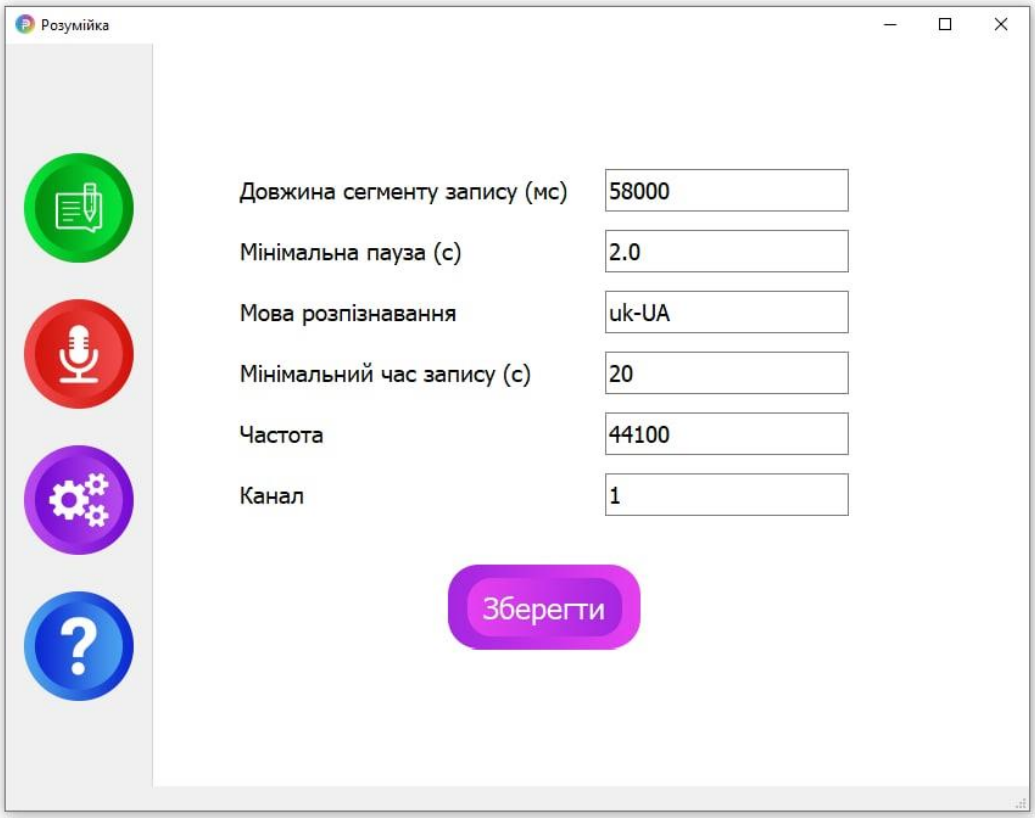


Рисунок 3.4 – Опція зміни налаштувань

На рисунку 3.5 подано останній пункт бокового меню – опцію допомоги користувачу, де наведено короткий опис програми та інструкцію до користування.

Варто зазначити, що програма є стійкою до виключних ситуацій. Так, якщо користувач натисне кнопку «Розпізнати текст», попередньо не завантаживши файл для розпізнавання до системи, з’явиться додаткове віконечко, що інформує про помилку (рисунок 3.8).

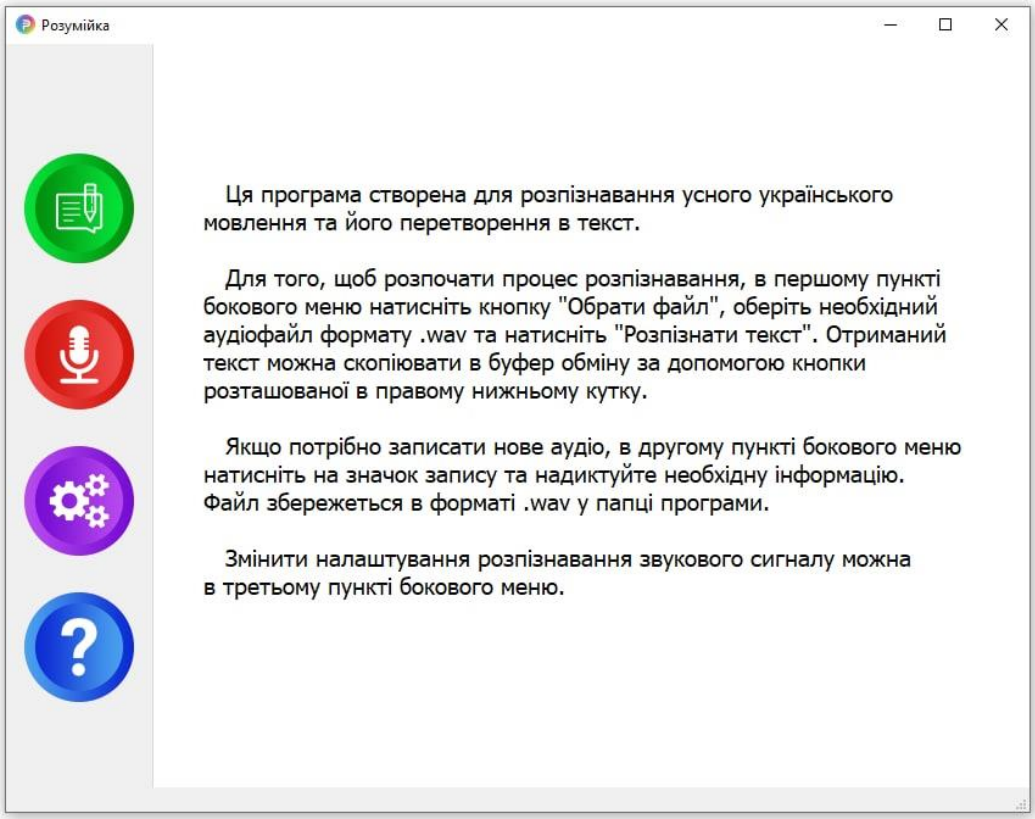


Рисунок 3.5 – Опція допомоги користувачеві

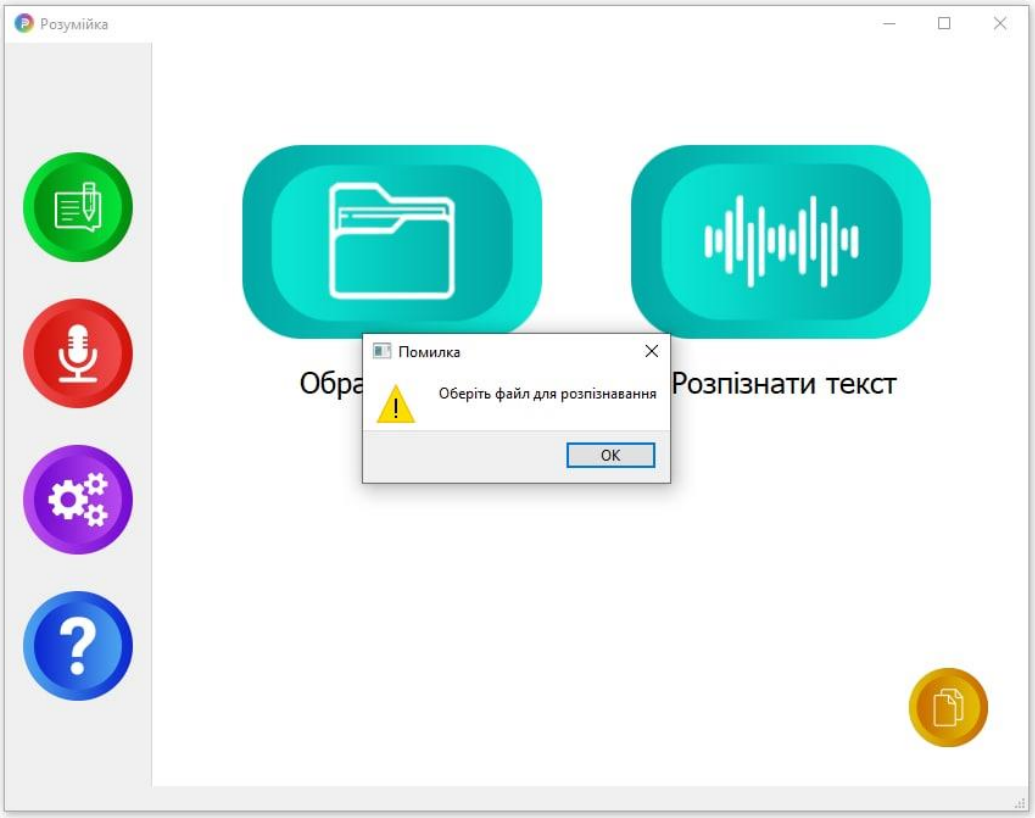


Рисунок 3.6 – Обробка виключення – не завантажено файл

Схоже віконечко з’явиться і якщо користувач введе некоректні дані під час зміни налаштувань параметрів розпізнавання (рисунок 3.7).

При введені коректних даних, після натискання кнопки «Зберегти» користувач побачить віконечко, що інформує про збереження змін (рисунок 3.8).

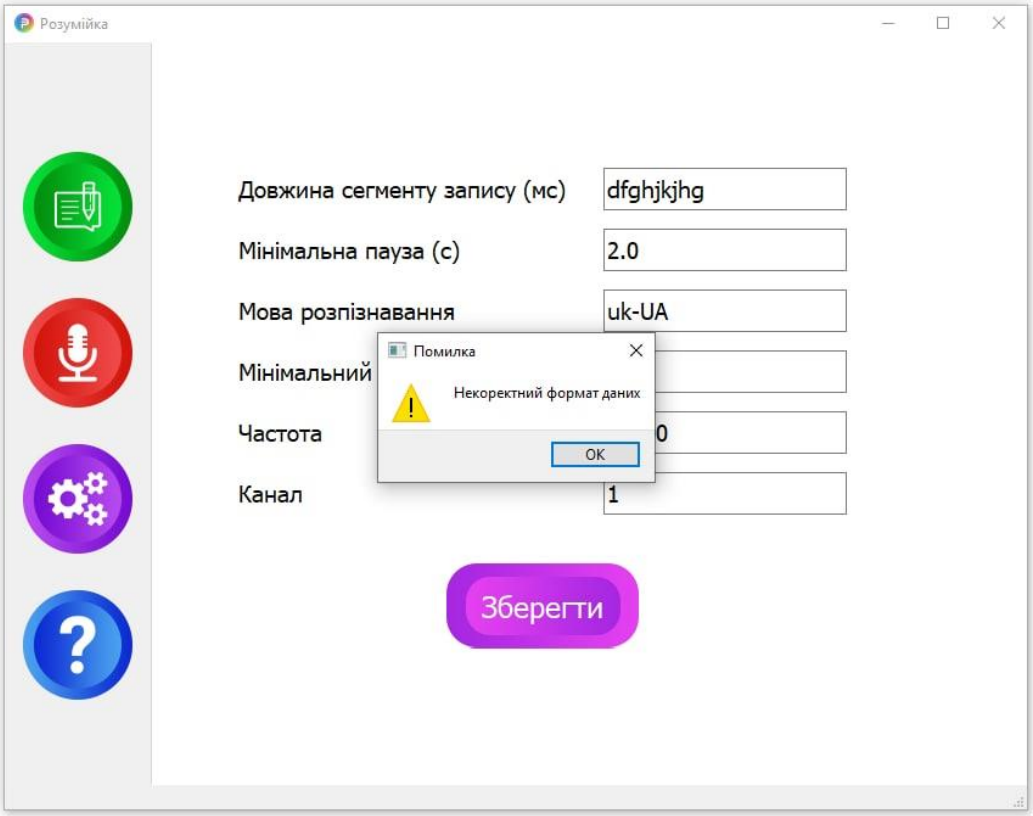


Рисунок 3.7 – Обробка виключення – введення некоректних даних

Проведення тестування саме системи розпізнавання варто почати із запису аудіо для розпізнавання. Обравши другий пункт бокового меню, натиснувши на кнопку запису та завершивши диктування, користувач побачить підпис «Запис збережено», що свідчить про збереження файлу на комп’ютер (рисунок 3.9).

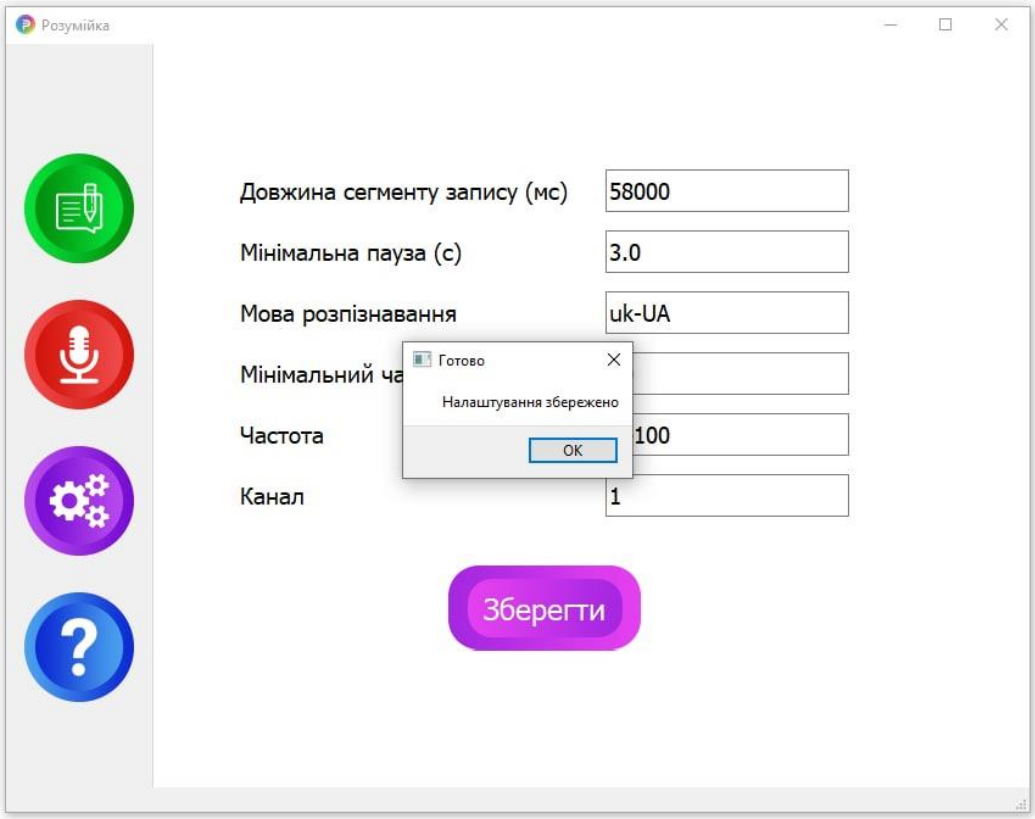


Рисунок 3.8 – Вдале збереження налаштувань користувача

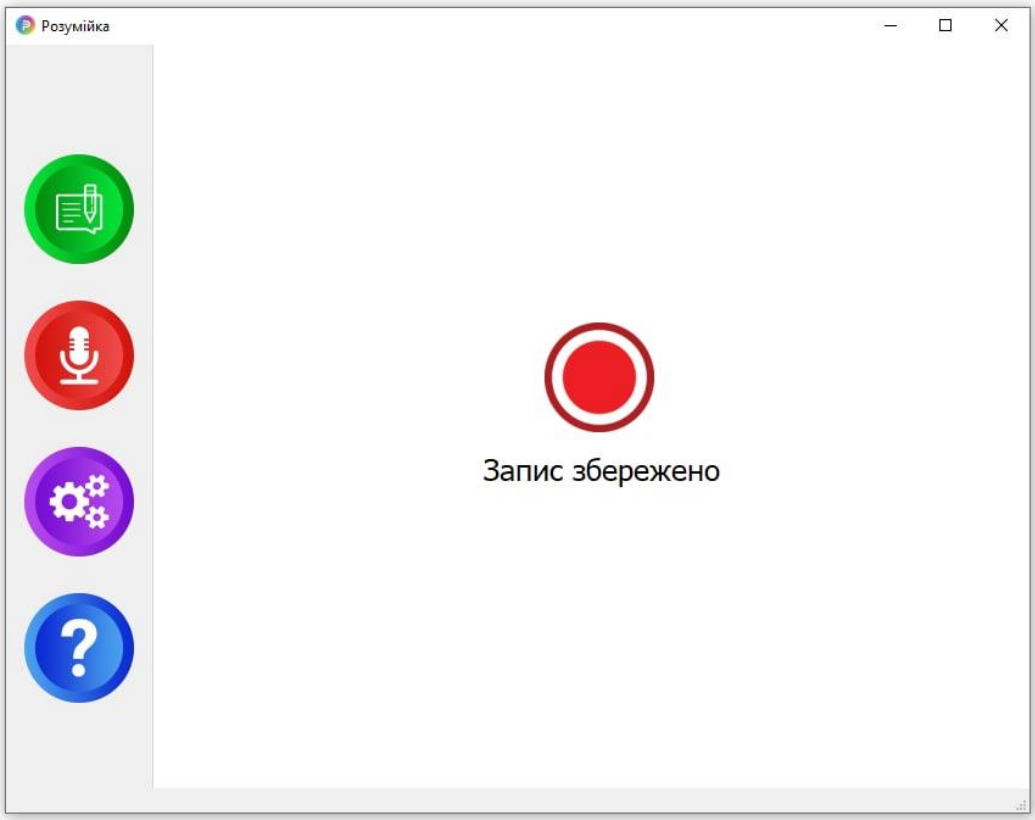


Рисунок 3.9 – Завершення запису аудіофайлу

Перейшовши у перший пункт меню та обравши щойно записаний файл за допомогою Select file, запустити розпізнавання за допомогою Run.

Для тестування програми було використано той самий текст, що і для програм-аналогів у першому розділі (рисунок 3.10).

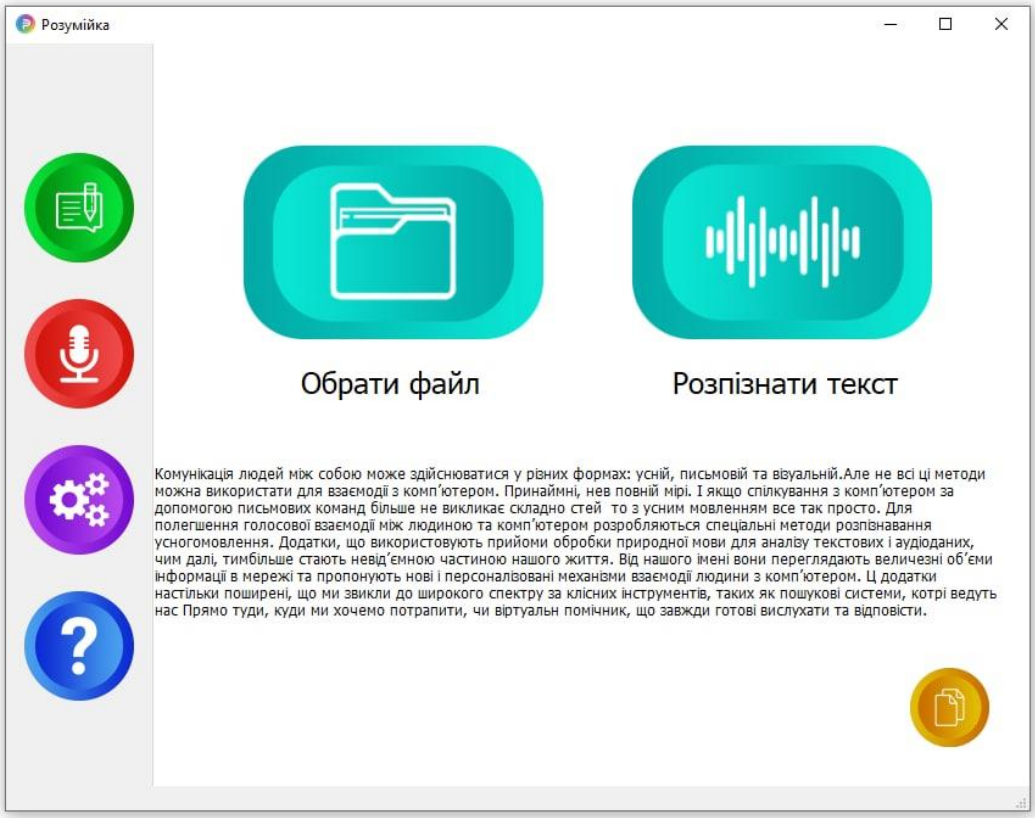


Рисунок 3.10 – Результат розпізнавання

Як видно з рисунка 3.10, програма вивела на екран надиктований текст, що їй вдалося розпізнати. Далі, за допомогою кнопки копіювання, що знаходиться у правому нижньому кутку програми, отриманий текст можна скопіювати для подальшого використання.

Для оцінки ефективності роботи програми, а точніше, точності її розпізнавання, за допомогою того ж онлайн-ресурса, що і в попередньому розділі, порівняємо одержаний текст з початковим (рисунок 3.11).

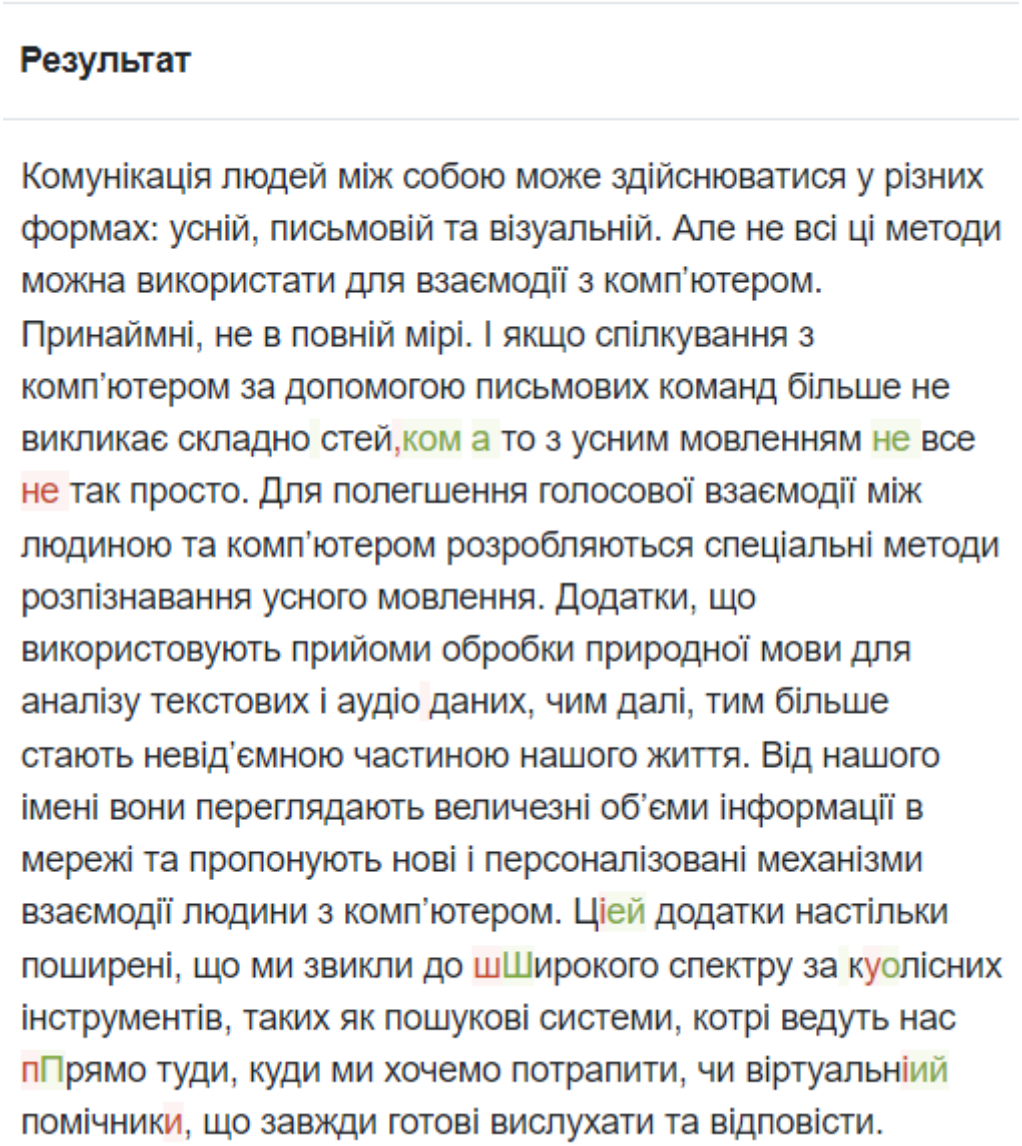


Рисунок 3.11 – Результат розпізнавання українського мовлення за допомогою програми Розумійка

З рисунків 3.10, 3.11 видно, що текст, отриманий в результаті розпізнавання зберігає послідовність викладення думки та інформаційне навантаження. Усі розділові знаки, що були розпізнані, вдало замінені на відповідні символьні позначення, а після «крапки» завжди слідує велика літера. Щоправда, кілька слів починаються з великої літери довільним чином. Таке явище може залежати від інтонації диктора, що зробив дещо більшу паузу, в результаті чого програма сприйняла це як початок нового речення.

Розрахунки точності, які були проведені на основі кількісного

співпадіння символів обох текстів показали, що якість розпізнавання за допомогою розробленого програмного забезпечення становить 99,2%, що на 3,1% перевищує найкращі результати існуючих програм-аналогів.

3.2.1 Оцінка ефективності роботи програми

В системах автоматичного розпізнавання мовлення основним показником є точність розпізнавання, що визначається як відсоток правильно розпізнаних слів (WRR – Word Recognition Rate) чи, навпаки, неправильно розпізнаних слів (WER – Word Error Rate). Інколи також використовується показник помилок розпізнавання фраз/речень (SER – Sentence Error Rate), який є важливим в діалогових системах, де коригування гіпотези розпізнавання неможливе на відміну від задачі диктування тексту.

Метод визначення показника WER складається з вирівнювання двох текстових рядків (перший – це результат розпізнавання, а другий – дійсний запис того, що було вимовлено) за допомогою алгоритму динамічного програмування з розрахунком відстані Левенштейна. Ця відстань являє собою «вартість» редагування даних (мінімальна кількість чи зважена сума операцій редагування) для перетворення першого рядка в другий з найменшим числом операцій ручної заміни (S), видалення (D) і вставки (I) слів:

$$WER = \frac{S+D+I}{T}, \quad WRR = 1 - WER, \tag{1}$$

де T – кількість слів в тексті, що розпізнається.

В синтетичних мовах для оцінки точності автоматичного розпізнавання мовлення можуть застосовуватися інші показники: помилки розпізнавання букв/символів, фонем (звуків мовлення), складів чи морфем. Окрім того, для деяких синтетичних мов, таких як українська, адекватним їх структурі показником є флективна помилка розпізнавання слів (IWER – Inflectional Word Error Rate), яка визначається наступним чином:

$$IWER = \frac{S_{hard} \cdot C_{hard} + S_{soft} \cdot C_{soft} + D + I}{T}, \quad (2)$$

$$C_{soft} < C_{hard}, C_{hard} \geq 1, C_{soft} < 1$$

Показник IWER приписує вагу C_{hard} всім неправильним замінам слів, які приводять до заміни лексеми слова, тобто до грубих помилок розпізнавання (S_{hard} – кількість помилок), і менша вага C_{soft} – усім негрубим помилкам в словах, де було невірно розпізнано закінчення словоформи, але основа слова розпізнана правильно (S_{soft} – кількість негрубих помилок) [53].

Для автоматизованої оцінки ефективності роботи використаного в розробленому програмному забезпеченні модуля розпізнавання було створено окрему програму для його тестування (https://colab.research.google.com/drive/1JAKhBg7sFh_fWLkqbB2qkjQNh3lIeUs2#scrollTo=qTsD7QI4ydm_).

Тестування було проведено з використанням тексту про історію ВНТУ [54], де присутні наукові слова, числа та власні назви. Обраний матеріал було надиктовано окремими реченнями та створено тестову вибірку, що містить двадцять елементів. Звукові файли разом із оригінальним текстом було завантажено в програму і здійснено їх початкову обробку.

Початковий текст було переведено у нижній регістр та пропущено через фільтрацію на предмет наявності невалідних символів. Після цього було здійснено його розбиття на речення.

Звукові файли було розпізнано за допомогою speech recognition та проведено над ними аналогічні маніпуляції (окрім розбиття на речення).

Порівняння отриманих текстових даних було виконано за допомогою вище описаних метрик, а саме WER і WRR. Хоча, як видно з формули 1, їх сума має становити одиницю, але дана залежність не завжди прослідковується на практиці. Так трапляється через те, що WER враховує не тільки пропуски слів, а також кількість видалень, кількість замін та кількість вставок. Варто відзначити, що було використано саме метрику WER, оскільки IWER складна для кодування.

Результат перевірки зображено на рисунку 3.12, а детальне порівняння початкового та отриманого текстів наведено в додатку В.

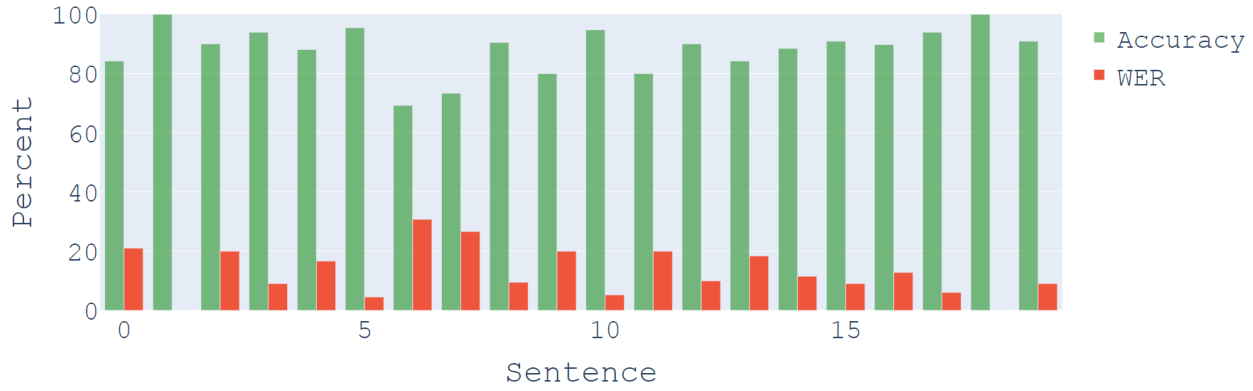


Рисунок 3.12 – Оцінка ефективності програми розпізнавання

Як видно з отриманих даних та рисунка, де Accuracy позначає точність розпізнавання, її значення не спускається нижче 69,23% за усі двадцять речень. А максимальне значення становить 100% і трапляється два рази. Середнє ж значення точності становить 88,38%, а середнє значення помилки – 13,03%. Їх залежність у сумі давати 100% майже збережено, що теж є гарним показником.

Варто зазначити, що при використанні метрики IWER, що краще обробляє особливості української мови, результати оцінки були б ще вищими.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення наукового аудиту науково-дослідної роботи

За описом показників ступеня новизни науково-дослідна робота належить до відносно нових робіт і має значення показника ступеня новизни з проміжку 10…40 – 30.

За показниками рівня теоретичного опрацювання науково-дослідна робота належить до розробки способу (алгоритму, програми), пристрою, отримання нової речовини та має значення показника рівня теоретичного опрацювання 20…60 – 45.

Показник наукового ефекту визначається за формулою:

$$\mathrm{E}_{\text{нау}} = 0{,}6 \cdot k_{\text{нов}} + 0{,}4 \cdot k_{\text{твор}} = 0{,}6 \cdot 30 + 0{,}4 \cdot 40 = 18 + 18 = 36$$

де $k_{\text{нов}}$, $k_{\text{твор}}$ – показники ступенів новизни та рівня теоретичного опрацювання науково-дослідної роботи;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

Значення показника наукового ефекту отримане при розрахунках становить 36 та належить до проміжку 15…49, що є достатнім рівнем. Такі результати обумовлені тим, що ступінь новизни розроблюваного в роботі продукту є не дуже високим, адже на даний момент існує значна кількість програм-аналогів, які, щоправда, не демонструють достатньої точності функціонування, що і зумовлює актуальність проведеного дослідження.

4.2 Проведення комерційного та технологічного аудиту науково-технічної розробки

Проведення комерційного та технологічного аудиту передбачає оцінювання науково-технічного рівня розробки за різними критеріями. Оцінювання проводилось за 12-ма критеріями та в межах оцінювання від 1 до

5 (таблиця 4.1).

Таблиця 4.1 – Результати оцінювання розробки

№	Критерії оцінювання	Експерт 1	Експерт 2	Експерт 3
1	Технічна здійсненність концепції	4	4	4
2	Наявність аналогів	3	3	3
3	Ціна продукту	3	4	4
4	Технічні властивості	4	4	3
5	Експлуатаційні витрати	4	4	4
6	Розміри ринку	3	3	3
7	Конкуренція	2	4	3
8	Наявність фахівців	4	4	3
9	Наявність фінансів	4	2	4
10	Необхідність нових матеріалів	4	4	4
11	Термін реалізації	4	4	4
12	Розробка документів	4	4	4
	Сума балів	43	44	43
	Середньоарифметична сума балів	43,3		

За результатами розрахунків оцінювання середній бал розробки становить 43,3, що належить до проміжку 41…48 та позначає високий науково-технічний рівень та комерційний потенціал розробки. Так, ефективність роботи розробленого програмного продукту є вищою за показники існуючих аналогів, оскільки використовує дещо інший підхід до розпізнавання мовлення, а терміни та вартість його впровадження є прийнятними.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

4.3.1 Витрати на оплату праці

Витрати на основну заробітну плату розробників $З_р$ розраховують

відповідно до посадових окладів працівників, за формулою:

$$З_р = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}$$

де k – загальна кількість розробників;

M_{ni} – місячний посадовий оклад конкретного розробника, грн;

t_i – кількість днів роботи конкретного розробника, дн.;

T_p – середня кількість робочих днів в місяці 22 дні.

Таблиця 4.2 – Витрати на заробітну плату розробників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	40 000	1 820,00	40	72 800,00
Програміст	35 000	1 590,00	30	47 700,00
Дизайнер	20 000	910, 00	30	27 300,00
Тестувальник	30 000	1 365,00	10	13 650,00
Всього				161 450,00

Оскільки в даній роботі розробляється саме програмний продукт, то розробник виконує одразу кілька зазначених посад, а саме програміста, дизайнера та тестувальника.

Додаткова заробітна плата розраховується як 10…12% від суми основної заробітної плати розробників за формулою:

$$З_{дод} = З_р \cdot \frac{Н_{дод}}{100\%} = 161\,450{,}00 \cdot \frac{12}{100\%} = 19\,374{,}00 \text{ (грн.)}$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату розробників розраховується як 22% від суми основної та додаткової заробітної плати розробників за формулою:

$$З_н = (З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} = (161\,450{,}00 + 19\,374{,}00\,) \cdot \frac{22}{100\%} = 39\,781{,}28 \text{ (грн.)}$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

4.3.3 Сировина, матеріали та комплектуючі

Оскільки в даній роботі розробляється саме програмний продукт, то витрати на сировину, матеріали та комплектуючі не передбачаються.

4.3.4 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні нарахування можуть бути розраховані за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12}$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Амортизаційні нарахування на нематеріальні ресурси розраховуються за формулою:

$$А_{н.р.} = Ц_{н.р.} \cdot Н_а \cdot \frac{t_{вик}}{12}$$

де $Н_а$ – норма амортизації, 15%.

Таблиця 4.3 – Амортизаційні нарахування по кожному виду обладнання

	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	32 000	2	1,8	2 400,00
Інвентар, меблі	25 000	4	1,8	937,50
Приміщення	780 000	20	1,8	5 850,00
Ліцензійні програмні засоби (Windows 10)	7 000	–	1,8	157,50
Всього				9 345,00

4.3.5 Витрати на електроенергію

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).

Витрати на силову електроенергію розраховуються за формулою:

$$В_{ел} = В \cdot П \cdot Ф \cdot К_{п} = 6{,}2 \cdot 0{,}7 \cdot 40 \cdot 8 \cdot 0{,}9 = 1\,249{,}92 \text{ (грн)}$$

де В – вартість 1 кВт-години електроенергії для 1 класу напруги, грн./кВт;

П – встановлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Кп – коефіцієнт використання потужності.

4.3.6 Інші 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 розробників за формулою:

$$В_і = З_р \cdot \frac{Н_{ів}}{100\%} = 161\,450{,}00 \cdot \frac{75}{100\%} = 121\,087{,}5 \text{ (грн.)}$$

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

4.3.7 Накладні (загальновиробничі) витрати

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 розробників за формулою:

$$В_{нзв} = З_р \cdot \frac{Н_{нзв}}{100\%} = 161\,450{,}00 \cdot \frac{125}{100\%} = 201\,812{,}5 \text{ (грн.)}$$

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

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_р + З_{дод} + З_н + А_{обл} + В_{ел} + В_і + В_{нзв} = 161\,450{,}00 +$$
$$+19\,374{,}00 + 39\,781{,}28 + 9\,345{,}00 + 1\,249{,}92 + 121\,087{,}5 +$$

$$+201\,812{,}5 = 554\,100{,}2 \text{ (грн.)}$$

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

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{554\,100{,}2}{0{,}5} = 1\,108\,200{,}4 \text{ (грн.)}$$

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.

4.4 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним процесом і часто базується на експертних оцінках, тому має вірогіднісний характер.

Комплексний показник КР рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_p = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = \frac{3^3 \cdot 2 \cdot 4}{554{,}1 \cdot 0{,}25} = 1{,}56$$

де I – коефіцієнт важливості роботи, I = 2...5;

n – коефіцієнт використання результатів роботи; n = 0, коли результати не будуть використовуватись; n =1, коли результати будуть використовуватись частково; n = 2, коли результати будуть використовуватись в дослідно-

конструкторських розробках; n = 3, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок;

T_C – коефіцієнт складності роботи, T_C = 1...3;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то R = 4 ; якщо результати роботи відповідають відомому рівню, то R = 3; якщо нижче відомих результатів, то R =1;

В – вартість науково-дослідної роботи, тис. грн;

t – час проведення дослідження, років.

Якщо $K_P > 1$, то науково-дослідну роботу можна вважати ефективною з високим науковим, технічним і економічним рівнями.

Як видно з розрахунків, комплексний показник рівня даної науково-дослідної роботи становить 1,56. Він є більшим одиниці, що свідчить про високу ефективність розробки.

4.5 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 4-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.

Тип даної розробки збігається з наступною ситуацією – 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:

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

- N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;
- $Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки;
- $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right)$$

де $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm \Delta Ц_о$;

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. Ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, ϑ =18%.

Припустимо, що ціна програмного продукту складає 4 000 грн. Розрахуємо можливе збільшення чистого прибутку за три роки. Зміну вартості продукту приймемо рівною 1 000 грн. До моменту впровадження результатів наукової розробки реалізації продукту не було.

$$\Delta П_1 = (1\,000 \cdot 0 + (4\,000 + 1\,000) \cdot 1\,000) \cdot 0{,}8333 \cdot 0{,}25 \cdot (1 - 0{,}18) =$$
$$= 5\,000 \cdot 1\,000 \cdot 0{,}208325 \cdot 0{,}82 = 854\,132{,}5 \text{ (грн.)}$$
$$\Delta П_2 = (1\,000 \cdot 0 + (4\,000 + 1\,000) \cdot 3\,000) \cdot 0{,}8333 \cdot 0{,}25 \cdot (1 - 0{,}18) =$$
$$= 5\,000 \cdot 3\,000 \cdot 0{,}208325 \cdot 0{,}82 = 2\,562\,397{,}5 \text{ (грн.)}$$
$$\Delta П_3 = (1\,000 \cdot 0 + (4\,000 + 1\,000) \cdot 5\,000) \cdot 0{,}8333 \cdot 0{,}25 \cdot (1 - 0{,}18) =$$
$$= 5\,000 \cdot 5\,000 \cdot 0{,}208325 \cdot 0{,}82 = 4\,270\,662{,}5 \text{ (грн.)}$$

Таким чином, можливе збільшення чистого прибутку від впровадження розроблюваного програмного продукту за три роки становить 7 687 192,5 грн.

Розрахунок приведеної вартості збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки здійснюється за формулою:

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

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

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

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{854\,132{,}5}{(1+0{,}1)^1} + \frac{2\,562\,397{,}5}{(1+0{,}1)^2} + \frac{4\,270\,662{,}5}{(1+0{,}1)^3} = \frac{854\,132{,}5}{1{,}1} +$$

$$+\frac{2\,562\,397{,}5}{1{,}21}+\frac{4\,270\,662{,}5}{1{,}331}=776\,484{,}09+2\,117\,683{,}88+$$
$$+3\,208\,611{,}95=6\,102\,779{,}92\ (\text{грн.})$$

Величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки можна розрахувати за формулою:

$$PV=k_{\text{інв}}\cdot \text{ЗВ}=2\cdot 1\,108\,200{,}4=2\,216\,400{,}8\ (\text{грн.})$$

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.

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:

$$E_{абс}=\text{ПП}-PV=6\,102\,779{,}92-2\,216\,400{,}8=3\,886\,379{,}12\ (\text{грн.})$$

Оскільки величина $E_{абс}$ має велике додатне значення, то це свідчить про потенційну зацікавленість інвесторів у впровадженні та комерціалізації цієї науково-технічної розробки.

Але для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність $E_в$ або показник внутрішньої норми 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.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 = \sqrt[3]{1 + \frac{3\,886\,379{,}12}{2\,216\,400{,}8}} - 1 = \sqrt[3]{1 + 1{,}75} - 1 =$$

$$= 1{,}4 - 1 = 0{,}4$$

де $T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

Далі визначають бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}12 + 0{,}05 = 0{,}17$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,9...0,12;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f = 0,05...0,5, але може бути і значно вищою.

Оскільки $E_в > \tau_{мін}$, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_{ок}$ (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у

впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} = \frac{1}{0,4} = 2,5 \text{ (роки)}$$

Оскільки $T_{ок}$ менше трьох років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено людино-машинний інтерфейс, що дозволяє розпізнавати усне українське мовлення та перетворювати його у текст. Даний програмний продукт пропонує до використання алгоритм розпізнавання мовлення від Google, що демонструє вищій показник точності, порівняно з програмами-аналогами.

У першому розділі загальної частини роботи розглянуто особливості розробки подібних інтерфейсів, методи обробки природної мови та засоби їх програмної реалізації.

В другому розділі досліджено існуючі програмні продукти розпізнавання мовлення та перетворення його у текст та обраховано їх показники точності. На основі отриманої інформації сформульовано основні функціональні вимоги до розроблюваного програмного забезпечення. А також виконано його проектування у вигляді трьох UML-діаграм (варіантів використання, активності, розгортання), що демонструють структурні особливості та доступні можливості продукту.

В третьому розділі магістерської кваліфікаційної роботи детально обґрунтовано обрану мову програмування та наведено результати тестування розробленого людино-машинного інтерфейсу. Окремо приділено увагу розрахункам ефективності системи, що подані двома різними підходами: мануального порівняння двох текстів (надиктованого та отриманого шляхом розпізнавання) і розрахунку точності на основі посимвольного співпадіння, та за допомогою окремого програмного коду, що реалізує оцінку за допомогою метрики WER. Отримані результати точності (99,2% першим способом та 88,38% – другим) є доволі високими. Варто зазначити, що тестування ефективності за допомогою метрики було здійнено на більш складних текстових даних, ніж у першому випадку. Також нижчій результат зумовлений тим, що метод розрахунку точності WER не враховує структурні особливості української мови, що означає більш високі результати при врахуванні цих особливостей.

В економічному розділ даної роботи представлено розрахунки собівартості розробки програмного продукту з урахуванням заробітної плати розробників, вартості обслуговування персональних комп’ютерів, на яких здійснюється розробка програмного коду, витрат на електроенергію амортизаційних та інших нарахувань тощо. Підтверджено високий науково-технічний рівень та комерційний потенціал розробки, і відповідно потенційну зацікавленість інвесторів у її впровадженні. А також визначено термін окупності, що становить 2,5 роки.

Таким чином можна стверджувати, що розробка людино-машинного інтерфейсу перетворенн «усне українське мовлення - текст» є актуальним питанням сучасності, демонструє високі показники точності розпізнавання та має прийнятні економічні характеристики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бенгфорт Бенджамин Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка / Бенджамин Бенгфорт, Ребекка Билбро, Тони Охеда. — СПб.: Питер, 2019. – 368 с.: ил. – (Серия «Бестселлеры O'Reilly»).
2. Людино-машинний інтерфейс : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Людино-машинний_інтерфейс
3. Людино-машинна взаємодія : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Людино-машинна_взаємодія
4. Інтерфейс : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Інтерфейс
5. Розпізнавання мовлення : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Розпізнавання_мовлення
6. Програми для транскрибації української мови – топ 10 : [Електронний ресурс]: Режим доступу: https://sendpulse.ua/blog/transcribe-audio-into-text
7. Распознавание речи : [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/Распознавание_речи
8. Методи і алгоритми розпізнавання та розуміння мови і мовлення : [Електронний ресурс]: Режим доступу: https://www.cybermova.com/speech/теорія-розпізнавання.html
9. Розпізнавання мови: етапи розвитку, сучасні технології і перспективи їх застосування / М.Ф. Бондаренко, А.В. Работягов, С.В. Щепковський // Біоніка інтелекту: наук.-техн. журнал. – 2010. – № 2 (73). – С. 164–168.
10. Технології і програми розпізнавання та розуміння мовлення : [Електронний ресурс]: Режим доступу: https://www.cybermova.com/speech/розпізнавання-мовлення.html
11. Обробка природної мови : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Обробка_природної_мови

12. Автореферат 601 Мельничук Іван Олегович : [Електронний ресурс]: Режим доступу: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/1686/1/Автореферат%20601%20Мельничук%20Іван%20Олегович.pdf
13. Обробка природної мови : [Електронний ресурс]: Режим доступу: https://www.wiki.uk-ua.nina.az/Обробка_природної_мови.html
14. Онищенко К. Аналіз Методів Обробки Природної Мови / Онищенко К., Данієль Я., Каменєв Р. – Інформаційні системи та технології ІСТ-2020 Секція 6. Програмна інженерія. : [Електронний ресурс]: Режим доступу: https://openarchive.nure.ua/server/api/core/bitstreams/a26452b9-d866-4aaa-aa04-524a9cae5f55/content
15. Морфологічна класифікація мов : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Морфологічна_класифікація_мов
16. Кореневі мови : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Кореневі_мови
17. Аглютинативні мови : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Аглютинативні_мови
18. Флективні мови : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Флективні_мови
19. Полісинтетичні мови : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Полісинтетичні_мови
20. J. Le. (2018). – «The 7 NLP Techniques That Will Change How You Communicate in the Future (Part I)» : [Електронний ресурс]: Режим доступу: https://heartbeat.fritz.ai/the-7-nlp-techniques-that-will-change-how-youcommunicate-in-the-future-part-i-f0114b2f0497
21. Як працює machine learning та його застосування на практиці: [Електронний ресурс]: Режим доступу: https://nachasi.com/tech/2019/01/31/yak-pratsyuye-machine-learning/
22. Машинне навчання: [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Машинне_навчання

23. Швець М. Ю. Магістерська дисертація: Машинне навчання на віртуальному ринку електроенергії: [Електронний ресурс]: Режим доступу: https://ela.kpi.ua/bitstream/123456789/30914/1/Shvets_magistr.pdf
24. M. Bates (1995). – «Models of natural language understanding» : [Електронний ресурс]: Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC40721/
25. Близнюк Б. О. Современные методы обработки естественного языка / О. Б. Близнюк, Л. В. Васильева, И. Д. Стрельникова, Д. С. Ткачук. – Вісник ХНУ ім. В.Н. Каразіна, 2017.
26. Jay Alammar The Illustrated Word2vec : [Електронний ресурс]: Режим доступу: https://jalammar.github.io/illustrated-word2vec/
27. Бібліотеки обробки природних мов у предметній області великих даних А.М. Луцків, Н.М. Попович, Х.Б. Юркевич : [Електронний ресурс]: Режим доступу: https://core.ac.uk/download/pdf/287919009.pdf
28. MontyLingua : [Електронний ресурс]: Режим доступу: https://en.wikipedia.org/wiki/MontyLingua
29. Natural Language Toolkit : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Natural_Language_Toolkit
30. Natural Language Processing tools and libraries : [Електронний ресурс]: Режим доступу: https://theappsolutions.com/blog/development/nlp-tools
31. TensorFlow : [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/TensorFlow
32. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. -СПб. : ООО "Диалектика", 2020. - 848 с.: ил. - Парал. тит. англ.
33. What is a Chatbot: Definition and Guide : [Електронний ресурс]: Режим доступу: https://sendpulse.com/support/glossary/chatbot

34. Чат-бот: что это такое, принцип работы и секреты использования : [Електронний ресурс]: Режим доступу: https://helpcrunch.com/blog/ru/chto-takoe-chat-bot/
35. Стартап за $500: як створили бот для розпізнавання української мови : [Електронний ресурс]: Режим доступу: https://www.redbull.com/ua-uk/speech-recognition-for-ukrainian
36. Офлайн распознавание речи. Библиотека Vosk : [Електронний ресурс]: Режим доступу: https://vc.ru/dev/247450-oflayn-raspoznavanie-rechi-biblioteka-vosk
37. Vosk is a speech recognition toolkit : [Електронний ресурс]: Режим доступу: https://alphacephei.com/vosk/index
38. Компания Mozilla представила движок распознавания речи DeepSpeech 0.9 : [Електронний ресурс]: Режим доступу: https://www.opennet.ru/opennews/art.shtml?num=54053#:~:text=Компания%20Mozilla%20представила%20движок%20распознавания%20речи%20DeepSpeech%200.9,-09.11.2020%2011
39. Silero STT - Realtime Speech-to-Text : [Електронний ресурс]: Режим доступу: https://assetsale.herokuapp.com/cn/contents/84285
40. Розпізнавання мовлення – введення тексту голосом українською мовою (VoiceTypist) : [Електронний ресурс]: Режим доступу: https://www.cybermova.com/products/stt-demo.htm
41. Сравнение двух текстов онлайн : [Електронний ресурс]: Режим доступу: https://pr-cy.ru/difference/
42. Програма для переведення говоріння у текст і транскрипції : [Електронний ресурс]: Режим доступу: https://www.textfromtospeech.com/uk/voice-to-text/
43. Дубовой В. М. Моделювання процесів і систем керування : навчальний посібник / Дубовой В. М., Москвіна С.М., Никитенко О.Д. – Вінниця : ВНТУ, 2009. – 103 с.

44. Діаграма прецедентів : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_прецедентів
45. Діаграма діяльності : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_діяльності
46. Python : [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Python
47. Easy Speech-to-Text with Python : [Електронний ресурс]: Режим доступу: https://towardsdatascience.com/easy-speech-to-text-with-python-3df0d973b426
48. Введение в распознавание речи с Python : [Електронний ресурс]: Режим доступу: https://dev-gang.ru/article/vvedenie-v-raspoznavanie-reczi-s-python-uxr050lia2
49. os – Miscellaneous operating system interfaces : [Електронний ресурс]: Режим доступу: https://docs.python.org/3/library/os.html
50. Python JSON : [Електронний ресурс]: Режим доступу: https://www.w3schools.com/python/python_json.asp
51. Модуль shutil : [Електронний ресурс]: Режим доступу: https://pythonworld.ru/moduli/modul-shutil.html
52. PyQt : [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/PyQt
53. Карпов А. А. Методология оценивания работы системы автоматического распознавания речи, А. А. Карпов, И. С. Кипяткова // Известия вузов. Приборостроение. – 2012. – Т. 55, №11. – С. 38-43. : [Електронний ресурс]: Режим доступу: http://pribor.ifmo.ru/file/article/5994.pdf
54. Історія університету : [Електронний ресурс]: Режим доступу: https://vntu.edu.ua/uk/about-university/history.html
55. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

56. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Моделювання сценаріїв розвитку інфокомунікаційного процесу в бездротовому централізованому мережевому кластері», Вісник ВПІ, № 6, 2021; с. 100-113. DOI: https://doi.org/10.31649/1997-9266-2021-159-6-100-113.

57. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Визначення параметричного простору показників для оцінювання доступності інфокомунікаційного процесу в бездротовому централізованому мережевому кластері», Вісник ВПІ, № 1, 2022; с. 50-64. DOI: https://doi.org/10.31649/1997-9266-2022-160-1-50-64.

58. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Дослідження бездротового централізованого мережевого кластера з реалізацією сеансів інфокомунікаційної взаємодії в незалежних віртуальних сегментах», Вісник ВПІ, № 2, 2022; с. 68-80. https://doi.org/10.31649/1997-9266-2022-161-2-68-80.

59. Данильчук О.М., Ковтун В.В., Никитенко О.Д., Нестюк Ю.Ю., Присяжнюк В.В., «Елементи методології прецизійного фонетичного аналізу фонограм усного мовлення», Вісник ВПІ, № 2, 2022; с. 36-51. https://doi.org/10.31649/1997-9266-2022-162-3-36-51.

ДОДАТКИ

Додаток А
(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Людино-машинний інтерфейс перетворення "усне українське мовлення – текст"»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 80,4% Схожість 19,6%

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені 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 автора. 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 _____ Галущак А.В.
(підпис) (прізвище, ініціали)

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

Автор роботи _____ Нестюк Ю.Ю.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Ковтун В.В.
(підпис) (прізвище, ініціали)

Додаток Б
(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____ В'ячеслав Ковтун

"30" жовтня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС ПЕРЕТВОРЕННЯ «УСНЕ УКРАЇНСЬКЕ МОВЛЕННЯ – ТЕКСТ»

08-01.МКР.005.00.000 ТЗ

Студентка групи 2АКІТ-21м

_____ Юлія Нестюк

Керівник д.т.н., доцент, зав. кафедри КСУ

_____ В'ячеслав Ковтун

Вінниця 2022

1. Назва та галузь застосування

1.1. Назва – Людино-машинний інтерфейс перетворення «усне українське мовлення – текст».

1.2. Галузь застосування – Комп'ютеризовані системи обробки усного мовлення.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ № 66 від 24. 03. 2022 р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення якості автоматичного перетворення усного мовлення в текст.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Бенгфорт Бенджамин Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка / Бенджамин Бенгфорт, Ребекка Билбро, Тони Охеда. — СПб.: Питер, 2019. — 368 с.: ил. — (Серия «Бестселлеры O'Reilly»).
2. Розпізнавання мови: етапи розвитку, сучасні технології і перспективи їх застосування / М.Ф. Бондаренко, А.В. Работягов, С.В. Щепковський // Біоніка інтелекту: наук.-техн. журнал. – 2010. – № 2 (73). – С. 164–168.
3. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. -СПб. : ООО "Диалектика", 2020. - 848 с.: ил. - Парал. тит. англ.
4. Карпов А. А. Методология оценивания работы системы автоматического распознавания речи, А. А. Карпов, И. С. Кипяткова // Известия вузов. Приборостроение. – 2012. – Т. 55, №11. – С. 38-43. : [Електронний ресурс]: Режим доступу: http://pribor.ifmo.ru/file/article/5994.pdf

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- запис звукового файлу;
- завантаження звукового файлу в програму;
- розпізнавання мовлення з отриманого аудіо та перетворення його у текст;
- виведення отриманого тексту на екран;
- можливість копіювання виведеного тексту.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:
- WINDOWS 10;
- Python 3.6.

5.2.2. Умови експлуатації системи:
- робота на стандартних ПЕОМ в приміщеннях зі стандартними умовами.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:
1. Аналіз методів обробки природної мови та огляд варіантів їх програмної реалізації «15» жовтня 2022 р.
2. Дослідження програм-аналогів та визначення основних функцій розроблюваного програмного забезпечення «27» жовтня 2022 р.
3. Проектування структури програмного забезпечення системи «01» листопада 2022 р.
4. Розробка програмного забезпечення системи «20» листопада 2022 р.
5. Розрахунок економічних характеристик розробки «04» грудня 2022 р.

6.2 Графічні матеріали:
- Розробка UML-діаграм системи «04» листопада 2022 р.
- Тестування програмного забезпечення «28» листопада 2022 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28» листопада 2022 р.

7.2. Атестація здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «12» грудня 2022 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «20» грудня 2022 р.

Додаток В
(довідковий)

Лістинг програми

Модуль розробки інтерфейсу

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import QtCore, QtGui, QtWidgets
from loguru import logger
from tools_2 import Params

logger.add('debug.log', format="{time} {level} {message}", level="DEBUG", rotation="1 weeks",
compression='zip')
logger.info('RUN')

class Ui_MainWindow(object):

    def __init__(self):
        self.buffer = ""
        self.path_to_file = ""
        self.config = Params()

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(840, 630)
        MainWindow.setWindowIcon(QtGui.QIcon("design/icons/Logo.png"))
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.btn_menu_logo = QtWidgets.QPushButton(self.centralwidget)
        self.btn_menu_logo.setGeometry(QtCore.QRect(0, 20, 120, 40))
        self.btn_menu_logo.setText("")
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("design/icons/bg.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_menu_logo.setIcon(icon)
        self.btn_menu_logo.setIconSize(QtCore.QSize(120, 40))
        self.btn_menu_logo.setStyleSheet("background-color: rgb(240, 240, 240);")
        self.btn_menu_logo.setObjectName("btn_main_page")
        self.btn_menu_speech = QtWidgets.QPushButton(self.centralwidget)
        self.btn_menu_speech.setGeometry(QtCore.QRect(0, 90, 120, 90))
        self.btn_menu_speech.setText("")
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("design/icons/writing.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_menu_speech.setIcon(icon)
        self.btn_menu_speech.setIconSize(QtCore.QSize(90, 90))
        self.btn_menu_speech.setStyleSheet("border-radius: 40; \n"
                                           "cursor: pointer;")
        self.btn_menu_speech.setObjectName("btn_menu_speech")
        self.btn_menu_record = QtWidgets.QPushButton(self.centralwidget)
        self.btn_menu_record.setGeometry(QtCore.QRect(0, 210, 120, 90))
        self.btn_menu_record.setText("")
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("design/icons/recording.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_menu_record.setIcon(icon1)
        self.btn_menu_record.setIconSize(QtCore.QSize(90, 90))
        self.btn_menu_record.setStyleSheet("border-radius: 40; \n"
                                           "cursor: pointer;")
        self.btn_menu_record.setObjectName("btn_menu_record")
        self.btn_menu_setting = QtWidgets.QPushButton(self.centralwidget)
```

```
        self.btn_menu_setting.setGeometry(QtCore.QRect(0, 330, 120, 90))
        self.btn_menu_setting.setText("")
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap("design/icons/settings.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_menu_setting.setIcon(icon2)
        self.btn_menu_setting.setIconSize(QtCore.QSize(90, 90))
        self.btn_menu_setting.setStyleSheet("border-radius: 40; \n"
                                            "cursor: pointer;")
        self.btn_menu_setting.setObjectName("btn_menu_setting")
        self.btn_menu_help = QtWidgets.QPushButton(self.centralwidget)
        self.btn_menu_help.setGeometry(QtCore.QRect(0, 450, 120, 90))
        self.btn_menu_help.setText("")
        icon5 = QtGui.QIcon()
        icon5.addPixmap(QtGui.QPixmap("design/icons/help.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_menu_help.setIcon(icon5)
        self.btn_menu_help.setIconSize(QtCore.QSize(90, 90))
        self.btn_menu_help.setStyleSheet("border-radius: 40; \n"
                                         "cursor: pointer;")
        self.btn_menu_help.setObjectName("btn_menu_help")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(120, -30, 730, 700))
        font = QtGui.QFont()
        font.setPointSize(9)
        self.tabWidget.setFont(font)
        self.tabWidget.setIconSize(QtCore.QSize(50, 50))
        self.tabWidget.setTabBarAutoHide(True)
        self.tabWidget.setObjectName("tabWidget")
        self.widget_logo = QtWidgets.QWidget()
        self.widget_logo.setGeometry(QtCore.QRect(120, 0, 730, 700))
        self.widget_logo.setStyleSheet("background-image: url(design/icons/mainPage.png);")
        self.widget_logo.setObjectName("widget_logo")
        self.tabWidget.addTab(self.widget_logo, "")
        self.widget_speech = QtWidgets.QWidget()
        self.widget_speech.setObjectName("widget_speech")
        self.btn_select_file = QtWidgets.QPushButton(self.widget_speech)
        self.btn_select_file.setGeometry(QtCore.QRect(70, 90, 250, 160))
        self.btn_select_file.setText("")
        icon10 = QtGui.QIcon()
        icon10.addPixmap(QtGui.QPixmap("design/icons/select.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_select_file.setIcon(icon10)
        self.btn_select_file.setIconSize(QtCore.QSize(250, 160))
        self.btn_select_file.setStyleSheet("border-radius: 20;")
        self.btn_select_file.setObjectName("btn_select_file")
        self.label_10 = QtWidgets.QLabel(self.widget_speech)
        self.label_10.setGeometry(QtCore.QRect(120, 270, 150, 30))
        self.label_10.setObjectName("label_10")
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(60)
        self.label_10.setFont(font)
        self.btn_run_speech = QtWidgets.QPushButton(self.widget_speech)
        self.btn_run_speech.setGeometry(QtCore.QRect(390, 90, 250, 160))
        self.btn_run_speech.setText("")
        icon11 = QtGui.QIcon()
        icon11.addPixmap(QtGui.QPixmap("design/icons/recognize.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_run_speech.setIcon(icon11)
        self.btn_run_speech.setIconSize(QtCore.QSize(250, 160))
        self.btn_run_speech.setStyleSheet("border-radius: 20;")
        self.btn_run_speech.setObjectName("btn_run_speech")
        self.label_11 = QtWidgets.QLabel(self.widget_speech)
        self.label_11.setGeometry(QtCore.QRect(425, 270, 200, 30))
        self.label_11.setObjectName("label_11")
        font = QtGui.QFont()
        font.setPointSize(18)
```

```
        font.setBold(True)
        font.setWeight(60)
        self.label_11.setFont(font)
        self.btn_copy = QtWidgets.QPushButton(self.widget_speech)
        self.btn_copy.setGeometry(QtCore.QRect(620, 520, 65, 65))
        self.btn_copy.setText("")
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap("design/icons/copy.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_copy.setIcon(icon3)
        self.btn_copy.setIconSize(QtCore.QSize(65, 65))
        self.btn_copy.setStyleSheet("border-radius: 40;")
        self.btn_copy.setObjectName("btn_copy")
        self.label = QtWidgets.QLabel(self.widget_speech)
        self.label.setGeometry(QtCore.QRect(0, 250, 691, 331))
        self.label.setText("")
        self.label.setWordWrap(True)
        self.label.setObjectName("label")
        self.label.raise_()
        self.btn_select_file.raise_()
        self.btn_run_speech.raise_()
        self.btn_copy.raise_()
        self.tabWidget.addTab(self.widget_speech, "")
        self.widget_record = QtWidgets.QWidget()
        self.widget_record.setObjectName("widget_record")
        self.btn_run_record = QtWidgets.QPushButton(self.widget_record)
        self.btn_run_record.setGeometry(QtCore.QRect(320, 235, 90, 90))
        self.btn_run_record.setText("")
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap("design/icons/record.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_run_record.setIcon(icon4)
        self.btn_run_record.setIconSize(QtCore.QSize(90, 90))
        self.btn_run_record.setStyleSheet("border-radius: 40;")
        self.btn_run_record.setObjectName("btn_run_record")
        self.text_under_btn_record = QtWidgets.QLabel(self.widget_record)
        self.text_under_btn_record.setGeometry(QtCore.QRect(270, 325, 200, 60))
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(60)
        self.text_under_btn_record.setFont(font)
        self.text_under_btn_record.setObjectName("text_under_btn_record")
        self.tabWidget.addTab(self.widget_record, "")
        self.widget_setting = QtWidgets.QWidget()
        self.widget_setting.setObjectName("widget_setting")
        settings_font = QtGui.QFont()
        settings_font.setPointSize(14)
        settings_font.setWeight(60)
        self.label_2 = QtWidgets.QLabel(self.widget_setting)
        self.label_2.setGeometry(QtCore.QRect(70, 110, 280, 35))
        self.label_2.setFont(settings_font)
        self.label_2.setObjectName("label_2")
        self.line_setting_chunk_len_ms = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_chunk_len_ms.setGeometry(QtCore.QRect(370, 110, 200, 35))
        self.line_setting_chunk_len_ms.setFont(settings_font)
        self.line_setting_chunk_len_ms.setObjectName("line_setting_chunk_len_ms")
        self.label_3 = QtWidgets.QLabel(self.widget_setting)
        self.label_3.setGeometry(QtCore.QRect(70, 160, 280, 35))
        self.label_3.setFont(settings_font)
        self.label_3.setObjectName("label_3")
        self.line_setting_pause_threshold = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_pause_threshold.setGeometry(QtCore.QRect(370, 160, 200, 35))
        self.line_setting_pause_threshold.setFont(settings_font)
        self.line_setting_pause_threshold.setObjectName("line_setting_pause_threshold")
        self.label_4 = QtWidgets.QLabel(self.widget_setting)
        self.label_4.setGeometry(QtCore.QRect(70, 210, 280, 35))
        self.label_4.setFont(settings_font)
        self.label_4.setObjectName("label_4")
```

```
        self.line_setting_language = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_language.setGeometry(QtCore.QRect(370, 210, 200, 35))
        self.line_setting_language.setFont(settings_font)
        self.line_setting_language.setObjectName("line_setting_language")
        self.label_5 = QtWidgets.QLabel(self.widget_setting)
        self.label_5.setGeometry(QtCore.QRect(70, 260, 280, 35))
        self.label_5.setFont(settings_font)
        self.label_5.setObjectName("label_5")
        self.line_setting_record_sec = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_record_sec.setGeometry(QtCore.QRect(370, 260, 200, 35))
        self.line_setting_record_sec.setFont(settings_font)
        self.line_setting_record_sec.setObjectName("line_setting_record_sec")
        self.label_6 = QtWidgets.QLabel(self.widget_setting)
        self.label_6.setGeometry(QtCore.QRect(70, 310, 280, 35))
        self.label_6.setFont(settings_font)
        self.label_6.setObjectName("label_6")
        self.line_setting_rate = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_rate.setGeometry(QtCore.QRect(370, 310, 200, 35))
        self.line_setting_rate.setFont(settings_font)
        self.line_setting_rate.setObjectName("line_setting_rate")
        self.label_7 = QtWidgets.QLabel(self.widget_setting)
        self.label_7.setGeometry(QtCore.QRect(70, 360, 280, 35))
        self.label_7.setFont(settings_font)
        self.label_7.setObjectName("label_7")
        self.line_setting_channel = QtWidgets.QLineEdit(self.widget_setting)
        self.line_setting_channel.setGeometry(QtCore.QRect(370, 360, 200, 35))
        self.line_setting_channel.setFont(settings_font)
        self.line_setting_channel.setObjectName("line_setting_channel")
        self.btn_save_setting = QtWidgets.QPushButton(self.widget_setting)
        self.btn_save_setting.setGeometry(QtCore.QRect(240, 435, 160, 70))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.btn_save_setting.setFont(font)
        self.btn_save_setting.setText("Зберегти")
        self.btn_save_setting.setStyleSheet("color: rgb(255, 255, 255);\n"
                                            "background-image: url(design/icons/save.png);\n"
                                            "pady: 150;\n"
                                            "padx = 200;\n"
                                            "border-radius: 40;")
        self.btn_save_setting.setObjectName("btn_save_setting")
        self.tabWidget.addTab(self.widget_setting, "")
        self.widget_help = QtWidgets.QWidget()
        self.widget_help.setObjectName("widget_help")
        self.tabWidget.addTab(self.widget_help, "")
        self.information_text = QtWidgets.QLabel(self.widget_help)
        self.information_text.setGeometry(QtCore.QRect(40, 90, 635, 400))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.information_text.setFont(font)
        self.information_text.setObjectName("information_text")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 840, 22))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionAbout = QtWidgets.QAction(MainWindow)
        self.actionAbout.setObjectName("actionAbout")
        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        self.Handel_Buttons()

    def Handel_Buttons(self):
        self.btn_menu_logo.clicked.connect(self.Open_logo_tab)
        self.btn_menu_speech.clicked.connect(self.Open_speech_tab)
        self.btn_menu_record.clicked.connect(self.Open_record_tab)
```

```
        self.btn_menu_setting.clicked.connect(self.Open_setting_tab)
        self.btn_menu_help.clicked.connect(self.Open_help_tab)
        self.btn_select_file.clicked.connect(self.Select_file)
        self.btn_copy.clicked.connect(self.Copy_on_buffer)
        self.btn_run_speech.clicked.connect(self.Run_extract_text)
        self.btn_save_setting.clicked.connect(self.Save_settings)
        self.btn_run_record.clicked.connect(self.Run_record)

    def Open_logo_tab(self):
        self.tabWidget.setCurrentIndex(0)

    def Open_speech_tab(self):
        self.tabWidget.setCurrentIndex(1)

    def Open_record_tab(self):
        self.tabWidget.setCurrentIndex(2)

    def Open_setting_tab(self):
        self.tabWidget.setCurrentIndex(3)

    def Open_help_tab(self):
        self.tabWidget.setCurrentIndex(4)

    def Run_extract_text(self):
        if self.path_to_file:
            text = self.config.get_text_with_speech(pth=self.path_to_file)
            self.Outout_result(text)
        else:
            logger.error('User didn\'t select file')
            error = QMessageBox()
            error.setWindowTitle("Помилка")
            error.setText("Оберіть файл для розпізнавання")
            error.setIcon(QMessageBox.Warning)
            error.setStandardButtons(QMessageBox.Ok)
            error.setDefaultButton(QMessageBox.Cancel)
            error.exec_()

    def Select_file(self):
        logger.info("Load file")
        self.text_under_btn_record.setText(" Розпочати запис")
        qfd = QFileDialog()
        path = ""
        filter = "wav(*.wav)"
        filename = QFileDialog.getOpenFileName(qfd, "", path, filter)
        if filename[0]:
            self.path_to_file = filename[0]
            logger.info(f"Path => {filename[0]}")
        else:
            logger.error("No file selected")

    def Outout_result(self, res):
        self.label.setText(res)
        self.buffer = res
        f = open('text.txt', 'w')
        f.write(res)
        f.close()

    def Copy_on_buffer(self):
        logger.info('Copy_on_buffer')
        cb = QApplication.clipboard()
        cb.clear(mode=cb.Clipboard)
        cb.setText(self.buffer, mode=cb.Clipboard)

    def Save_settings(self):
        try:
            config = {
                "split_dir": "tmp",
                "chunk_length_ms": int(self.line_setting_chunk_len_ms.text()),
                "pause_threshold": float(self.line_setting_pause_threshold.text()),
```

```
                "selected_language": self.line_setting_language.text(),
                "RECORD_SECONDS": int(self.line_setting_record_sec.text()),
                "RATE": int(self.line_setting_rate.text()),
                "CHANNELS": int(self.line_setting_channel.text())
            }
            self.config.save_params(config)
            self.config = Params()
            logger.info("Save")
            done = QMessageBox()
            done.setWindowTitle("Готово")
            done.setText("Налаштування збережено")
            done.setStandardButtons(QMessageBox.Ok)
            done.exec_()

        except ValueError:
            logger.error("Invalid input format")
            error = QMessageBox()
            error.setWindowTitle("Помилка")
            error.setText("Некоректний формат даних")
            error.setIcon(QMessageBox.Warning)
            # error.setStandardButtons(QMessageBox.Ok|QMessageBox.Cancel)
            error.setStandardButtons(QMessageBox.Ok)
            error.setDefaultButton(QMessageBox.Cancel)
            #error.setDetailedText("Check if the parameters are correct")
            error.exec_()

    def Run_record(self):
        self.text_under_btn_record.setText("Rec. in progress")
        self.config.rec()
        self.text_under_btn_record.setText("Запис збережено")

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Розумійка"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.widget_logo),
_translate("MainWindow", "Tab 0"))
        self.label_10.setText(_translate("MainWindow", "Обрати файл"))
        self.label_11.setText(_translate("MainWindow", "Розпізнати текст"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.widget_speech),
_translate("MainWindow", "Tab 1"))
        self.text_under_btn_record.setText(_translate("MainWindow", " Розпочати запис"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.widget_record),
_translate("MainWindow", "Tab 2"))
        self.label_2.setText(_translate("MainWindow", "Довжина сегменту запису (мс)"))
        self.line_setting_chunk_len_ms.setText(_translate("MainWindow", "58000"))
        self.label_3.setText(_translate("MainWindow", "Мінімальна пауза (с)"))
        self.line_setting_pause_threshold.setText(_translate("MainWindow", "2.0"))
        self.label_4.setText(_translate("MainWindow", "Мова розпізнавання"))
        self.line_setting_language.setText(_translate("MainWindow", "uk-UA"))
        self.label_5.setText(_translate("MainWindow", "Мінімальний час запису (с)"))
        self.line_setting_record_sec.setText(_translate("MainWindow", "20"))
        self.label_6.setText(_translate("MainWindow", "Частота"))
        self.line_setting_rate.setText(_translate("MainWindow", "44100"))
        self.label_7.setText(_translate("MainWindow", "Канал"))
        self.line_setting_channel.setText(_translate("MainWindow", "1"))
        #self.btn_save_setting.setText(_translate("MainWindow", "SAVE"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.widget_setting),
_translate("MainWindow", "Page"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.widget_help),
_translate("MainWindow", "Page"))
        self.actionAbout.setText(_translate("MainWindow", "About"))
        t = "   Ця програма створена для розпізнавання усного українського\nмовлення та його
перетворення в текст. \n\n   Для того, щоб розпочати процес розпізнавання, в першому
пункті\nбокового меню натисніть кнопку \"Обрати файл\", оберіть необхідний\nаудіофайл формату
.wav та натисніть \"Розпізнати текст\". Отриманий\nтекст можна скопіювати в буфер обміну за
допомогою кнопки\nрозташованої в правому нижньому кутку.\n\n   Якщо потрібно записати нове
аудіо, в другому пункті бокового меню\nнатисніть на значок запису та надиктуйте необхідну
інформацію.\nФайл збережеться в форматі .wav у папці програми. \n\n   Змінити налаштування
розпізнавання звукового сигналу можна\nв третьому пункті бокового меню."
```

```
        self.information_text.setText(_translate("MainWindow", t))

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Модуль розпізнавання мовлення

```
import os
import os.path as path
import wave
import json
import shutil
import pyaudio
from loguru import logger
from pydub import AudioSegment
import speech_recognition as sr
from pydub.utils import make_chunks


class Params:
    def __init__(self, path='settings.json'):
        self.path = path
        self.params = self.get_params()

    def get_params(self) -> dict:
        with open(self.path, "r") as file:
            res = json.load(file)
        return res

    def save_params(self, new_params):
        with open(self.path, "w") as file:
            json.dump(new_params, file)

    def get_text_with_speech(self, pth: str) -> str:
        result = ""

        os.mkdir(self.params["split_dir"])
        myaudio = AudioSegment.from_file(pth, "wav")
        chunks = make_chunks(myaudio, self.params["chunk_length_ms"])
        for i, chunk in enumerate(chunks):
            chunk_name = "tmp/chunk{0}.wav".format(i)
            chunk.export(chunk_name, format="wav")

        r = sr.Recognizer()
        r.pause_threshold = self.params["pause_threshold"]
        dl = os.listdir(self.params["split_dir"])
        os.chdir(self.params["split_dir"])

        for f in dl:
            fs = f.split(".")
            if fs[1] == "wav":
                logger.info(f)
                with sr.AudioFile(f) as source:
                    audio_text = r.listen(source)
                try:
                    text = r.recognize_google(audio_text, language='uk-UA')
                    result = f"{result} {modified_text}"
                except:
                    logger.debug('Sorry.. run again...')

        modified_text = text.split()
```

```
    for_del = []
    modified_text[0] = modified_text[0].title()
    for i in range(len(modified_text)):
        if modified_text[i] == "крапка":
            modified_text[i] = "."
            if i+1 < len(modified_text):
                modified_text[i+1] = modified_text[i+1].title()
        if modified_text[i] == "кома":
            modified_text[i-1] += ","
            for_del.append(i)
        if i+1 < len(modified_text):
            if modified_text[i] == "знак" and modified_text[i+1] == "питання":
                modified_text[i] = "?"
                if i+2 < len(modified_text):
                    modified_text[i+2] = modified_text[i+2].title()
                for_del.append(i+1)
            if modified_text[i] == "знак" and modified_text[i+1] == "оклику":
                modified_text[i] = "!"
                if i+2 < len(modified_text):
                    modified_text[i+2] = modified_text[i+2].title()
                for_del.append(i+1)

    for_del = for_del[::-1]
    for i in range(len(for_del)):
        del modified_text[for_del[i]]

    logger.info(result)
    os.chdir(path.abspath(path.join(__file__, "../")))
    pth = os.path.join(os.path.abspath(os.path.dirname(__file__)), 'tmp')
    shutil.rmtree(pth)

    return result

def rec(self):
    CHUNK = 1024
    FORMAT = pyaudio.paInt16
    WAVE_OUTPUT_FILENAME = "tmp.wav"
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT,
                    channels=self.params["CHANNELS"],
                    rate=self.params["RATE"],
                    input=True,
                    frames_per_buffer=CHUNK)
    logger.info("* recording")
    frames = []
    for i in range(0, int(self.params["RATE"] / CHUNK * self.params["RECORD_SECONDS"])):
        data = stream.read(CHUNK)
        frames.append(data)
    logger.info("* done recording")
    stream.stop_stream()
    stream.close()
    p.terminate()
    with wave.open(WAVE_OUTPUT_FILENAME, 'wb') as wf:
        wf.setnchannels(self.params["CHANNELS"])
        wf.setsampwidth(p.get_sample_size(FORMAT))
        wf.setframerate(self.params["RATE"])
        wf.writeframes(b''.join(frames))
```

Додаток Г
(довідковий)

Таблиця Д1 – Результат автоматизованого тестування

№	Original	Hipothesis	Accura- cy, %	WER, %
0	історія нашого університету розпочалась у 1960 році коли у вінниці був створений загальнотехнічний факультет київського технологічного інституту харчової промисловості	історія нашого університету розпочалася у 1960 році коли у вінниці був створений загально технічний факультет київського технологічного інституту харчової промисловості	84,21	21,05
1	через рік він став складовою частиною цього інституту на базі якого у 1961 році був створений вінницький філіал	через рік він став складовою частиною цього інституту на базі якого у 1961 році був створений вінницький філіал	100,00	0,00
2	1 січня 1962 року загальнотехнічний факультет виділений на самостійний баланс	1 січня 1962 року загально технічний факультет виділений на самостійний баланс	90,00	20,00
3	за роки діяльності у складі київського технологічного інституту харчової промисловості вінницький загальнотехнічний факультет підготував і випустив близько тисячі студентів третього курсу які були переведені на другий етап навчання у різні вищі навчальні заклади	за роки діяльності у складі київського технологічного інституту харчової промисловості вінницькі загально технічний факультет підготував і випустив близько тисячі студентів третього курсу які були переведені на другий етап навчання у різні вищі навчальні заклади	93,93	9,09
4	з метою вдосконалення якості підготовки фахівців без відриву від виробництва та враховуючи відкриття у вінницькому загальнотехнічному факультеті спеціальностей які відповідали профілю київського політехнічного інституту кпі наказом номер 536 від 7 жовтня 1964 року вінницький загальнотехнічний факультет підпорядковано	з метою вдосконалення якості підготовки фахівців без відриву від виробництва та враховуючи відкриття у вінницькому технічному факультеті і спеціальності які відповідали профілю київського політехнічного інституту кпі наказом номер 536 від 7 жовтня 1964 року вінницький західний технічний факультет підпорядковано	88,09	16,66

	останньому як вінницький філіал кпі	останніми вінницький філіал кпі		
5	на початку вересня 1965 року вінницький філіал кпі розмістився в приміщенні по вулиці леніна 19 і в школі номер 15 мікрорайону вишенька	на початку вересня 1965 року вінницький філіал кпі розмістився в приміщенні по вулиці леніна 19 і в школі номер 15 мікрорайон вишенька	95,45	4,54
6	в цей же час проводилось будівництво навчального комплексу для філії в мікрорайоні вишенька	в цей же час проводились будівництво навчального комплексу для сірі мікрорайонів вишенька	69,23	30,77
7	в його будівництві брали участь 65 організацій і підприємств вінницької області різних міністерств і відомств	його подібності брали участь 65 організації підприємств вінницької області різних міністерств і відомств	73,33	26,67
8	рішенням вінницької обласної ради народних депутатів для розбудови у вінниці вищого навчального закладу політехнічного профілю на околиці міста було виділено 25 гектарів землі на яких уже в 1967 році було зведено два перших навчальних корпуси перший студентський гуртожиток та декілька господарських приміщень	рішення вінницької обласної ради народних депутатів для розбудови у вінниці вищого навчального закладу політехнічному профілю на околиці міста було виділено 25 гектарів землі на яких уже в 1967 році було зведено два перших навчальних корпуси 1 студентський гуртожиток та декілька господарських приміщень	90,48	9,52
9	вже у 1968 69 навчальному році в структурі філіалу були 4 факультети та 17 кафедр	вже у 1968 65 навчальному році в структурі серіалу були 4 факультети та 17 карта	80.0	20.0
10	у 1979 році введено в експлуатацію спортивно оздоровчий табір супутник в селі степашки на 160 відпочиваючих на одну зміну	у 1979 році введено в експлуатацію спортивно оздоровчий табір супутник в селі степашки на 160 відпочиваючих на одну змін	94,74	5,26
11	в 1984 році на базі вінницького філіалу кпі було створено вінницький політехнічний інститут сьомий в україні в цій групі вишів	1984 році на базі вінницького філіалу кпі було створено вінницький політехнічний інститут 7 в україні в цій групі	80.0	20.0
12	у 1994 році вінницький політехнічний інститут реорганізовано у вінницький державний технічний університет а в	у 1994 році вінницький політехнічний інститут реорганізовано у вінницькій державний технічний університет а в	90.0	10.0

	2003 році йому надано статус національного	2003 році йому надано статус		
13	сьогодні на території університету розташовано 12 корпусів 6 із яких об’єднані переходами в єдиний комплекс стадіон з рекортановим покриттям бігових доріжок підземний легкоатлетичний манеж стрілецький тир 12 відкритих спортивних майданчиків для ігрових видів спорту 6 гуртожитків та їдальня	сьогодні на території університету розташовано 12 корпусів шість із яких об'єднані переходили в єдиний комплекс стадіон з новим покриттям бігових доріжок підземний легкоатлетичний манеж стрілецький тир 12 відкритих спортивних майданчиків та ігрових видів спорту 6 гуртожитків такі дані	84,21	18,42
14	сьогодні вінницький національний технічний університет є одним з найбільших навчальних закладів на поділлі єдиним представником від україни в міжнародній організації винахідницьких організацій з центром у будапешті	сьогодні вінницький національний технічний університет є одним з найбільших навчальних закладів на поділі єдиним представником від україни в міжнародні організації винахідницьких організації з центром у будапешті	88,46	11,54
15	університет як центр освіти науки і культури в регіоні є одним із небагатьох вишів в україні що прийняті до міжнародної асоціації університетів	університет як центр освіти науки і культури в регіоні є одним із небагатьох вищих в україні що прийняті до міжнародної асоціації університет	90,91	9,09
16	вінницький національний технічний університет має двосторонні угоди про наукове та науково технічне співробітництво з технічними університетами навчальними закладами та організаціями азербайджану великобританії в’єтнаму грузії італії казахстану китаю нідерландів норвегії німеччини південної кореї польщі румунії сша франції швеції та іншими	вінницький національний технічний університет має двосторонні угоди про наукову та науково технічну співробітництво з психічними університетами навчальними закладами та організаціями азербайджану великобританії в'єтнаму грузії італії казахстану китаю нідерландів норвегії німеччини південної кореї в польщі румунії сша франції швеції та іншими	89,74	12,82
17	крім того університет залишається авторитетним експертним середовищем готуючи наукові висновки законопроектів надаючи консультації комерційним і бізнес структурам	крім того університет залишається активним експертним середовищем готуючи наукові висновки законопроектів надаючи консультації комерційним і бізнес структурам	93,94	6,06

	здійснюючи експертизу стану промислових об’єктів ґрунту води тощо на замовлення органів державної влади і місцевого самоврядування	здійснюючи експертизу стану промислових об'єктів ґрунту води тощо на замовлення органів державної влади і місцевого самоврядування		
18	університет використовує ефективну систему підготовки наукових кадрів вищої кваліфікації магістратура аспірантура докторантура	університет використовує ефективну систему підготовки наукових кадрів вищої кваліфікації магістратура аспірантура докторантура	100.0	0.0
19	плідно працюють спеціалізовані вчені ради організовуються і проводяться науково практичні конференції	працюють спеціалізовані вчені ради організовуються і проводяться науково практичні конференції	90,91	9,09

Додаток Д
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС ПЕРЕТВОРЕННЯ
«УСНЕ УКРАЇНСЬКЕ МОВЛЕННЯ – ТЕКСТ»
(тема)

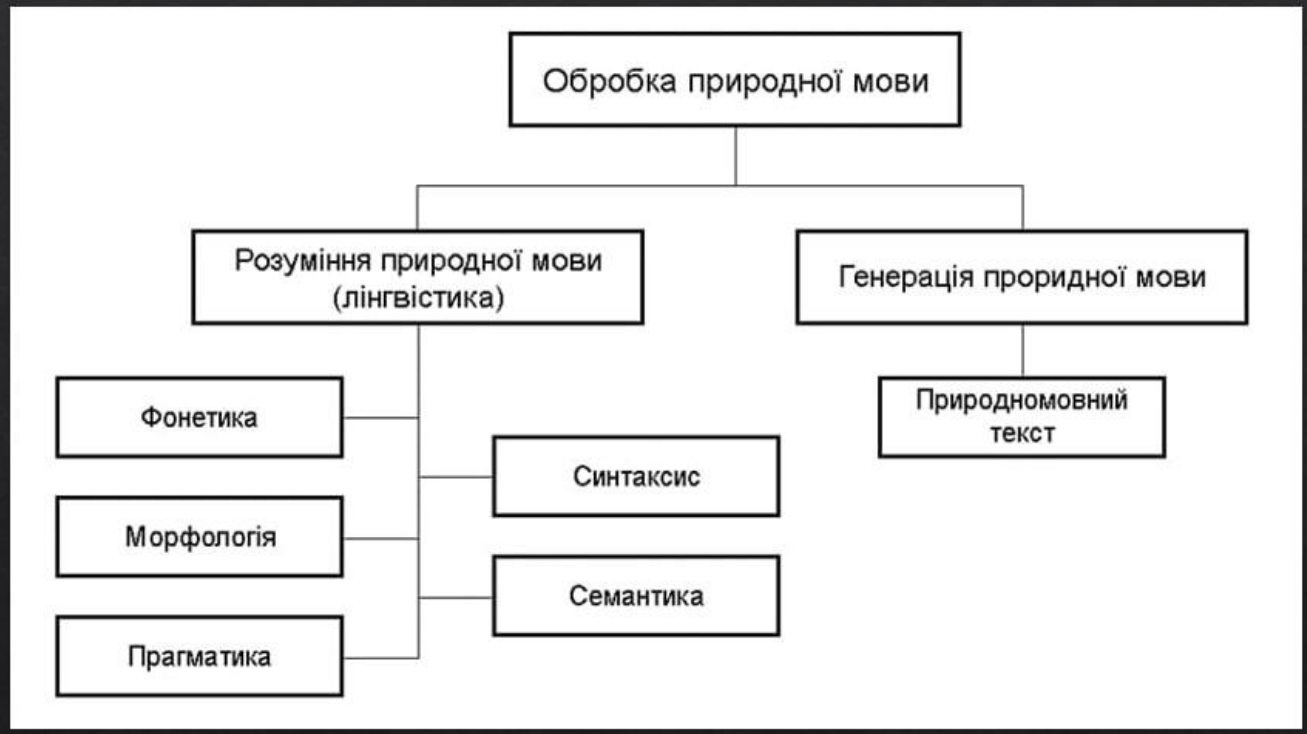
Студентка групи 2АКІТ-21м

_____ Юлія Нестюк
Підпис *Ім'я ПРІЗВИЩЕ*

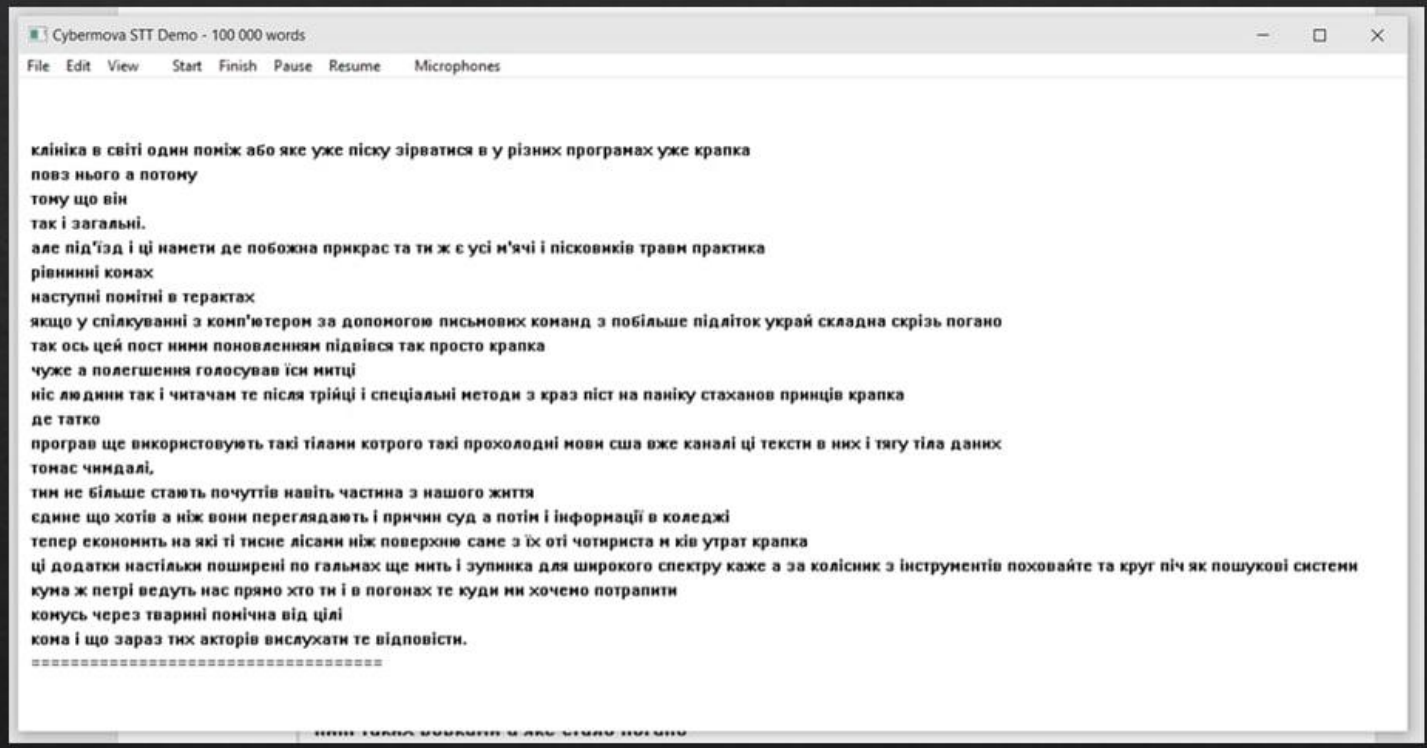
Керівник д.т.н., доцент, зав. кафедри КСУ

_____ В'ячеслав Ковтун
Підпис *Ім'я ПРІЗВИЩЕ*

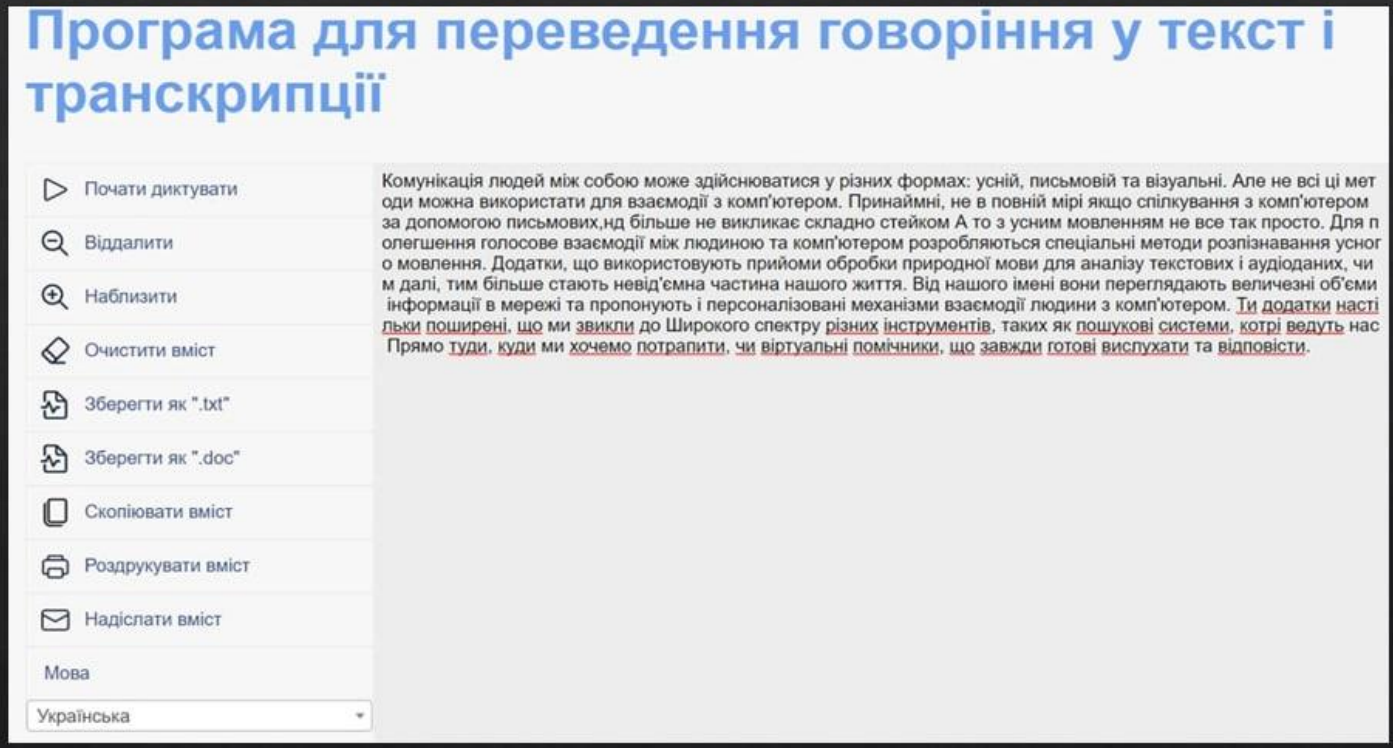
Завдання обробки природної мови



Результат розпізнавання ресурсом «Кібермова»

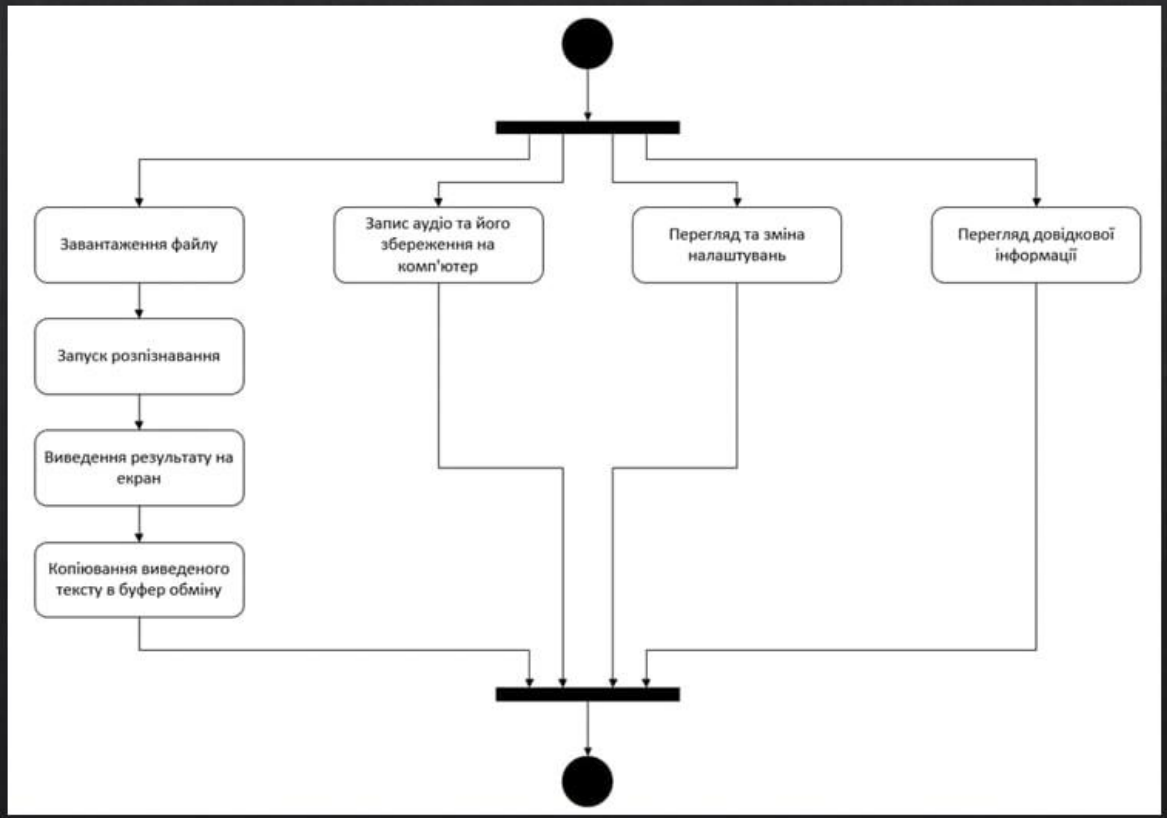


Результат розпізнавання Інтернет-ресурсом

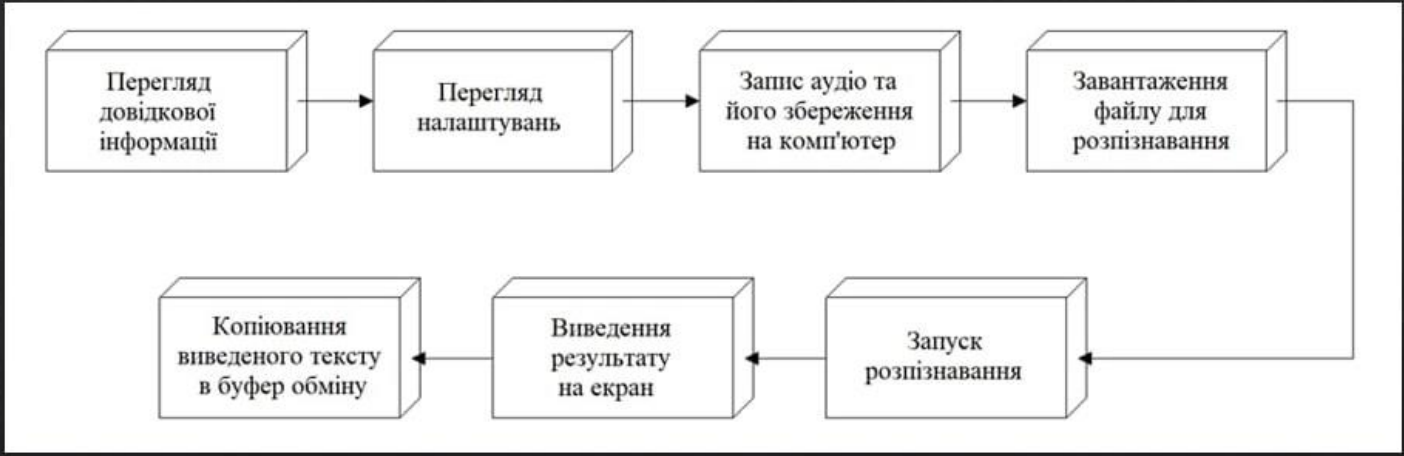


UML-діаграма варіантів використання

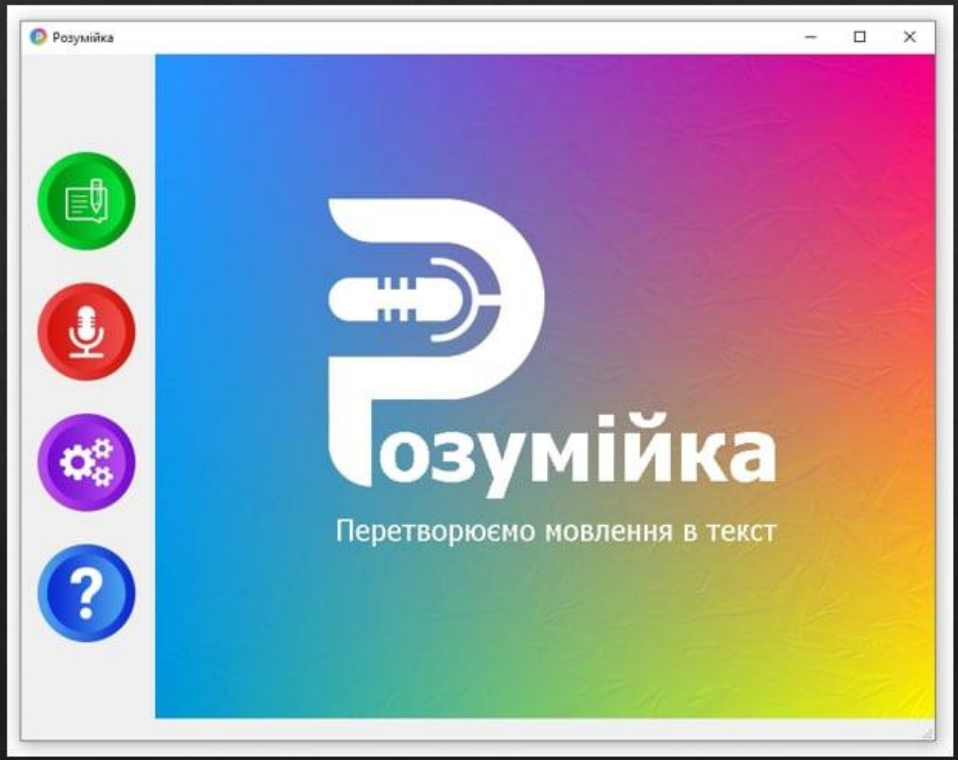
UML-діаграма активності



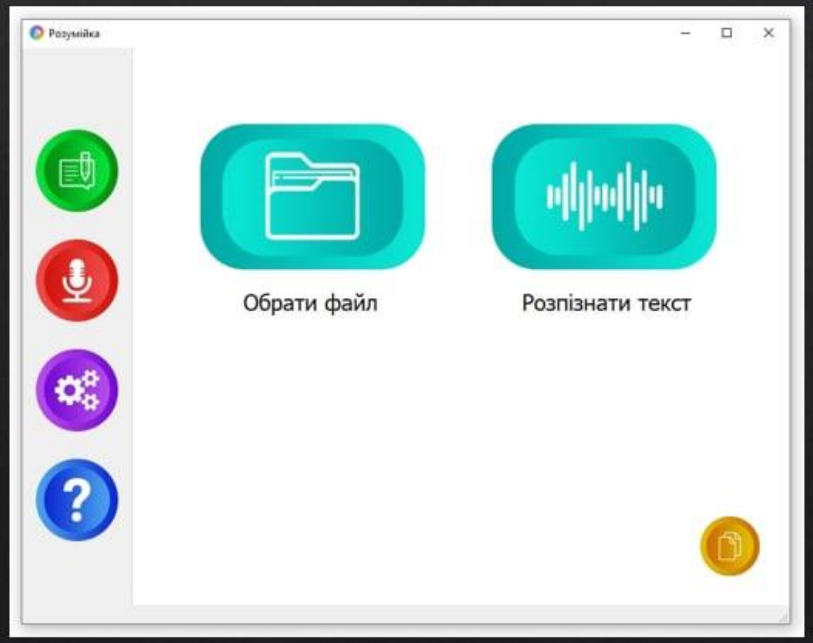
UML-діаграма розгортання

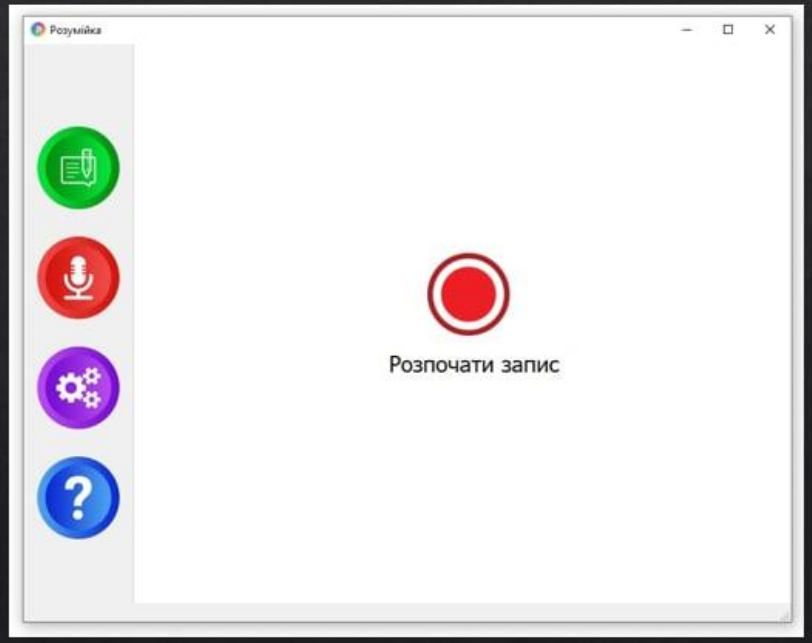


Стартове вікно програми

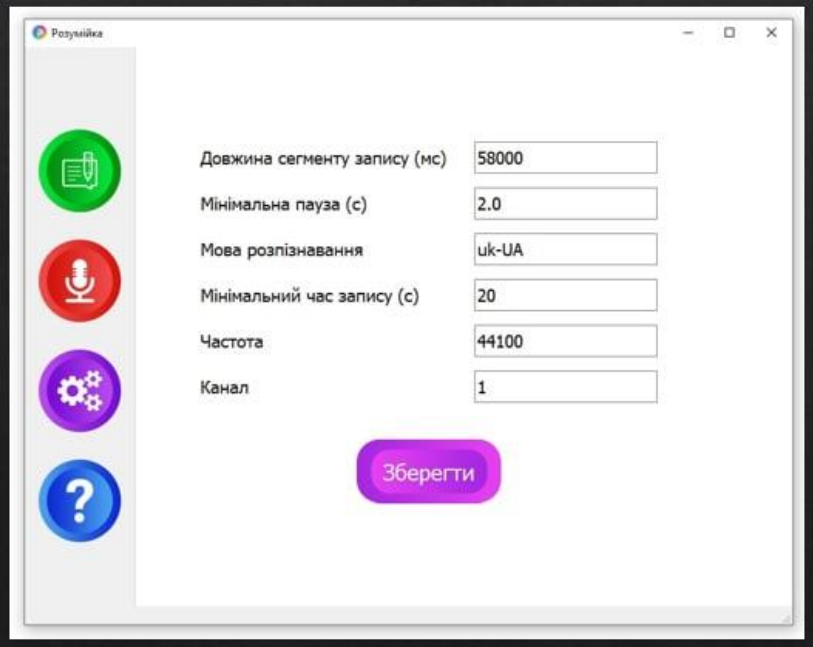


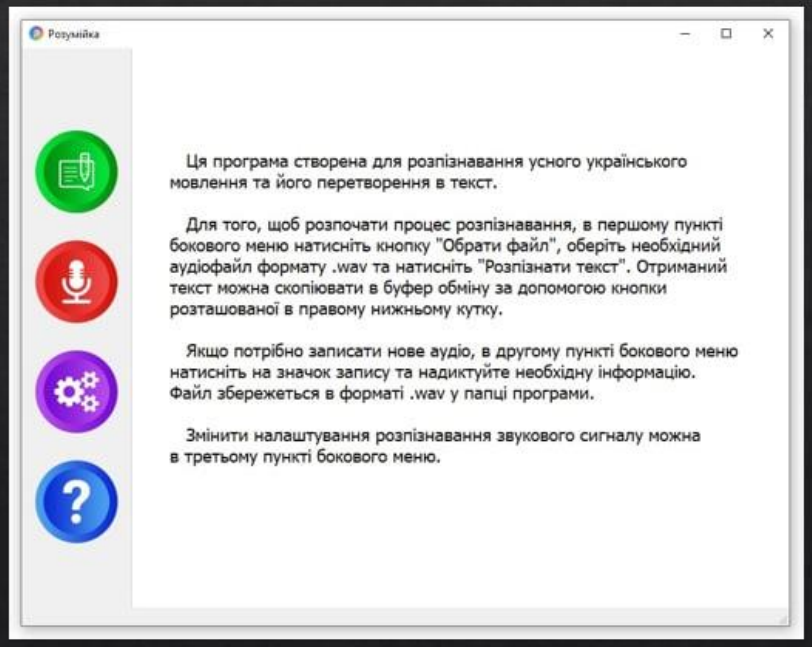
Опції розпізнавання мовлення та запису аудіофайлу



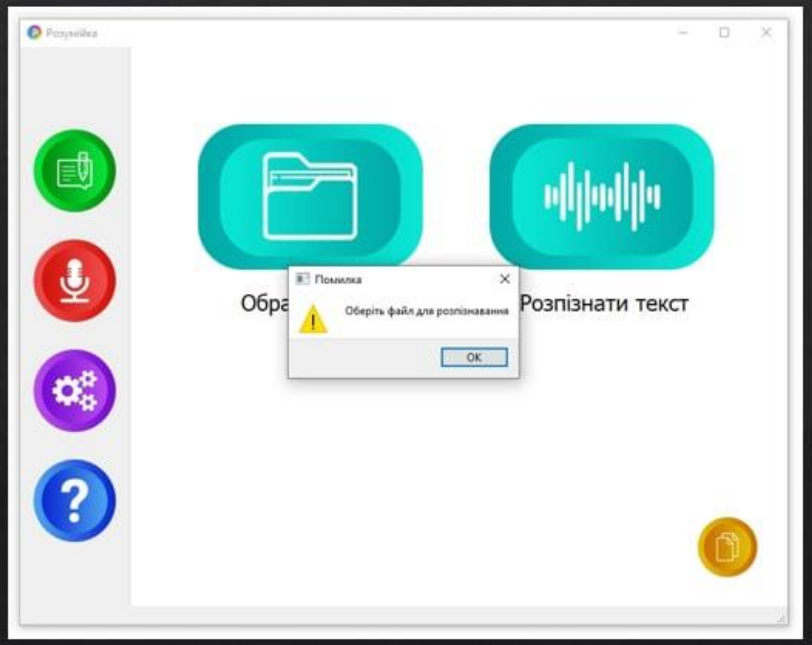


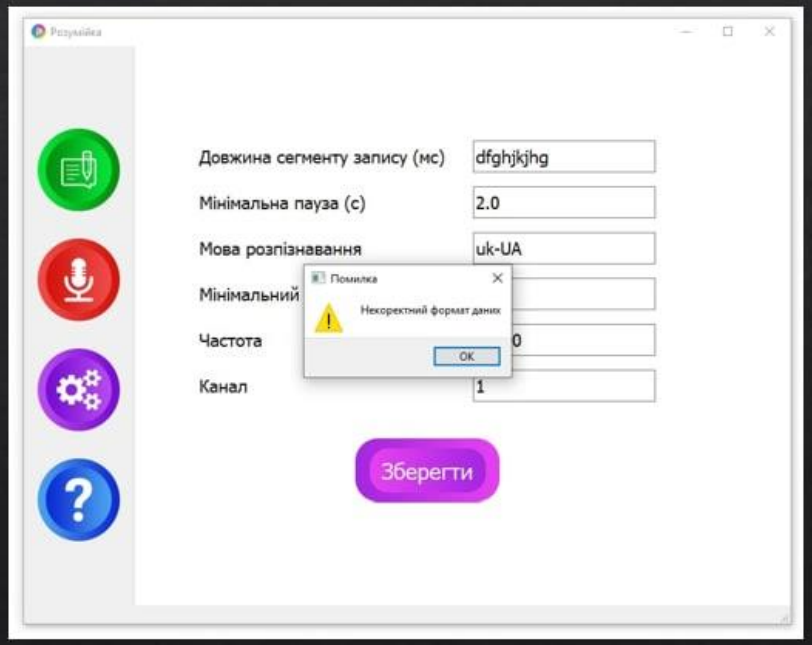
Опції зміни налаштувань та допомоги користувачеві



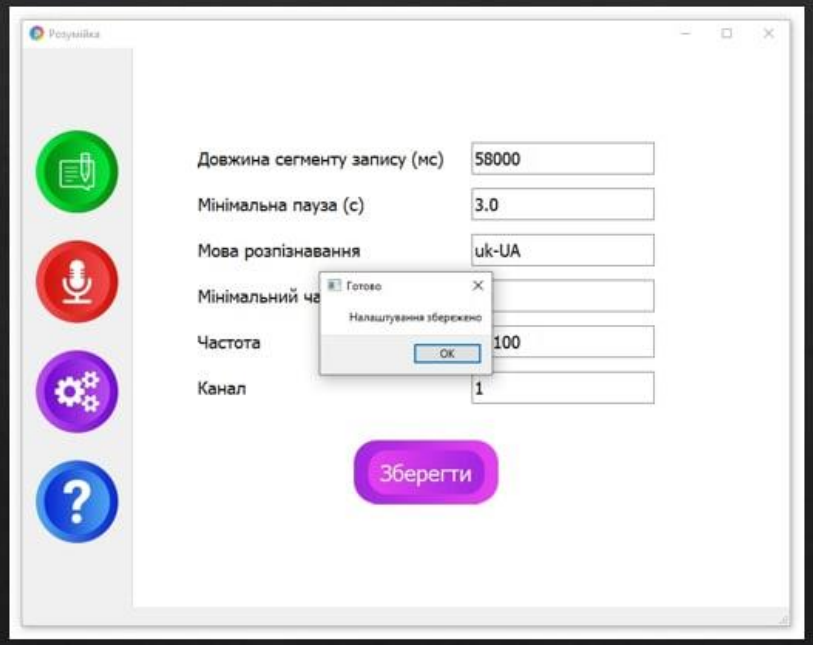


Обробка виключень – не завантажено файл та введення некоректних даних

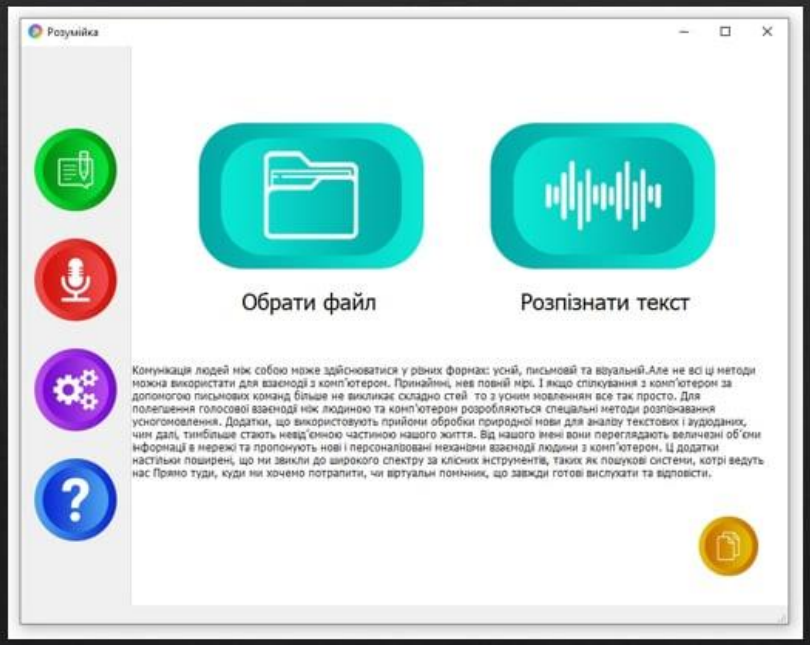




Вдале збереження налаштувань користувача та запису аудіофайлу



Результат розпізнавання



Результат

Комунікація людей між собою може здійснюватися у різних формах: усній, письмовій та візуальній. Але не всі ці методи можна використати для взаємодії з комп'ютером. Принаймні, не в повній мірі. І якщо спілкування з комп'ютером за допомогою письмових команд більше не викликає складно стей,ком а то з усним мовленням не все не так просто. Для полегшення голосової взаємодії між людиною та комп'ютером розробляються спеціальні методи розпізнавання усного мовлення. Додатки, що використовують прийоми обробки природної мови для аналізу текстових і аудіо даних, чим далі, тим більше стають невід'ємною частиною нашого життя. Від нашого імені вони переглядають величезні об'єми інформації в мережі та пропонують нові і персоналізовані механізми взаємодії людини з комп'ютером. Цієй додатки настільки поширені, що ми звикли до шШирокого спектру за куолісних інструментів, таких як пошукові системи, котрі ведуть нас пПрямо туди, куди ми хочемо потрапити, чи віртуальній помічники, що завжди готові вислухати та відповісти.

Результат автоматизованої оцінки точності розпізнавання