ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи

на ступінь вищої освіти магістр

на тему: «**РОЗРОБКА СИСТЕМИ ГОЛОСОВОГО УПРАВЛІННЯ РОЗУМНИМ БУДИНКОМ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ**»

Виконав: студент 6 курсу, групи ПДМ–62

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Ліщук І.В.
(прізвище та ініціали)

Керівник Дібрівний О.А.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Магістр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

Негоденко О.В.

“____” _____ 2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

ЛІЩУКУ ІГОРЮ ВАЛЕРІЙОВИЧУ

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка системи голосового управління розумним будинком з використанням штучного інтелекту»

Керівник роботи Дібрівний Олесь Андрійович, доцент кафедри, доктор філософії,

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «12» жовтня 2022 року № 122.

2. Строк подання студентом роботи 31.12.2022

3. Вихідні дані до роботи:

3.1 Методи розпізнавання мовлення;

3.2 Науково-технічна література з питань, пов’язаних з розпізнаванням мовлення за допомогою штучного інтелекту;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1 Оцінка та вимоги до якості системи;

4.2 Опис проектування системи;

4.3 Опис використаних технологій;

5. Перелік демонстраційного матеріалу (назва основних слайдів)

5.1 Титульний слайд;

5.2 Мета, об’єкт, предмет дослідження

5.3 Актуальність роботи

5.4 Приховані моделі Маркова

5.5 Покращення для прихований моделей Маркова

5.6 Опис розробленого алгоритму

5.7 Практичний результат

5.8 Висновки

6. Дата видачі завдання «14» жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з /п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		Виконано
2	Вимоги до системи		Виконано
3	Створення моделі для розпізнавання мовлення		Виконано
4	Проектування та розробка програмного забезпечення		Виконано
5	Оформлення розділів дипломної роботи		Виконано
6	Вступ, висновки, реферат		Виконано
7	Розробка обов‘язкових демонстраційних матеріалів		Виконано
8	Перевірка на антиплагіат		
9	Попередній захист роботи		
10	Проходження нормконтролю		
11	Здача роботи		

Студент _____ Ліщук І.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи с. 69, рис. 32, джерел 25.

ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОННА МЕРЕЖА, МАШИННЕ НАВЧАННЯ, РОЗУМНИЙ БУДИНОК, РОЗПІЗНАВАННЯ МОВЛЕННЯ, СИСТЕМА УПРАВЛІННЯ.

Об'єкт дослідження – процес управління розумним будинком.

Предмет дослідження – використання штучного інтелекту для голосового керування системою розумного будинку.

Мета роботи – вдосконалення існуючих алгоритмів голосового керування системою розумного будинку, з використанням штучного інтелекту.

Розумний дім – це будинок, який використовує інформаційні технології для моніторингу навколишнього середовища, керування електроприладами та спілкування із зовнішнім світом. Розумний дім – це складна технологія, яка водночас розвивається. Система автоматизації розумного дому була розроблена для автоматичного виконання деяких дій, які часто виконуються в повсякденному житті, щоб створити комфортніше та легше середовище життя.

Розпізнавання мовлення, яке часто неправильно називають розпізнаванням голосу, – це комп'ютерне програмне забезпечення, яке аналізує аудіосигнали, отримані мікрофоном, і перекладає їх на машинно-зрозумілу мову. Обробка мовлення базується на методах, які вимагають локального ЦП або онлайн-обробки через Інтернет-посилання.

У роботі проведено аналіз моделей розпізнавання мовлення та найбільш популярних систем керування розумним будинком, визначено їхні переваги та недоліки в рамках поставленого їм завдання, в тому числі розпізнавання мовлення.

ABSTRACT

The text part of the master's thesis 69 pages, 32 pictures, 25 references.

ARTIFICIAL INTELLIGENCE, NEURAL NETWORK, MACHINE LEARNING, SMART HOME, SPEECH RECOGNITION, CONTROL SYSTEM.

The object of research is the process of managing a smart home.

The subject of the research is the use of artificial intelligence for voice control of the smart home system.

The purpose of the work is to improve the existing voice control algorithms of the smart home system, using artificial intelligence.

A smart home is a home that uses information technology to monitor the environment, control electrical appliances, and communicate with the outside world. A smart home is a complex technology that is developing at the same time. A smart home automation system has been designed to automatically perform some actions that are often performed in daily life to create a more comfortable and easier living environment.

Speech recognition, often incorrectly called voice recognition, is computer software that analyzes audio signals picked up by a microphone and translates them into machine-readable language. Speech processing is based on methods that require a local CPU or online processing via an Internet link.

The work analyzes speech recognition models and the most popular smart home control systems, identifies their advantages and disadvantages within the framework of the task set for them, including speech recognition.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 10
ВСТУП 11
1 АНАЛІЗ ПРОБЛЕМАТИКИ ТА АНАЛОГІВ 13
1.1 Огляд предметної області 13
1.2 Глибоке навчання 14
1.2.1 Методи глибокого навчання 14
1.2.2 Глибоке навчання для розпізнання мовлення 17
1.2.3 Нейронні мережі проти глибокого навчання 17
1.2.4 Типи глибинних нейронних мереж 18
1.2.5 Досягнення в глибокому навчанні 20
1.3 Машинне навчання 21
1.3.1 Принцип роботи машинного навчання 23
1.3.2 Типи машинного навчання 23
1.4 Розпізнавання мовлення 25
1.4.1 Основні характеристики розпізнавання мовлення 27
1.4.2 Аналіз та порівняння існуючих рішень 28
Висновки до розділу 33
2 АНАЛІЗ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 34
2.1 Архітектура розпізнавання мовлення на основі HMM 34
2.1.1 Виділення ознак 35
2.1.2 Акустичні моделі HMM (базові-однокомпонентні) 37
2.1.3 Моделі мови N-грам 45
2.1.4 Декодування та генерація решітки 47

2.2 Розширення для покращення продуктивності ASR 51
2.2.1 Динамічні байєсовські мережі 51
2.2.2 Моделі суміші Гауса 53
2.2.3 Прогнозування даних 55
2.2.4 Коваріаційне моделювання 59
Висновки до розділу 64
3 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 65
3.1 Вибір інструментарію для розробки 65
3.1.1 Node.Js 65
3.1.2 Docker 65
3.1.3 TypeScript 66
3.1.4 Nx.Js 66
3.1.5 Angular 66
3.2 Налаштування інструментів розробки 67
3.3 Розробка алгоритму розпізнавання мовлення 70
ВИСНОВКИ 78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТКИ 83

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ШІ	Штучний інтелект
НМ	Нейронна мережа
ML	Machine learning
REST	Representational State Transfer
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
MVC	Model-view-controller
SQL	Structured query language
VPS	Virtual Private Server
BLL	Business logic layer
ILL	Infrastructure logic layer
ALL	Application logic layer
DDD	Domain driven design
DTW	Dynamic time warping
IoT	Internet of Things
CNN	Convolutional neural network
LSTM	Long short-term memory
RNN	Recurrent neural network

ВСТУП

Концепція домашньої автоматизації стає все більш актуальною темою в цьому столітті, і вона відіграє життєво важливу роль у нашому повсякденному житті як окремих людей, як новаторів, так і як колективного суспільства. Це скорочує людську працю, час і кількість зусиль, які витрачаються на виконання щоденних важливих і виснажливих завдань.

Сьогодні попит на кращі системи домашньої безпеки значно зріс. Домівки повинні бути не тільки безпечними, але й мати просту та вдосконалену систему керування. У наші дні домашня автоматизація впроваджується в кожному розумному домі, і вона враховує всі фактори, включаючи безпеку приладів, особливо коли людини немає вдома, що врешті-решт також допомагає мінімізувати споживання електроенергії. Це широкомасштабна концепція, оскільки вона може мати від одного автоматизованого пристрою до кількох автоматизованих пристроїв і може мати весь будинок під контролем однієї системи в будь-який момент часу. Це величезне завдання може породжувати різноманітні виклики, але як тільки їх буде розв’язано, воно може призвести до низки можливостей і, отже, стати однією з найкорисніших і передових інновацій нашого технологічного покоління та нашого суспільства, що швидко розвивається.

Актуальність тематики дослідження пов’язана з тим що системи керування розумним будинком – це сучасні інструменти підвищення рівню комфорту та життя людини за рахунок автоматизації та спрощення рутинних занять за рахунок приладів котрі керуються системою, одним з елементів керування котрої являється голос. З розвитком технологій розпізнавання мови з’являється більше варіантів використання цих технологій, що і робить тематику дослідження актуальною для вивчення, впровадження та вдосконалення подібних технологій у майбутніх проектах.

Об’єкт дослідження – процес управління розумним будинком.

Предмет дослідження – використання штучного інтелекту для голосового керування системою розумного будинку.

Мета роботи – вдосконалення існуючих алгоритмів голосового керування системою розумного будинку, з використанням штучного інтелекту.

Методи дослідження – порівняльні методи дослідження систем керування розумних будинків та методів розпізнавання мовлення.

В результаті магістерської роботи були проаналізовані існуючі системи керування розумним будинком та розпізнавання мовлення за допомогою нейронних мереж що дозволить спроектувати власну систему для керування розумним будинком за допомогою голосу.

1 АНАЛІЗ ПРОБЛЕМАТИКИ ТА АНАЛОГІВ

1.1 Огляд предметної області

Розумний будинок – це житловий будинок, в якому всі системи належним чином інтегровані між собою. До них належать, серед іншого, кондиціонування повітря, освітлення, жалюзі, а також опалення та рекуперація. Усі системи в розумному домі взаємопов'язані, працюють разом і ними легко керувати. Це рішення є дуже практичним і робить центральне керування всією будівлею надзвичайно легким для користувача.

Визначна еволюція Інтернету речей (IoT) дозволила реалізувати розумні будинки майбутнього. В ідеалі пристрої IoT у розумному домашньому середовищі можуть безперебійно спілкуватися один з одним через Інтернет, маючи інтелектуальні можливості для прийняття рішень на основі великих даних. Хоча очікується покращення якості людського життя, поточні прогалини у практичному впровадженні створюють перешкоди для впровадження розумних пристроїв у домашніх умовах.

Дослідження показують, що потенційна цінність розумних будинків може бути реалізована, якщо усунути існуючі бар'єри, такі як сумісність між системами IoT, проблеми безпеки та конфіденційності, а також прогалини в даних IoT щодо зручності використання. Крім того, існує недостатня обізнаність щодо повного використання можливостей різних пристроїв IoT для використання машинного інтелекту для голосової інтегрованої автоматизації розумного будинку. Наразі користувачі вважають здоров'я та фітнес двома основними категоріями додатків для успішного впровадження Інтернету речей, а низка пристроїв і приладів все ще з'являється для створення розумних будинків майбутнього. Більше економічних міркувань приділяється продуктивності людини в офісному середовищі, а також безпеці на заводах або робочих майданчиках для автоматизації робочого місця. Отже, дослідження та комерційні розробки були зосереджені на додатках «бізнес-

бізнес», а не на споживчих додатках. Проте деякі інтелектуальні можливості з взаємопов’язаними термостатичними пристроями, пилососами з автоматичним керуванням, автоматизованим освітленням, дверними системами/системами безпеки, кухонними та пральними приладами можна застосувати для досягнення автоматизації роботи, енергоефективності та інших сфер економічної цінності в домашніх умовах. Важливо орієнтуватися на клієнта в наданні будь-якої інтелектуальної послуги в системах домашньої автоматизації, щоб покращити адаптацію користувачами.

1.2 Глибоке навчання

Глибоке навчання – це підгалузь машинного навчання (ML) і являє собою набір архітектур нейронних мереж, які вирішують складні найсучасніші проблеми. Ці архітектури (або моделі) називаються, серед інших, згортковими нейронними мережами (CNN) і довгою короткочасною пам’яттю (LSTM).

Архітектури глибокого навчання називають глибокими, оскільки вони мають багато рівнів. Глибина шарів важлива, як ви побачите в цьому розділі. Глибоке навчання відноситься до архітектур нейронних мереж, які включають багато рівнів і мають можливість навчитися (через навчання) відображати вхідні дані, наприклад зображення, на один або кілька вихідних даних, наприклад класифікацію. Класифікація може показувати, чи містить зображення кота чи не містить кота. Оскільки компанії експериментували з проблемами, що містять інтенсивні дані, такими як перетворення мови в текст і комп’ютерне бачення, науковці звернулися до глибокого навчання, щоб вирішити бізнес-проблеми, які неможливо вирішити за допомогою нелінійних алгоритмів.

1.2.1 Методи глибокого навчання

Глибоке навчання являє собою набір архітектур, побудованих на ідеях нейронних мереж. Нейронні мережі – це обчислювальні структури – мережі обчислювальних елементів, які можна налаштувати за допомогою навчання, а

потім застосувати до проблем. Давайте повернемося до витоків глибокого навчання та дослідимо основи нейронних мереж.

Нейрони до багаторівневих мереж

Нейронна мережа – це мережа нейронів, які реалізують математичну функцію. Вхід подається в мережу (зазвичай у вигляді вектора), окремі нейрони обчислюють свої виходи на кожному рівні до виходу. Цей процес називається передаванням і являє собою виконання мережі. Кожен нейрон може подаватись із вхідного або попереднього рівня через вагу, яка індивідуально регулює певний вхід. Потім вхідні дані, помножені на вагу, підсумовуються та передаються через функцію активації для визначення виходу (рис. 1).

Рисунок 1 – Окремий нейрон і його математичний еквівалент

$$O = f\big((i_0 * w_0) + (i_1 * w_1) + (i_2 * w_2)\big) \quad (1)$$

Функція активації може бути будь-якої з безлічі різних типів (крокова функція, сигмоїда тощо) і зазвичай вибирається для типу мережі та проблеми. Для багаторівневих мереж функція активації вибирається для введення нелінійності. Введення нелінійності дозволяє багатошаровим мережам вирішувати відносно складні проблеми з невеликою кількістю нейронів.

Ці нейрони можна зібрати для вирішення складних завдань, як правило, за допомогою шарів. Багаторівневі мережі включають щонайменше три рівні: вхідний рівень, прихований рівень і вихідний рівень (рис. 2).

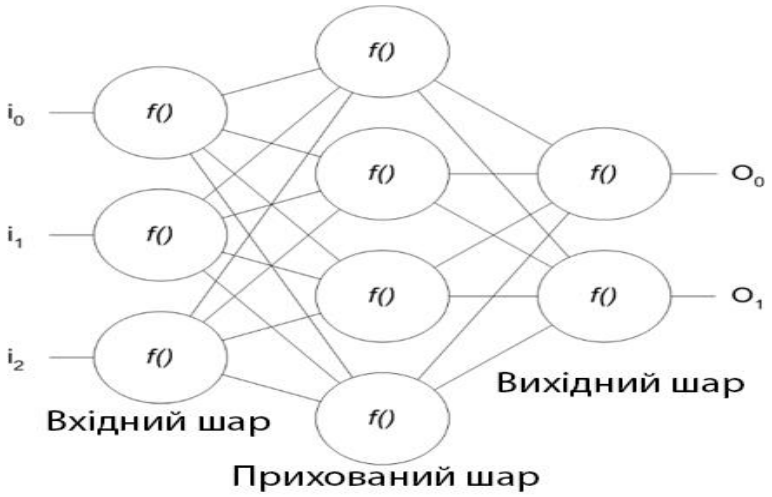


Рисунок 2 – Багатошарова (3-рівнева) нейронна мережа

Мережа працює простим способом: застосовуються вхідні дані, а потім обчислюється кожен рівень мережі, починаючи з вхідного рівня. Оскільки вхідний шар подає прихований шар, прихований шар обчислюється наступним. Така поведінка називається прямою передачею, оскільки обчислення просуваються вперед і в мережі не утворюються цикли чи петлі.

Навчання

Вагові коефіцієнти мережі визначаються індивідуально та забезпечують основу для відображення входів і виходів. Визначення цих вагових коефіцієнтів є тренуванням, і воно відбувається протягом багатьох ітерацій, щоб налаштувати вагові коефіцієнти для відображення вхідних даних і вихідних даних із прийнятним рівнем помилки.

У багаторівневих мережах зазвичай використовується клас алгоритмів, який називається зворотним поширенням, щоб налаштувати вагові коефіцієнти мережі. Вони роблять це, беручи вибірку з навчального набору, застосовуючи вхідні дані, обчислюючи вихідні дані, а потім порівнюючи їх з очікуваними результатами (прохід вперед). Різниця між результатом і очікуваним результатом є помилкою. Ця помилка використовується для налаштування вагових коефіцієнтів мережі, починаючи з вихідного рівня та переходячи до вхідного рівня у зворотному проході. Для багатьох навчальних зразків це зворотне поширення помилки для

коригування вагових коефіцієнтів у мережі мінімізує помилку для загального навчального набору.

Процес застосування навчального зразка до мережі, перевірка відповіді, а потім відповідне коригування мережі є основою навчання під наглядом. Він контролюється, оскільки навчальний набір включає бажану поведінку мережі.

1.2.2 Глибоке навчання для розпізнання мовлення

Розпізнавання мовлення було святим Граалем для машинного навчання з самого початку. Не дивно, що глибокі нейронні мережі просунули цю сферу. Глибокі нейронні мережі підвищили точність розпізнавання, а також працюють безпосередньо з мовою, а не з проміжними представленнями.

Двома ключовими методами, що застосовуються в автоматичному розпізнаванні мовлення, є CNN і рекурентні нейронні мережі (RNN), враховуючи природу мовлення на основі послідовності. Тимчасова класифікація коннекціоністів також використовувалася для навчання як мереж RNN, так і мереж LSTM, враховуючи різницю в синхронізації мовлення.

1.2.3 Нейронні мережі проти глибокого навчання

Ключовою особливістю глибоких нейронних мереж є глибина: кількість шарів на додаток до ширини або кількості елементів обробки в кожному шарі. Але глибокі нейронні мережі еволюціонували від типових багатошарових мереж, які привели нас до глибокого навчання. Архітектура глибоких нейронних мереж відрізняється від їхніх багатошарових предків. CNN, які добре працюють із даними зображення, вибіркою та пулом пікселів із зображення для обробки. RNN, які ідеально підходять для таких послідовних даних, як текст, розглядають не лише вхідні дані, але й вхідні дані, які передують і слідують за ним.

Структура нейронів також змінилася в міру розвитку глибинного навчання. Замість того, щоб просто підсумовувати зважені продукти, що проходять через функцію активації, нейрони в нових мережах глибокого навчання, таких як мережі

LSTM, включають ворота для регулювання потоку інформації та навіть для забуття інформації.

1.2.4 Типи глибинних нейронних мереж

Повторювані нейронні мережі

RNN мають різні архітектурні стилі, але всі включають поведінку внутрішнього стану, що означає, що їх можна застосовувати до проблем у часовій області. Як показано на рис. 3, проста мережа прямого зв’язку доповнюється двома нейронами, які живляться з прихованого шару (званого мережею Елмана), які потім повертаються на прихований рівень. Цей цикл забезпечує просту форму пам’яті в нейронній мережі, що робить їх ідеальними для прогнозування часових рядів.

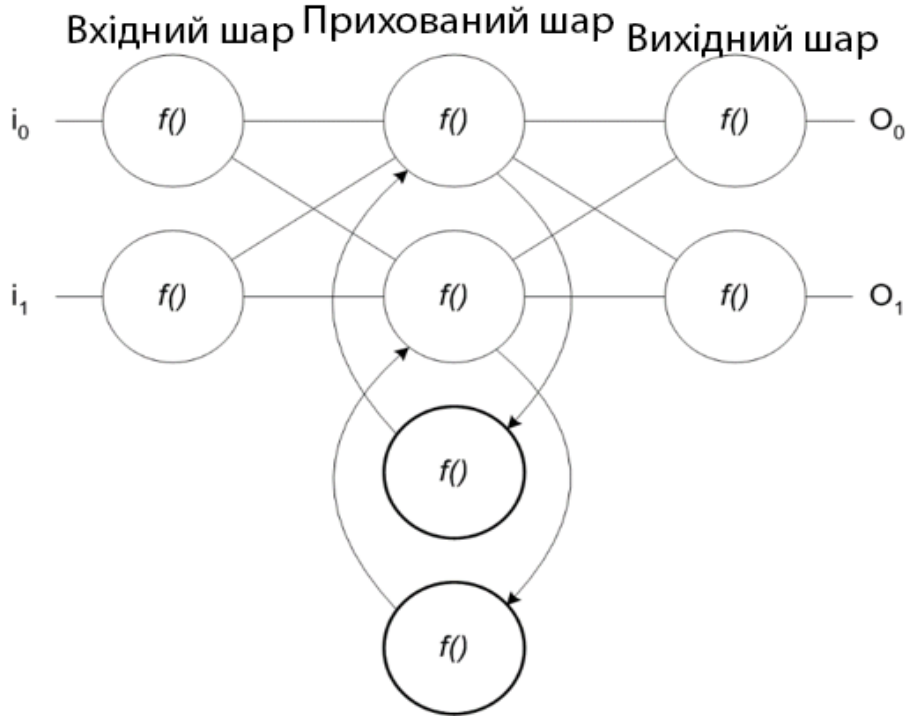


Рисунок 3 – Розгортання рекурентної нейронної мережі

Навчання RNN вимагає варіанту зворотного поширення, яке називається зворотним поширенням у часі. Це узагальнення стандартного алгоритму зворотного поширення, який зазвичай використовується в багатошарових нейронних мережах.

Згорткові нейронні мережі

CNN – це архітектура глибокої нейронної мережі, яка чудово класифікує зображеннязавдання. Він працює за допомогою серії операцій згортання та максимального об’єднання вхідних даних (див. рис. 4). Згортка (або фільтр) бере невелику частину вхідного зображення та застосовує фільтр, який зменшує розмірність вхідних даних до меншої матриці. Цей крок виконується для багатьох областей вхідного зображення. Наступним кроком є максимальне об’єднання, яке подібним чином зменшує розмірність зображення, повертаючи максимальне значення для заданої вхідної матриці (введення з шару згортки). Цей процес повторюється для певної кількості шарів, доки ми не досягнемо рівня класифікації. Останній рівень максимального об’єднання живить нейронну мережу, яка повністю зв’язана (кожен вихідний клас живиться з виходів останнього рівня максимального об’єднання). Рівень класифікації використовує ознаки високого рівня, щоб визначити клас для даного вхідного зображення.

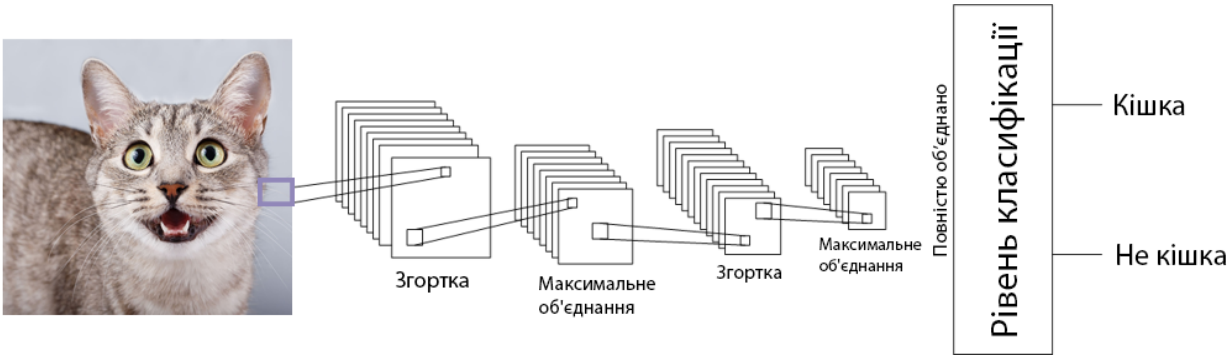


Рисунок 4 – Згорточна нейронна мережа

Зворотне розповсюдження зазвичай використовується для навчання CNN через контрольоване навчання (коригування вагових коефіцієнтів мережі як функції помилки класифікації).

Мережі довготривало-короткочасної пам’яті

LSTM є RNN із внутрішньою пам’яттю. Блок LSTM використовується для побудови мережі, яка існує в одному вимірі (як показано на малюнку 6) або в кількох вимірах, де кожен блок подає блоки праворуч і вище. Блок LSTM містить три вентиля, які регулюють потік інформації всередині блоку. Ці ворота є вхідними воротами, які контролюють надходження нової інформації до блоку; шлюз забуття,

який контролює, коли внутрішня пам'ять очищається; і вихідний вентиль, який використовується для обчислення виходу блоку. З’єднання між блоками та воротами зважені, завдяки чому їх можна регулювати під час навчання.

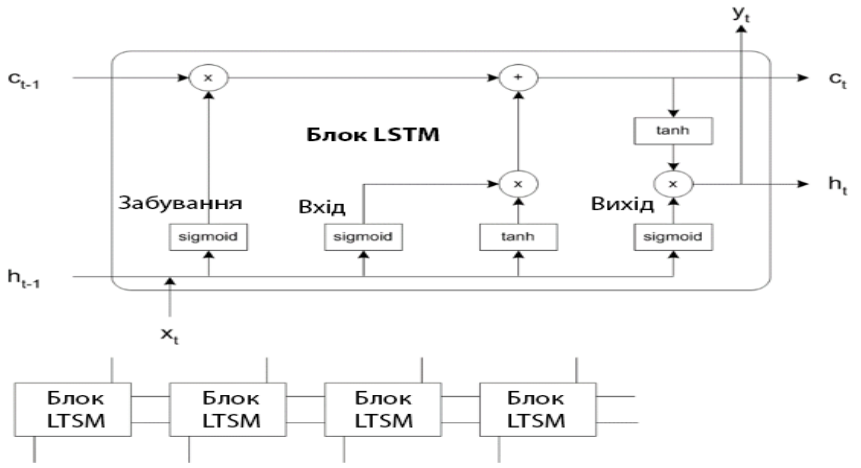


Рисунок 5 – Мережа та блок довгострокової пам’яті

Мережі LSTM, як і RNN, ідеально підходять для вирішення проблем послідовності. Одним із найцікавіших застосувань мереж LSTM було побудова людського мовного опису для зображення (подається з CNN). Мережі LSTM можна навчити за допомогою зворотного поширення в часі з контрольованим навчанням.

1.2.5 Досягнення в глибокому навчанні

Глибинне навчання розвинуло сучасні технології в ряді бізнес-проблем. В одній важливій сфері CNN досягли точності сертифікованих дерматологів у класифікації зображень уражень шкіри як доброякісних або злоякісних. Але в міру того, як архітектури глибокого навчання розвиватимуться, разом із їхніми методами, вони застосовуватимуться до нових проблем і просуватимуться далі.

Однією з ключових проблем глибокого навчання є навчання. Графічні процесори скоротили час, необхідний для навчання глибоких нейронних мереж, але з появою нових проблемних областей також з’являються складніші архітектури, які потребують більших наборів даних. Апаратне забезпечення продовжує застосовуватися в цих областях, з новими інноваціями в графічних процесорах, щоб зосередитися на завданнях глибокого навчання. На перший погляд, збільшення кількості ядер графічного процесора та пропускної здатності

пам’яті позитивно впливає на глибоке навчання нейронної мережі. Нові технології, які дозволяють графічним процесорам спілкуватися один з одним напряму, а не через свою хост-систему, також показали переваги для певних навчальних навантажень.

Хоча глибоке навчання зазвичай обмежувався серверами з високопродуктивними багатоядерними процесорами та графічними процесорами, тепер він розширює свою сферу дії на невеликі вбудовані пристрої, такі як смартфони. Ця еволюція просувається на двох фронтах: модифікація алгоритмів глибокої нейронної мережі, щоб зробити їх більш дружніми до вбудованого середовища з обмеженими ресурсами, і спеціальні процесори, які оптимізують ці технології в більш енергоефективний спосіб.

Нещодавнє дослідження показало, що навчання глибокої нейронної мережі помірного розміру для програми NLP призвело до такої ж кількості викидів вуглекислого газу, що й п’ять середньостатистичних автомобілів США за весь термін служби. Одна з важливих тем досліджень для боротьби з цим називається перенесенням навчання. Трансферне навчання передбачає використання попередньо навченої глибокої нейронної мережі для схожої проблемної області. Таким чином, нейронній мережі не потрібно навчатися з нуля; замість цього він отримує фору, а потім налаштовує мережу для конкретного проблемного домену. Цей метод показав великі надії на скорочення часу навчання на додаток до мінімізації кількості нових даних, необхідних для навчання.

1.3 Машинне навчання

Машинне навчання (ML) – це дисципліна штучного інтелекту (ШІ), яка надає машинам здатність автоматично навчатися на основі даних і минулого досвіду, визначаючи закономірності, щоб робити прогнози з мінімальним втручанням людини.

Методи машинного навчання дозволяють комп’ютерам працювати автономно без явного програмування. Програми ML отримують нові дані, і вони можуть самостійно навчатися, рости, розвиватися та адаптуватися.

Машинне навчання отримує глибоку інформацію з великих обсягів даних, використовуючи алгоритми для визначення шаблонів і навчання в ітераційному процесі. Алгоритми ML використовують обчислювальні методи, щоб навчатися безпосередньо з даних замість того, щоб покладатися на будь-яке заздалегідь визначене рівняння, яке може слугувати моделлю.

Продуктивність алгоритмів ML адаптивно покращується зі збільшенням кількості доступних зразків під час процесів «навчання». Наприклад, глибоке навчання – це субдомен машинного навчання, який навчає комп’ютери імітувати природні людські риси, наприклад навчання на прикладах. Він пропонує кращі параметри продуктивності, ніж звичайні алгоритми ML.

Незважаючи на те, що машинне навчання не є новою концепцією, починаючи з часів Другої світової війни, коли використовувалася машина Enigma, здатність автоматично застосовувати складні математичні обчислення до зростаючих обсягів і різновидів доступних даних є відносно нещодавньою розробкою.

Сьогодні, з розвитком великих даних, Інтернету речей і повсюдних обчислень, машинне навчання стало необхідним для вирішення проблем у багатьох сферах, таких як:

- обчислювальне фінансування (кредитний скоринг, алгоритмічна торгівля);
- комп'ютерний зір (розпізнавання обличчя, відстеження руху, виявлення об'єктів);
- обчислювальна біологія (секвенування ДНК, виявлення пухлин головного мозку, відкриття ліків);
- автомобільна, аерокосмічна та промислова промисловість (прогнозне технічне обслуговування);
- обробка природної мови (розпізнавання голосу);

1.3.1 Принцип роботи машинного навчання

Алгоритми машинного навчання формуються на основі навчального набору даних для створення моделі. Коли нові вхідні дані вводяться в навчений алгоритм ML, він використовує розроблену модель для прогнозування.

Далі прогноз перевіряється на точність. Залежно від точності алгоритм ML або розгортається, або постійно навчається за допомогою розширеного навчального набору даних, доки не буде досягнуто бажаної точності.

1.3.2 Типи машинного навчання

Алгоритми машинного навчання можна навчити багатьма способами, у кожного з яких є свої плюси та мінуси. Виходячи з цих методів і способів навчання, машинне навчання поділяється на чотири основні типи:

- контрольоване машинне навчання;
- машинне навчання без нагляду;
- напівконтрольоване навчання;
- навчання з підкріпленням.

Контрольоване машинне навчання

Цей тип ML передбачає нагляд, коли машини навчаються на позначених наборах даних і можуть передбачати результати на основі наданого навчання. Позначений набір даних вказує на те, що деякі вхідні та вихідні параметри вже зіставлено. Таким чином, машина навчається з введенням і відповідним виходом. Пристрій створено для прогнозування результату за допомогою тестового набору даних на наступних етапах.

Наприклад, розглянемо вхідний набір даних зображень папуги та ворони. Спочатку машину навчають розуміти зображення, включаючи колір, очі, форму та розмір папуги та ворони. Після навчання надається вхідне зображення папуги, і очікується, що машина ідентифікує об’єкт і передбачить результат. Навчена машина перевіряє різні характеристики об’єкта, такі як колір, очі, форма тощо, у

вхідному зображенні, щоб зробити остаточний прогноз. Це процес ідентифікації об’єктів у керованому машинному навчанні.

Основна мета методики навчання під керівництвом полягає в тому, щоб зіставити вхідну змінну (a) з вихідною змінною (b). Кероване машинне навчання далі класифікується на дві великі категорії:

- Класифікація: вони стосуються алгоритмів, які вирішують проблеми класифікації, де вихідна змінна є категоричною; наприклад, так чи ні, правда чи хибність, чоловік чи жінка тощо. Реальні програми цієї категорії очевидні у виявленні спаму та фільтрації електронної пошти.

Деякі відомі алгоритми класифікації включають алгоритм випадкового лісу, алгоритм дерева рішень, алгоритм логістичної регресії та алгоритм опорних векторних машин.

- Регресія: Алгоритми регресії обробляють проблеми регресії, коли вхідні та вихідні змінні мають лінійний зв’язок. Відомо, що вони передбачають безперервні вихідні змінні. Приклади включають прогноз погоди, аналіз ринкових тенденцій тощо.

Популярні алгоритми регресії включають алгоритм простої лінійної регресії, алгоритм багатовимірної регресії, алгоритм дерева рішень і регресію ласо.

Машинне навчання без нагляду

Навчання без нагляду стосується техніки навчання, яка позбавлена нагляду. Тут машина навчається за допомогою немаркованого набору даних і може передбачати результат без будь-якого нагляду. Алгоритм неконтрольованого навчання має на меті згрупувати несортований набір даних на основі подібностей, відмінностей і шаблонів вхідних даних.

Наприклад, розглянемо вхідний набір даних зображень контейнера, наповненого фруктами. Тут зображення не відомі моделі машинного навчання. Коли ми вводимо набір даних у модель ML, завдання моделі полягає в тому, щоб ідентифікувати шаблон об’єктів, наприклад колір, форму чи відмінності, які видно на вхідних зображеннях, і класифікувати їх. Після категоризації машина прогнозує результат, коли його перевіряють за допомогою тестового набору даних.

Неконтрольоване машинне навчання далі класифікується на два типи:

- Кластеризація: Техніка кластеризації стосується групування об’єктів у кластери на основі таких параметрів, як подібність або відмінність між об’єктами. Наприклад, групування клієнтів за продуктами, які вони купують.

Деякі відомі алгоритми кластеризації включають алгоритм кластеризації K-Means, алгоритм середнього зсуву, алгоритм DBSCAN, аналіз основних компонентів і аналіз незалежних компонентів.

- Асоціація: навчання асоціаціям стосується визначення типових зв’язків між змінними великого набору даних. Він визначає залежність різних елементів даних і відображає відповідні змінні. Типові програми включають аналіз використання Інтернету та аналіз ринкових даних.

Популярні алгоритми, які підкоряються правилам асоціації, включають алгоритм Apriori, алгоритм Eclat і алгоритм FP-Growth.

Напівконтрольоване навчання

Напівконтрольоване навчання містить характеристики як контрольованого, так і неконтрольованого машинного навчання. Він використовує комбінацію позначених і не позначених наборів даних для навчання своїх алгоритмів. Використовуючи обидва типи наборів даних, напівконтрольоване навчання долає недоліки згаданих вище варіантів.

Розглянемо приклад студента коледжу. Студент, який вивчає концепцію під наглядом викладача в коледжі, називається навчанням під наглядом. Під час навчання без нагляду учень самостійно вивчає ту саму концепцію вдома без керівництва вчителя. Тим часом перегляд концепції студентом після навчання під керівництвом викладача в коледжі є напівконтрольованою формою навчання.

1.4 Розпізнавання мовлення

Програмне забезпечення для розпізнавання мовлення визначається як технологія, яка може обробляти мовлення, вимовлене природною мовою, і перетворювати його на читабельний текст із високим ступенем точності за допомогою методів штучного інтелекту (AI), машинного навчання (ML) і природної мови (NLP).

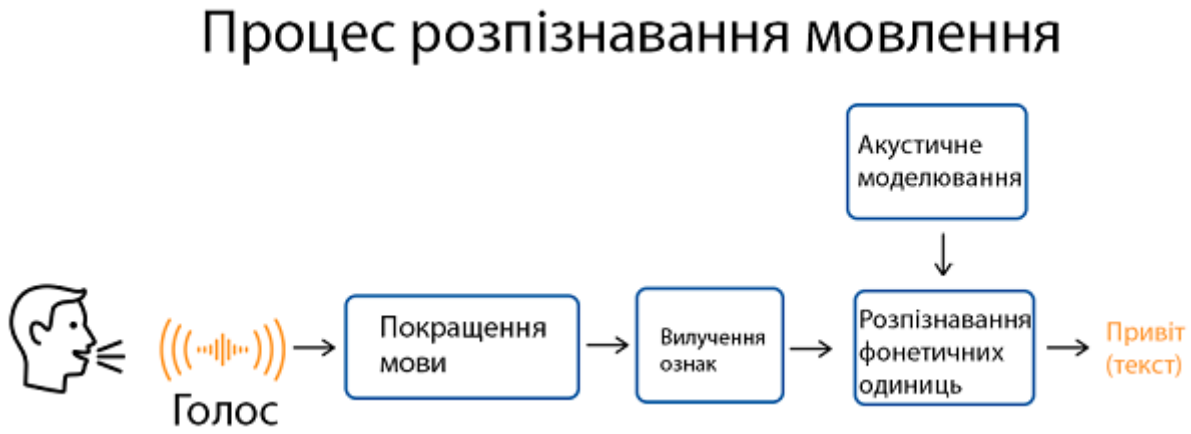


Рисунок 6 – Процес розпізнавання мовлення

Розпізнавання мовлення – це процес перетворення звукових сигналів людини на слова чи інструкції. Розпізнавання мовлення базується на мовленні. Це важливий напрямок досліджень обробки мовних сигналів і розділ розпізнавання образів. Дослідження розпізнавання мови охоплює багато предметних галузей, таких як комп'ютерні технології, штучний інтелект, цифрова обробка сигналів, розпізнавання образів, акустика, лінгвістика та когнітивні науки. Це багатодисциплінарна всеохоплююча область досліджень. На основі дослідницьких завдань з різними обмеженнями виникли різні напрямки досліджень. Відповідно до вимог способу мовлення мовця ці області можна розділити на ізольовані слова, зв'язані слова та безперервне розпізнавання мовлення системи. За ступенем залежності від мовця ці області можна розділити на системи розпізнавання мови для конкретної людини і неконкретної людини. Відповідно до розміру словникового запасу їх можна розділити на системи розпізнавання мовлення з малим словниковим запасом, середнім словниковим запасом, великим словниковим запасом і нескінченним словниковим запасом.

З точки зору моделі розпізнавання мовлення, теорія системи розпізнавання мовлення базується на розпізнаванні образів. Метою розпізнавання мовлення є перетворення вхідної векторної послідовності ознак мовлення в послідовність слів за допомогою фонетичної та лінгвістичної інформації. Відповідно до структури системи розпізнавання мовлення, повна система розпізнавання мовлення включає алгоритм виділення ознак, акустичну модель, а також мовну модель і алгоритм пошуку. Система розпізнавання мови по суті є багатовимірною системою розпізнавання образів . Для різних систем розпізнавання мовлення конкретні методи та прийоми розпізнавання, які використовують люди, різні, але основні принципи однакові. Зібрані мовні сигнали надсилаються до функціївидобуток. Процеси модуля та отримані мовленнєві характеристики надсилаються до модуля бібліотеки моделей. Модуль зіставлення шаблону мовлення ідентифікує мовлення сегмента відповідно до бібліотеки моделей і, нарешті, отримує результат розпізнавання.

1.4.1 Основні характеристики розпізнавання мовлення

Висока точність

Коли машина перетворює виголошену мову в письмовий текст, вона повинна мати можливість робити це з помірною або високою точністю. Неточне розпізнавання не приносить користі та часто суперечить інтуїції продуктивності, оскільки процес виправлення помилок займає більше часу, ніж ручна транскрипція чи введення. Як правило, рівень точності вище 70% вважається «хорошим», тобто програмне забезпечення правильно розпізнає 70 слів із кожних 100 сказаних слів.

Можливості транскрипції

Хоча механізм розпізнавання мовлення може підключатися до зовнішнього інструменту транскрипції, корисно мати дві функції в одній системі. Програмне забезпечення може розуміти та обробляти голосовий ввід, генерувати транскрипцію тексту та подавати його в зручному для читання форматі, доступному для завантаження у вигляді файлів із субтитрами або документів.

Навчання моделі AI та ML

Розпізнавання мовлення базується на складному штучному інтелекті (ШІ) , який перетворює голосові введення у великі обсяги машинозчитуваної інформації. Однією з ключових переваг штучного інтелекту є те, що він може ставати точнішим з кожним сеансом використання, навчаючись на винятках і помилках, які виникають. Це відбувається за допомогою машинного навчання, і потрібно мати можливість навчити програмний AI та модель ML для підвищення точності.

Підтримка розробників

Хоча кілька платформ розпізнавання мовлення готові до використання, слід також шукати підтримку розробників. Це означає, що інтерфейси прикладного програмування (API) повинні бути доступні для вбудовування функціональності в інші програми. Наприклад, розробник може використовувати API розпізнавання мовлення, щоб створити свій галузевий голосовий помічник для пошуку в складних сховищах знань.

Готовність підприємства

Окрім підтримки розробників, підприємства повинні мати можливість використовувати програмне забезпечення для розпізнавання мовлення у своїх бізнес-процесах. Це включає керування документами, голосовий пошук, обробку голосових даних великого обсягу тощо. Крім того, програмне забезпечення має розміщувати та обробляти голосові дані в сумісному центрі обробки даних, який не порушує конфіденційність користувачів і не компрометує конфіденційну корпоративну інформацію.

1.4.2 Аналіз та порівняння існуючих рішень

Alibaba Cloud Intelligent Speech Interaction

Основний китайський хмарний сервіс Alibaba використовує такі технології, як синтез мовлення, розпізнавання голосу та розуміння природної мови, щоб створити свою пропозицію інтелектуальної мовної взаємодії. Наразі він доступний

такими мовами: кантонська китайська, мандаринська китайська, японська, англійська, французька, корейська та індонезійська, але з’являться інші мови.

Основні функції:

- Висока точність: хоча компанія не розкриває точний рівень точності, платформа може самонавчатися.
- Можливості транскрипції: він може обробляти багатомовні транскрипції в режимі реального часу та з попередньо записаних файлів.
- Навчання моделі штучного інтелекту та машинного навчання: користувачі можуть навчити модель, щоб зменшити кількість помилок на 20%.
- Підтримка розробників: пропонує широкий спектр API та посібник розробника.
- Підготовка до роботи на підприємстві: він має готові корпоративні рішення для обслуговування клієнтів, субтитрів у реальному часі та аналізу викликів служби.

Платформа має багато функцій і підходить для розпізнавання коротких речень. Однак крива навчання може бути крутою для компаній, які тільки починають працювати з хмарним середовищем Alibaba.

Ціна починається від 1,00 доларів США за годину для записаних файлів і 1,40 доларів США за годину для розпізнавання мовлення в реальному часі.

Amazon Transcribe

Amazon Transcribe – це програмне забезпечення для розпізнавання мовлення від Amazon Web Services (AWS). У ньому можна легко додати можливості перетворення мовлення в текст за допомогою обробки природної мови. Його можливості дозволяють використовувати аудіовхід, створювати зручні для читання та переглядати стенограми, фільтрувати матеріал для збереження конфіденційності клієнта та підвищувати точність за допомогою налаштування. Transcribe – це хмарна платформа транскрипції.

Основні функції:

- Висока точність: програмне забезпечення забезпечує рівні точності приблизно 80%.
- Можливості транскрипції: створює транскрипції, які легко читати та інтегрувати в бізнес-програми.
- Навчання моделям штучного інтелекту та машинного навчання: він надає десять альтернативних транскрипцій для кожного речення та вивчає ваші дані, підтримуючи вашу спеціальну мовну модель (CLM).
- Підтримка розробників: це надзвичайно зручно для розробників, з навчанням користуванню платформою.
- Готовність для підприємства: він відповідає корпоративним нормам, таким як Закон про перенесення та підзвітність медичного страхування (HIPAA), і підтримує автоматичне редагування вмісту.

Amazon Transcribe приділяє особливу увагу конфіденційності, безпеці та відповідності. Це означає, що діють спеціальні заходи для секторів, які обробляють конфіденційні дані, наприклад охорони здоров’я.

Amazon Transcribe надається безкоштовно протягом 60 хвилин на місяць протягом року та коштує 0,00780 доларів США за хвилину.

Nuance Dragon

Це програмне забезпечення для розпізнавання мовлення було вперше розроблено в 1997 році та придбано багатьма компаніями, доки воно не стало власністю Nuance Communications і згодом Microsoft. Він пропонує рішення ASR для різних випадків використання, включаючи професійні та індивідуальні програми, корпоративні команди, юристів, правоохоронні органи та домашнє використання, охоплюючи програми як для Windows, так і для мобільних середовищ.

Основні функції:

- Висока точність: забезпечує точність до 99%.

- Можливості транскрипції: користувачі можуть скористатися готовим до використання програмним забезпеченням для транскрипції та голосовим керуванням для редагування документів.
- Навчання моделям штучного інтелекту та машинного навчання: він має обмежену підтримку налаштування, але ви можете визначити власні голосові команди.
- Підтримка розробників: пропонує багато ресурсів для розробників, які допоможуть створити чат-ботів, системи обміну повідомленнями та інші програми для розпізнавання мовлення.
- Готовність до корпоративного рівня: корпоративні користувачі можуть інсталювати програмне забезпечення на робочому столі та негайно почати його використовувати.

Nuance Dragon простий у використанні та застосуванні. Він ідеально підходить для бізнес-користувачів. Він також підтримує Citrix, інші віртуалізовані середовища та централізований центр адміністрування.

Ціни починається від 200 доларів США для Dragon Home для Windows, а професійна версія – від 150 доларів США за річну підписку.

Deepgram

Deepgram пропонує автоматичне розпізнавання мовлення з транскрипцією в реальному часі, використовуючи наскрізне глибоке навчання, створене для масштабування. Організації можуть використовувати Deepgram окремо або в поєднанні зі своїм поточним набором технологій, щоб побачити результати за кілька тижнів. Deepgram є партнером NVIDIA, а також стартапом Y Combinator. У жовтні 2021 року він залучив 17,4 мільйона доларів США.

Основні функції:

- Висока точність: забезпечує понад 90% точності за допомогою навчання моделі.

- Можливості транскрипції: в основному він зосереджений на розмовному штучному інтелекті та аналітиці мовлення, але також може бути адаптований для послуг транскрипції.
- Навчання моделям AI та ML: користувачі можуть створювати та навчати власні моделі мовлення всього за кілька тижнів.
- Підтримка розробників: Deepgram пропонує API, комплекти розробки програмного забезпечення (SDK) та інструменти інтеграції для підтримки розробників.
- Готовність підприємства: він надає індивідуальні рішення для підприємств, яким потрібні масштабні рішення ASR.

Deepgram обіцяє найкращу в галузі швидкість транскрипції. Це означає, що ви можете транскрибувати годинний запис приблизно за три секунди.

Ціна на програмне забезпечення починається від 0,0125 доларів США за хвилину.

Google Speech-to-Text API

Google Speech-to-Text – це хмарне програмне забезпечення ASR та API, які працюють на основі складної технології ML компанії. Він підтримує понад 125 мов і колекцію попередньо навчених моделей для певних доменів.

Основні функції:

- Висока точність: має рівень точності 80-85%.
- Можливості транскрипції: він може транскрибувати аудіо на 125+ мовах і варіантах, включаючи попередньо записане аудіо та аудіо в реальному часі.
- Навчання моделям штучного інтелекту та машинного навчання: користувачі можуть навчати модуль із спеціальним словником і працювати в складних звукових умовах.
- Підтримка розробників: ця пропозиція насамперед призначена для розробників, із багатофункціональними API та детальною документацією.

- Готовність підприємства: підприємства можуть використовувати локальну пропозицію Speech-to-Text для забезпечення конфіденційності даних.

Google надає такі унікальні функції, як шумозаглушення, багатоканальне розпізнавання та фільтрація нецензурної лексики. Це значно скорочує навчання моделі та зусилля розробника.

Пропозиція безкоштовна протягом перших 60 хвилин і коштує $0,004 за 15 секунд або більше після цього.

Висновки до розділу

В даному розділі було описано предмет і об’єкт дослідження та було проведено огляд предметної області. Окрім цього було розкрито перспективи використання технології розпізнавання мовлення в проектах.

Було розкрито поняття глибокого навчання, машинного навчання та розпізнавання мовлення. В глибокому навчанні розкрито існуючі методи, використання для розпізнавання мовлення, проведено порівняння з нейронними мережами, розглянуто типи глибоких нейронних мереж а також досягнення в глибокому навчанні. В машинному навчанні розкрито його принцип роботи та типи. Щодо розпізнавання мовлення було розкрито основні характеристики розпізнавання мовлення та проведено аналіз і порівняння популярних існуючих рішень.

2 АНАЛІЗ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура розпізнавання мовлення на основі HMM

Основні компоненти безперервного розпізнавання мовлення з великим словниковим запасом описано рівнянням 2. Форма вхідного аудіосигналу з мікрофона перетворюється на послідовність акустичних векторів фіксованого розміру $Y_{1:T} = y_1, \cdots, y_t$ за допомогою виділення ознак. Потім декодер намагається знайти послідовність слів $w_{1:L} = w_1, \cdots, w_L$, котра породила Y, тобто декодер намагається знайти:

$$\widehat{w} = \arg\max_{w}\{P(w|Y)\} \tag{2}$$

Однак, оскільки $P(w|Y)$ важко змоделювати безпосередньо тому що існують деякі системи, які базуються на дискримінаційних моделях, де $P(w|Y)$ змодельована безпосередньо, а не з використанням генеративних моделей, таких як HMM, де моделюється послідовність спостереження, $p(Y|w)$. Саме тому правило Байєса використовується для перетворення (рівняння 2) в еквівалентну задачу пошуку:

$$\widehat{w} = \arg\max_{w}\{p(Y|w)P(w)\} \tag{3}$$

Імовірність p(Y|w) визначається акустичною моделлю, а попередній P(w) визначається мовною моделлю.

На практиці акустична модель не нормалізується, а мовна модель часто масштабується емпірично визначеною константою та додається штраф за вставку слів, тобто в логарифмічній області загальна ймовірність обчислюється як $\log p(Y|w) + \alpha \log p(Y|w) + \beta|w|$ де α зазвичай знаходиться в діапазоні 8 – 20, а β зазвичай знаходиться в діапазоні 0 – 20.

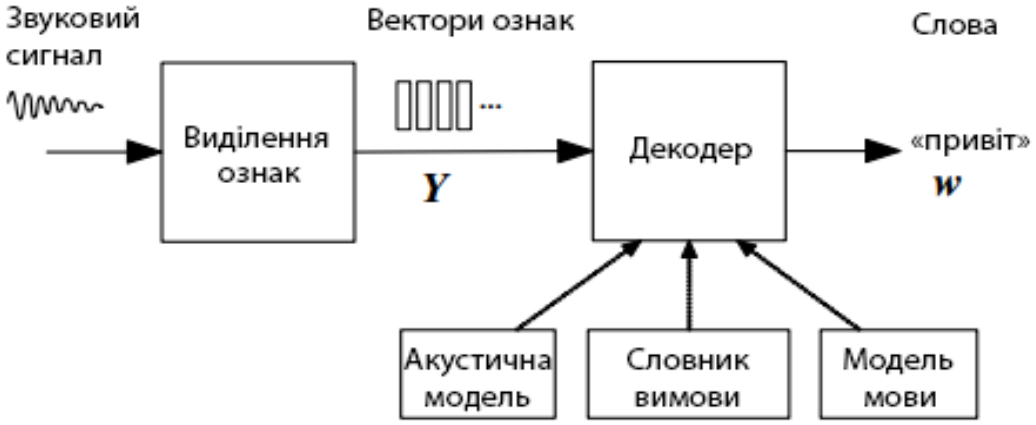


Рисунок 7 – Архітектура розпізнавання мовлення на основі HMM

Основною одиницею звуку, яку представляє акустична модель, є телефон.

Для будь-якого заданого w відповідна акустична модель синтезується за допомогою конкатенації телефонних моделей, щоб отримати слова, як визначено словником вимови. Параметри цих моделей телефонів оцінюються за навчальними даними, що складаються з форм мовлення та їх орфографічної транскрипції. Модель мови, як правило, є моделлю N-грам, у якій ймовірність кожного слова залежить лише від його $N-1$ попередників. Параметри N-грам оцінюються шляхом підрахунку N-кортежів у відповідних текстових корпусах. Декодер працює, шукаючи всі можливі послідовності слів за допомогою скорочення, щоб видалити малоймовірні гіпотези, таким чином зберігаючи пошук доступним. Коли висловлювання досягнуто, виводиться найбільш вірогідна послідовність слів. Крім того, сучасні декодери можуть генерувати решітки, що містять компактне представлення найбільш імовірних гіпотез.

2.1.1 Виділення ознак

Етап виділення ознак прагне забезпечити компактне представлення форми сигналу мовлення. Ця форма повинна звести до мінімуму втрату інформації, яка розрізняє слова, і забезпечувати хорошу відповідність із припущеннями розподілу, зробленими акустичними моделями. Наприклад, якщо для розподілу стан-вихід

використовуються діагональні коваріаційні розподіли Гауса, тоді характеристики повинні бути розроблені як гаусівські та некорельовані.

Вектори ознак зазвичай обчислюються кожні 10 мс з використанням вікна аналізу, що перекривається, приблизно 25 мс. Одна з найпростіших і найбільш широко використовуваних схем кодування базується на частотних кепстральних коефіцієнтах мел (MFCC). Вони генеруються шляхом застосування усіченого дискретного косинусного перетворення (DCT) до логарифмічної спектральної оцінки, обчисленої шляхом згладжування ШПФ із приблизно 20 розділами частоти, розподіленими нелінійно по мовному спектру. Використовувана нелінійна частотна шкала називається шкалою мел, і вона приблизно відповідає відгуку людського вуха. DCT застосовується для згладжування спектральної оцінки та приблизно декореляції елементів ознаки. Після косинусного перетворення перший елемент представляє середнє значення логарифмічної енергії частотних елементів. Іноді це замінюється логарифмічною енергією кадру або повністю видаляється.

Подальші психоакустичні обмеження включені у відповідне кодування, яке називається перцептивним лінійним передбаченням (PLP). PLP обчислює лінійні коефіцієнти передбачення з перцепційно зваженого нелінійно стисненого спектру потужності, а потім перетворює лінійні коефіцієнти передбачення в кепстральні коефіцієнти. На практиці PLP може дати невеликі покращення в порівнянні з MFCC, особливо в шумному середовищі, і, отже, це кодування, якому надають перевагу для багатьох систем.

На додаток до спектральних коефіцієнтів, коефіцієнти регресії першого порядку (дельта) і другого порядку (дельта-дельта) часто додаються в евристичній спробі компенсувати припущення умовної незалежності, зроблене акустичними моделями на основі HMM. Якщо початковий (статичний) вектор ознак є y_t^S, тоді дельта-параметр Δy_t^S визначається як:

$$\Delta y_t^S = \frac{\sum_{i=1}^{n} w_i \left(y_{t+i}^S - y_{t-i}^S \right)}{2 \sum_{i=1}^{n} w_i^2} \tag{4}$$

де n – ширина вікна, а w_i – коефіцієнти регресії. Параметри дельта-дельта, Δy_t^S, виводяться таким же чином, але з використанням відмінностей дельта-параметрів. Коли вони об'єднані разом, вони утворюють вектор ознак y_t,

$$y_t^S = [y_t^{S^T}\ \Delta y_t^{S^T}\ \Delta^2 y_t^{S^T}]^T. \tag{5}$$

Кінцевим результатом є вектор ознак, розмірність якого зазвичай становить близько 40 і який був частково, але не повністю декорельований.

2.1.2 Акустичні моделі HMM (базові-однокомпонентні)

Як зазначалося вище, кожне вимовлене слово w розкладається на послідовність базових звуків K_w, які називаються базовими звуками. Ця послідовність називається її вимовою $q_{1:K_w}^w = q_1, \cdots, q_{K_w}$. Щоб урахувати можливість кількох вимов, ймовірність $p(Y|w)$ може бути обчислена для кількох вимов

$$p(Y|w) = \sum_Q p(Y|Q)P(Q|w), \tag{6}$$

де підсумовування стосується всіх дійсних послідовностей вимови для w, Q є певною послідовністю вимов,

$$P(Q|w) = \prod_{l=1}^{L} P(q^{w_1}|w_1), \tag{7}$$

де кожен q^{w_1} є правильною вимовою для слова w_1. На практиці буде лише дуже невелика кількість альтернативних варіантів вимови для кожного wl, що робить підсумовування в (рівняння 6) легко простежуваним.

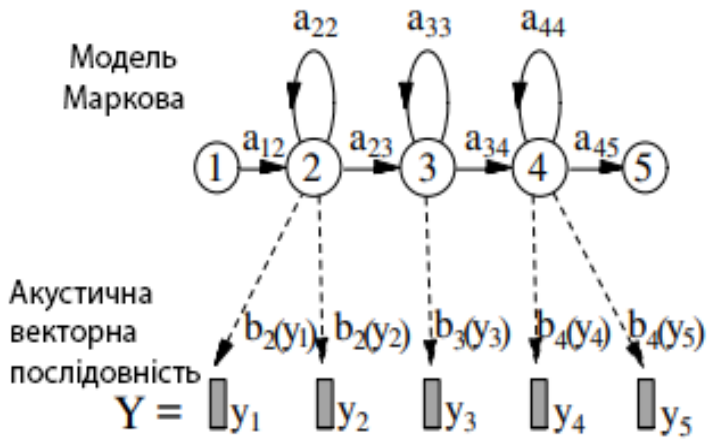


Рисунок 8 – Модель телефону на основі HMM

Кожен базовий телефон q представлений безперервною щільністю HMM у формі, показаний у формулі 3, з параметрами ймовірності переходу $\{a_{ij}\}$ і розподілами вихідного спостереження $\{b_i()\}$. Під час роботи HMM здійснює перехід від свого поточного стану до одного з підключених станів кожного кроку. Ймовірність здійснення конкретного переходу зі стану s_i в стан s_j визначається ймовірністю переходу $\{a_{ij}\}$. При вході в стан вектор ознак генерується з використанням розподілу, пов'язаного зі станом, у який вводиться, $\{b_i()\}$. Ця форма процесу дає стандартні припущення про умовну незалежність для HMM:

- стани є умовно незалежними від усіх інших станів з огляду на попередній стан;
- спостереження є умовно незалежними від усіх інших спостережень, враховуючи стан, який їх породив.

На даний момент для розподілу виходу буде прийнято єдині багатовимірні Гауссани:

$$b_j(y) = N\left(y;\ \mu^j, \Sigma^j\right), \tag{8}$$

де μ^j – середнє значення стану s_j, а Σ^j – його коваріація. Оскільки розмірність акустичного вектора y є відносно високою, коваріації часто мають діагональ.

Враховуючи композицію HMM Q, утворену конкатенацією всіх складових базових телефонів $q^{w_1}, \cdots, q^{w_L}$, тоді акустична правдоподібність визначається як

$$p(Y|Q) = \sum_{0} p(0, Y|Q), \tag{9}$$

де $\theta = \theta_0, \cdots, \theta_{T+1}$ являється послідовністю станів через складену модель і

$$p(\theta, Y|Q) = a_{\theta_0\theta_1} \prod_{t=1}^{T} b_{\theta_t} y_t a_{\theta_t\theta_{t+1}}. \tag{10}$$

У цьому рівнянні θ_0 і θ_{T+1} відповідають входу і виходу без випромінювання, показаним у рівнянні 3. Вони включені, щоб спростити процес об'єднання моделей телефонів для створення слів. Для простоти в подальшому ці невипромінювальні стани будуть проігноровані, а фокус буде зосереджений на послідовності станів $\theta_1, \cdots, \theta_T$.

Параметри акустичної моделі $\lambda = [\{a_{ij}\}, \{b_j()\}]$ можуть бути ефективно оцінені з корпусу навчальних висловлювань за допомогою алгоритму вперед-назад, який є прикладом максимізації очікування (EM). Для кожного висловлювання $Y^r, r = 1, \cdots, R$, довжини T^R знайдено послідовність базових форм HMM, які відповідають послідовності слів у висловлюванні, і створено відповідний складений HMM. На першому етапі алгоритму, етапі E, пряма ймовірність $\alpha_t^{rj} = p(Y_{1:t}^r, \theta_t = s_i; \lambda)$ і зворотна ймовірність $\beta_t^{ri} = p(Y_{t+1:T^r}^r | \theta_t = s_i; \lambda)$ обчислюються за допомогою наступних рекурсій:

$$\alpha_t^{ri} = \left[\sum_i \alpha_{t-1}^{ri} a_{ij} \right] b_j(y_t^r) \tag{11}$$

$$\beta_t^{ri} = \left[\sum_j a_{ij} b_j y_{t+1}^r \beta_{t+1}^{rj}\right], \tag{12}$$

де i і j підсумовуються по всіх станах. Під час виконання цих рекурсій для довгих сегментів мовлення може виникнути недопотік, тому на практиці логарифмічні ймовірності зберігаються, а для уникнення цієї проблеми використовується логарифмічна арифметика.

Враховуючи пряму та зворотну ймовірності, ймовірність того, що модель займе стан s_j у момент часу t для будь-якого даного висловлювання r, дорівнює просто:

$$\gamma_t^{rj} = P\left(\theta_t = s_j | Y^r; \lambda\right) = \frac{1}{P^r} \alpha_t^{rj} \beta_t^{rj}, \tag{13}$$

де $P^r = p(Y^r; \lambda)$. Ці ймовірності заповнення станів, які також називають підрахунками заповнень, представляють м'яке вирівнювання станів моделі з даними, і легко показати, що новий набір параметрів Гауса, визначений як:

$$\hat{\mu}^j = \frac{\Sigma_{r=1}^R \Sigma_{t=1}^{T^r} \gamma_t^{rj} y_t^r}{\Sigma_{r=1}^R \Sigma_{t=1}^{T^r} \gamma_t^{rj}}, \tag{14}$$

$$\hat{\Sigma}^j = \frac{\Sigma_{r=1}^R \Sigma_{t=1}^{T^r} \gamma_t^{rj} \left(y_t^r - \hat{\mu}^j\right)\left(y_t^r - \hat{\mu}^j\right)^T}{\Sigma_{r=1}^R \Sigma_{t=1}^{T^r} \gamma_t^{rj}}, \tag{15}$$

максимізувати вірогідність даних з урахуванням цих вирівнювань. Подібне рівняння повторної оцінки можна вивести для ймовірностей переходу:

$$\hat{a}_{ij} = \frac{\Sigma_{r=1}^R \frac{1}{P^r} \Sigma_{t=1}^{T^r} \alpha_t^{ri} a_{ij} b_j y_{t+1}^r \beta_{t+1}^{rj} y_t^r}{\Sigma_{r=1}^R \Sigma_{t=1}^{T^r} \gamma_t^{rj}}. \tag{16}$$

Це другий або М-крок алгоритму. Починаючи з деякої початкової оцінки параметрів, λ^0, послідовні ітерації алгоритму ЕМ дають набори параметрів $\lambda^1, \lambda^2, \cdots$, які гарантовано покращують вірогідність до деякого локального максимуму. Загальним вибором для початкового набору параметрів λ^0 є призначення глобального середнього значення та коваріації даних вихідним розподілам Гаусса та встановлення рівних усіх ймовірностей переходу. Це дає так звану модель плоского старту.

Цей підхід до акустичного моделювання часто називають моделлю «намистинки на нитці», оскільки всі мовні висловлювання представлені об'єднанням послідовності телефонних моделей разом. Основна проблема полягає в тому, що розкладання кожного словникового слова на послідовність контекстно-незалежних базових телефонів не в змозі охопити дуже великий ступінь контекстно-залежних варіацій, які існують у реальному мовленні. Наприклад, базова форма вимови для «настрою» та «круто» використовуватиме ту саму голосну для «оо», але на практиці реалізації «оо» у двох контекстах дуже відрізняються через вплив попереднього та наступного. приголосний. Контекстно-незалежні моделі телефонів називаються монофонами.

Простий спосіб пом'якшити цю проблему – використовувати унікальну модель телефону для кожної можливої пари лівих і правих сусідів. Отримані моделі називаються трифонами, і якщо є N базових телефонів, є N^3 потенційних трифонів. Щоб уникнути проблем розрідженості даних, повний набір логічних трифонів L можна зіставити зі скороченим набором фізичних моделей P шляхом кластеризації та зв'язування параметрів у кожному кластері. Цей процес відображення проілюстровано на рисунку 9, а зв'язування параметрів – на малюнку 10, де нотація $x - q + y$ позначає трифон, що відповідає телефону q, вимовленому в контексті попереднього звуку x і наступного звуку y. Кожна базова вимова телефону q виводиться шляхом простого пошуку зі словника вимови, потім вони зіставляються з логічними телефонами відповідно до контексту, нарешті, логічні телефони зіставляються з фізичними моделями. Зауважте, що залежність від контексту поширюється через межі слів, і це важливо для охоплення багатьох важливих

фонологічних процесів. Наприклад, /p/ у "stop that" має свій сплеск придушений наступним приголосним.

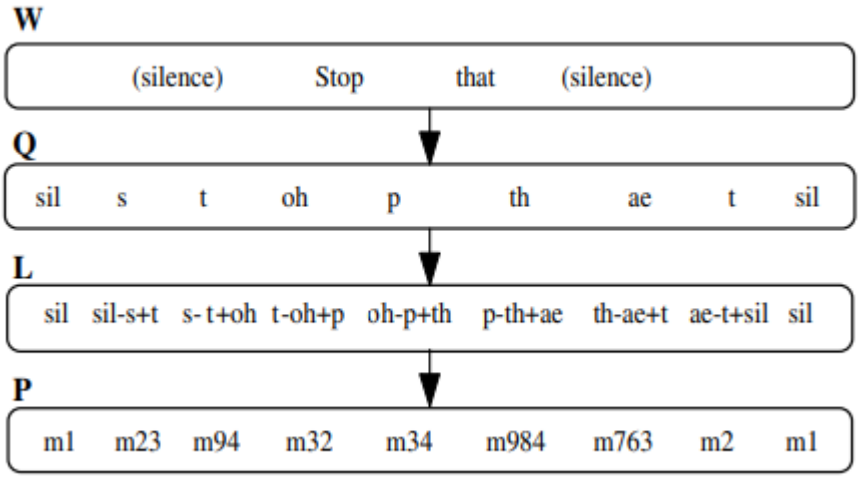


Рисунок 9 – Контекстно-залежне моделювання телефону

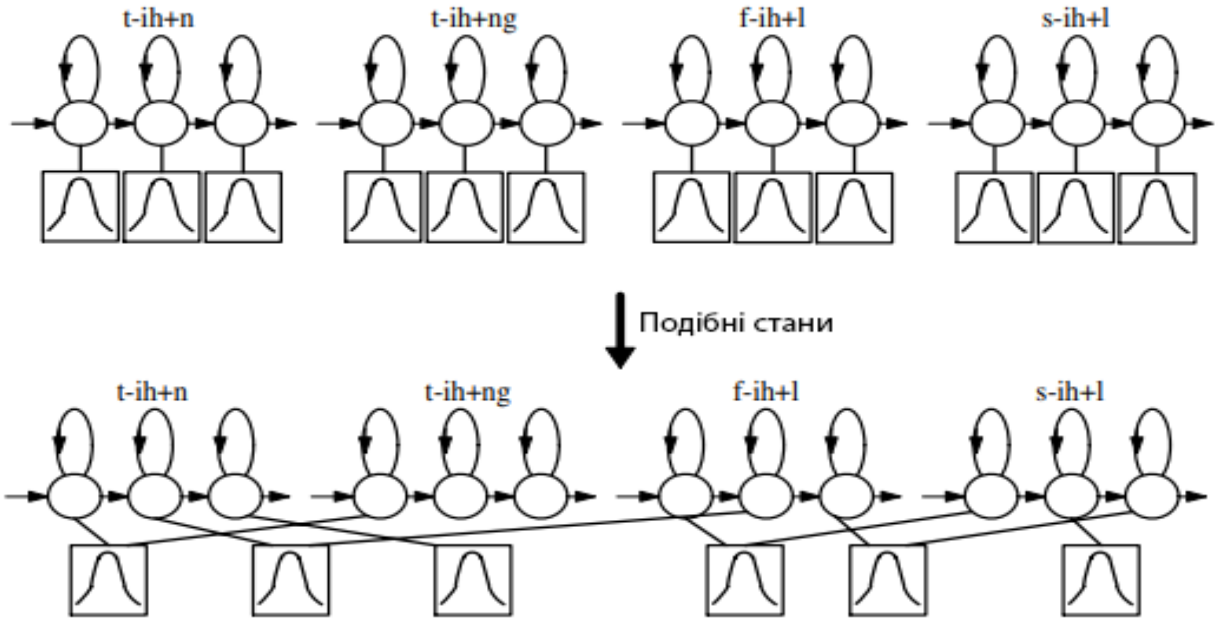


Рисунок 10 – Формування моделей зв'язаних телефонів

Кластеризація логічних і фізичних моделей зазвичай працює на рівні стану, а не на рівні моделі, оскільки це простіше і дозволяє надійно оцінити більший набір фізичних моделей. Вибір станів для пов’язування зазвичай здійснюється за допомогою дерев рішень. Кожна позиція стану кожного телефону q має двійкове дерево, пов’язане з ним. Кожен вузол дерева містить запитання щодо контексту.

Щоб кластеризувати стан i телефону q, усі стани i всіх логічних моделей, отриманих із q, збираються в єдиний пул у кореневому вузлі дерева. Залежно від відповіді на запитання на кожному вузлі пул станів послідовно розбивається, доки всі стани не перейдуть до кінцевих вузлів. Потім усі стани в кожному листовому вузлі зв'язуються, щоб сформувати фізичну модель. Питання в кожному вузлі вибираються із заздалегідь визначеного набору, щоб максимізувати вірогідність навчальних даних, враховуючи остаточний набір прив'язок станів. Якщо вихідні розподіли стану є однокомпонентними гауссами, а лічильники зайнятості станів відомі, то підвищення ймовірності, досягнуте шляхом поділу гауссів у будь-якому вузлі, можна просто обчислити з лічильників і параметрів моделі без посилання на навчальні дані. Таким чином, дерева рішень можуть бути вирощені дуже ефективно за допомогою жадібного ітераційного алгоритму поділу вузлів. Рисунок 11 ілюструє цю деревовидну кластеризацію. На малюнку логічні телефони $s - aw + n$ і $t - aw + n$ обидва будуть призначені кінцевому вузлу 3 і, отже, вони матимуть однаковий центральний стан репрезентативної фізичної моделі.

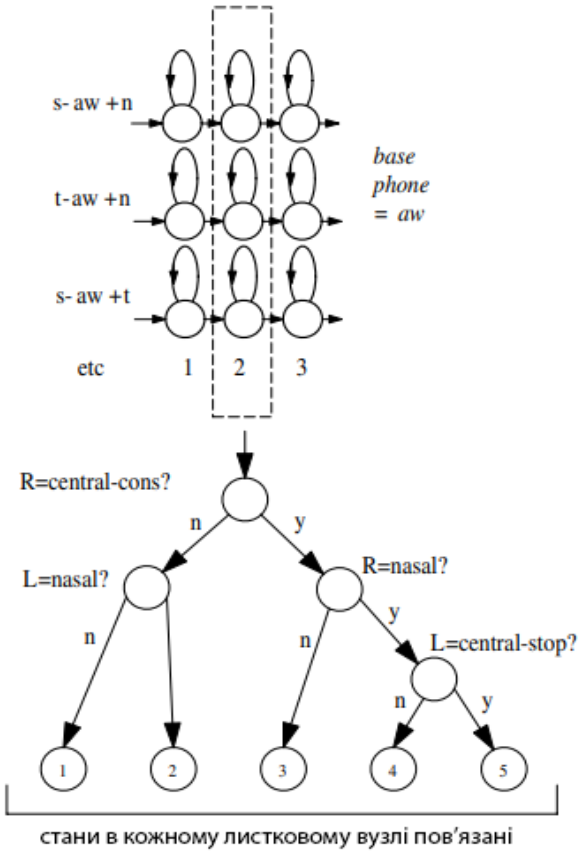


Рисунок 11 – Кластеризація дерева рішень

Поділ станів за допомогою фонетично керованих дерев рішень має кілька переваг. Зокрема, логічні моделі, які потрібні, але взагалі не відображаються в навчальних даних, можуть бути легко синтезовані. Одним із недоліків є те, що перегородка може бути досить грубою. Цю проблему можна зменшити за допомогою так званого м'якого зв'язування. У цій схемі етап постобробки групує кожен стан з одним або двома найближчими сусідами та об'єднує всі їхні гауссіани. Таким чином, окремі моделі Гауса перетворюються на змішані моделі Гауса, утримуючи загальну кількість гаусів у системі постійною.

Підводячи підсумок, основні акустичні моделі сучасного розпізнавача мовлення зазвичай складаються з набору пов'язаних HMM із трьома станами та вихідним розподілом Гауса. Це ядро зазвичай будується в наступні кроки:

- створюється монофонічний набір із рівним стартом, у якому кожен базовий телефон є монофонним одногаусовим HMM із середніми значеннями та коваріаціями, що дорівнюють середньому та коваріації навчальних даних;
- параметри монофонів Гауса переоцінюються за допомогою 3 або 4 ітерацій EM;
- кожен окремий монофон Гауса q клонується один раз для кожного окремого трифона x − q + y, який з'являється в навчальних даних.
- отриманий набір трифонів із навчальними даними знову переоцінюється за допомогою EM, а підрахунки стану зайнятості останньої ітерації зберігаються;
- дерево рішень створюється для кожного стану в кожному базовому телефоні, трифони з навчальними даними відображаються в менший набір зв'язаних трифонів стану та повторно оцінюються за допомогою EM.

Кінцевим результатом є необхідний набір акустичних моделей зв'язаного стану, залежний від контексту.

2.1.3 Моделі мови N-грам

Приорітна ймовірність послідовності слів $w = w_1, \cdots, w_K$, яка вимагається в (Рівняння 3), визначається як:

$$P(w) = \prod_{k=1}^{K} P(w_k | w_{k-1}, \cdots, w_1). \tag{17}$$

Для розпізнавання великого словникового запасу історія умовних слів у (рівняння 17) зазвичай скорочується до N − 1 слів для формування мовної моделі N-грам.

$$P(w) = \prod_{k=1}^{K} P(w_k | w_{k-1}, w_{k-2}, \cdots, w_{k-N+1}). \tag{18}$$

де N зазвичай знаходиться в діапазоні 2-4. Мовні моделі часто оцінюють з точки зору їхньої складності, H, яка визначається як:

$$H = \lim_{K \to \infty} \frac{1}{K} \log_2\big(P(w_1, \cdots, w_K)\big)$$

$$\approx -\frac{1}{K} \sum_{k=1}^{K} \log_2\big(P(w_k | w_{k-1}, w_{k-2}, \cdots, w_{k-N+1})\big),$$

де наближення використовується для мовних моделей N-грам із послідовністю слів кінцевої довжини.

Імовірності N-грам оцінюються з навчальних текстів шляхом підрахунку входжень N-грам для формування оцінок параметрів максимальної правдоподібності (ML). Наприклад, нехай $C(w_{k-2}w_{k-1}w_k)$ представляє кількість входжень трьох слів $w_{k-2}w_{k-1}w_k$ і аналогічно для $C(w_{k-2}w_{k-1})$, тоді:

$$P(w_k|w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})}. \quad (19)$$

Основною проблемою цієї простої схеми оцінки ML є розрідженість даних. Це можна пом'якшити за допомогою комбінації дисконтування та відступу. Наприклад, за допомогою так званого згладжування Каца:

$$P(w_k|w_{k-1}, w_{k-2}) = \begin{cases} d\dfrac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})}, & \text{якщо } 0 < C \leq \text{C}' \\ \dfrac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})}, & \text{якщо C} > C' \\ \alpha(w_{k-1}, w_{k-2})\, P(w_k|w_{k-1}), & \text{інакше} \end{cases} \quad (20)$$

де C' – поріг підрахунку, C – скорочення для $C(w_{k-2}w_{k-1}w_k)$, d – дисконтний коефіцієнт, а α – константа нормалізації. Таким чином, коли кількість N-грамів перевищує певний поріг, використовується оцінка ML. Якщо кількість невелика, використовується та сама оцінка ML, але з незначною знижкою. Дисконтована маса ймовірності потім розподіляється на невидимі N грамів, які апроксимуються зваженою версією відповідної біграми. Цю ідею можна застосувати рекурсивно для оцінки будь-якої розрідженої N-грами в термінах набору відступних ваг і $(N-1)$-грамів. Коефіцієнт дисконтування базується на оцінці Тьюрінга-Гуда $d = (r+1)n_{r+1}/rn_r$, де n_r – це кількість N-грамів, які трапляються рівно r разів у навчальних даних. Існує багато варіацій цього підходу. Наприклад, коли навчальні дані дуже розріджені, згладжування Кнезера-Нея є особливо ефективним.

Альтернативним підходом до надійної оцінки мовної моделі є використання моделей на основі класів, у яких для кожного слова w_k існує відповідний клас c_k. Потім,

$$P(w) = \prod_{k=1}^{K} P(w_k|c_k)p(c_k|c_{k-1}, \cdots, c_{k-N+1}). \quad (21)$$

Що стосується моделей на основі слів, ймовірності класу N-грам оцінюються за допомогою ML, але оскільки існує набагато менше класів (зазвичай кілька сотень) , розрідженість даних є набагато меншою проблемою. Самі класи вибираються для оптимізації ймовірності того, що навчальний набір передбачає модель класу біграми. Можна показати, що коли слово переміщується з одного класу в інший, зміна здивування залежить лише від підрахунку відносно невеликої кількості біграм. Таким чином, можна реалізувати ітераційний алгоритм, який неодноразово сканує словник, перевіряючи кожне слово, щоб побачити, чи перенесення його до іншого класу збільшить вірогідність.

На практиці виявлено, що для достатнього розміру навчальних наборів ефективна мовна модель для додатків із великим словниковим запасом складається зі згладженої 3 або 4-грамової інтерполяції на основі триграми на основі слів.

2.1.4 Декодування та генерація решітки

Як зазначено у вступі до цього розділу, найбільш вірогідна послідовність слів $\widehat{w}$ задана послідовністю векторів ознак $Y_{1:T}$ визначається шляхом пошуку всіх можливих послідовностей станів, що виникають із усіх можливих послідовностей слів, для послідовності, яка, швидше за все, породила спостережувані дані $Y_{1:T}$. Ефективним способом вирішення цієї проблеми є використання динамічного програмування. Нехай $\varphi_t^j = max_\theta\{p(Y_{1:T}, \theta_t = s_j; \lambda)\}$, тобто максимальна ймовірність спостерігати часткову послідовність $Y_{1:T}$ і потім перебувати в стані s_j у момент часу t, заданому моделлю параметри λ. Цю ймовірність можна ефективно обчислити за допомогою алгоритму Вітербі:

$$\varphi_t^j = \max_i\{\varphi_{t-1}^j a_{ij}\}b_j y_t. \tag{22}$$

Він ініціалізується встановленням φ_0^j до 1 для початкового стану входу без випромінювання та 0 для всіх інших станів. Тоді ймовірність найімовірнішої послідовності слів визначається як $max_j\{\varphi_T^j\}$ і якщо кожне рішення про

максимізацію записується, зворотне відстеження дасть необхідну найкращу відповідність послідовності стану/слова.

На практиці пряма реалізація алгоритму Вітербі стає некеровано складною для безперервного мовлення, де необхідно враховувати топологію моделей, обмеження мовної моделі та необхідність обмежувати обчислення. Особливо проблематичними є мовні моделі N-грам і трифонні контексти кросвордів, оскільки вони значно розширюють простір пошуку. Щоб впоратися з цим, було розроблено кілька різних архітектурних підходів. Для декодування Вітербі простір пошуку може бути або обмежений підтриманням кількох гіпотез паралельно, або він може динамічно розширюватися в ході пошуку. В якості альтернативи можна застосувати зовсім інший підхід, коли підхід алгоритму Вітербі в ширину замінюється пошуком в глибину. Це породжує клас розпізнавачів, які називаються стековими декодерами. Однак вони можуть бути дуже ефективними, оскільки вони повинні порівнювати гіпотези різної довжини, їхні характеристики пошуку під час виконання може бути важко контролювати. Нарешті, нещодавні досягнення в технології зважених кінцевих перетворювачів дозволяють інтегрувати всю необхідну інформацію (акустичні моделі, вимову, ймовірності мовної моделі тощо) в єдину дуже велику, але дуже оптимізовану мережу. Цей підхід забезпечує як гнучкість, так і ефективність, і тому надзвичайно корисний як для досліджень, так і для практичних застосувань.

Незважаючи на те, що декодери призначені в основному для пошуку рішення (Рівняння 22), на практиці відносно просто згенерувати не лише найвірогіднішу гіпотезу, а й N-найкращий набір гіпотез. N зазвичай знаходиться в діапазоні 100-1000. Це надзвичайно корисно, оскільки дозволяє багаторазово проходити дані без обчислювальних витрат на повторне вирішення (Рівняння 22) з нуля. Компактною та ефективною структурою для зберігання цих гіпотез є решітка слів.

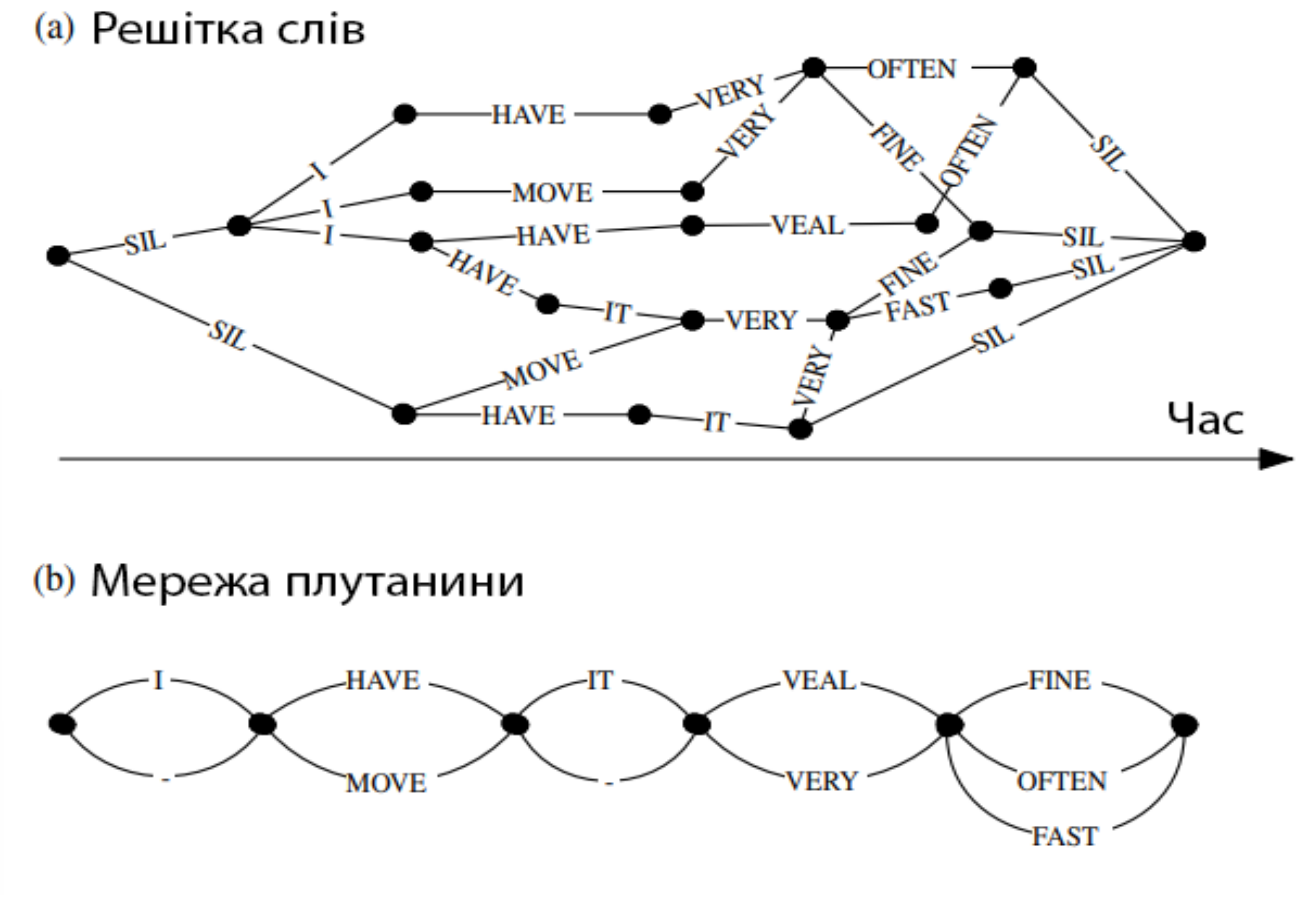


Рисунок 12 – Приклад решітки та мережі плутанини

Решітка слів складається з набору вузлів, що представляють моменти часу, і набору охоплюючих дуг, що представляють гіпотези слів. Приклад показано на рисунку 12 частина (a). На додаток до ідентифікаторів слів, показаних на малюнку, кожна дуга також може містити інформацію про бали, таку як бали акустичної та мовної моделі.

Решітки надзвичайно гнучкі. Наприклад, їх можна переоцінити, використовуючи їх як мережу розпізнавання вхідних даних, і їх можна розширити, щоб дозволити переоцінку мовною моделлю вищого порядку. Їх також можна стиснути в дуже ефективне представлення, яке називається мережею плутанини. Це показано на рисунку 12 (b), де мітки дуг «-» вказують на нульові переходи. У мережі плутанини вузли більше не відповідають окремим точкам часу, натомість вони просто накладають обмеження на послідовність слів. Таким чином, паралельні дуги в мережі змішування не обов'язково відповідають одному і тому ж акустичному сегменту. Однак передбачається, що більшу частину часу перекриття

є достатнім, щоб паралельні дуги можна було розглядати як конкуруючі гіпотези. Мережа плутанини має властивість, що для кожного шляху через початкову решітку існує відповідний шлях через мережу плутанини. Кожна дуга в мережі плутанини несе апостеріорну ймовірність відповідного слова w. Це обчислюється шляхом знаходження ймовірності зв'язку w у решітці за допомогою процедури прямого-назад, підсумовування всіх входжень w, а потім нормалізації таким чином, щоб усі конкуруючі дуги слів у мережі плутанини дорівнювали одиниці. Мережі плутанини можуть бути використані для декодування з мінімальною помилкою слова (приклад декодування з мінімальним ризиком Байєса (MBR), щоб забезпечити оцінку достовірності та для об'єднання виходів різних декодерів.

2.2 Розширення для покращення продуктивності ASR

2.2.1 Динамічні байєсовські мережі

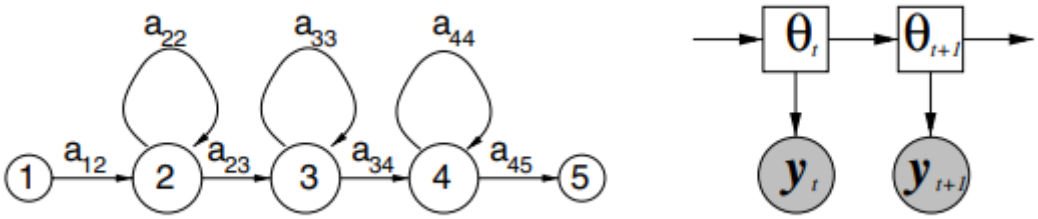


Рисунок 13 – Типова телефонна топологія HMM (ліворуч) і динамічна байєсовська мережа (праворуч)

В «Архітектурі розпізнавача на основі HMM» HMM описується як генеративна модель, яка для типового телефону має три стани випромінювання, як знову показано на рис. 13. Також на рис. 13 показано альтернативне, додаткове графічне представлення, яке називається динамічною байєсівською мережею (DBN), яке підкреслює умовні залежності моделі та яке є особливо корисним для опису різноманітних розширень базової структури HMM. У використаній тут нотації ДБН квадрати позначають дискретні змінні; кола безперервних змінних; затінення вказує на спостережувану змінну; і відсутність затінення неспостережуваної змінної. Відсутність дуги між змінними свідчить про умовну незалежність. Таким чином, рис. 13. показує, що спостереження, створені HMM, є умовно незалежними, враховуючи неспостережуваний, прихований стан, який його породив.

Одним із бажаних атрибутів використання DBN є те, що можна просто показати розширення з точки зору того, як вони змінюють припущення умовної незалежності моделі. Існує два підходи, які можна комбінувати, щоб розширити структуру HMM на рис. 13: додавання додаткових неспостережуваних змінних; і додавання додаткових дуг залежностей між змінними.

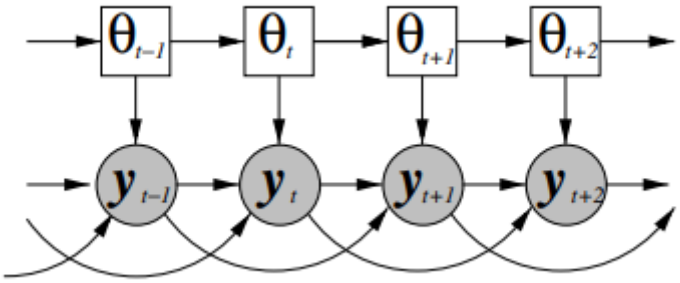


Рисунок 14 – Динамічні байєсівські мережі для прихованих марковських моделей і HMM з векторними лінійними предикторами

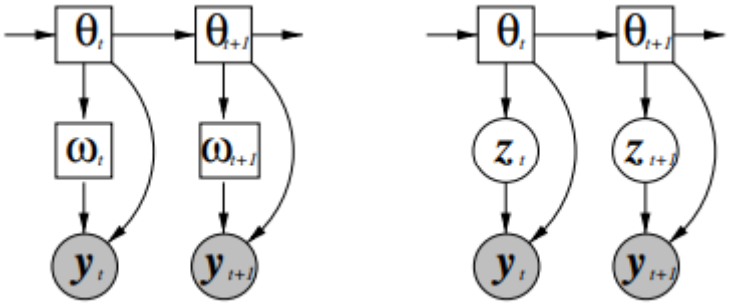


Рисунок 15 – Динамічні байєсовські мережі для моделей суміші Гаусса (ліворуч) і факторно-аналізованих моделей (праворуч)

Приклад додавання додаткових дуг, в даному випадку між спостереженнями, показано на рис. 14. Тут розподіл спостережень залежить від попередніх двох спостережень на додачу до стану, який його породив. Цей DBN описує HMM з явним моделюванням часової кореляції, векторними предикторами та прихованими моделями Маркова. Незважаючи на те, що цей підхід є цікавим напрямком удосконалення HMM, він ще не був прийнятий у основних найсучасніших системах.

Замість додавання дуг можна додати додаткові неспостережувані, приховані або приховані змінні. Щоб обчислити ймовірності, ці приховані змінні потім повинні бути маргіналізовані таким же чином, як послідовність неспостережуваних станів для HMM. Для безперервних змінних це вимагає безперервного інтегралу за всіма значеннями прихованої змінної, а для дискретних змінних – підсумовування за всіма значеннями. Дві можливі форми латентної змінної показані на рис. 15. В

обох випадках тимчасові залежності моделі, де дискретні стани є умовно незалежними від усіх інших станів з урахуванням попереднього стану, залишаються незмінними, що дозволяє використовувати стандартні процедури декодування Вітербі. У першому випадку спостереження залежать від дискретної прихованої змінної. Це DBN для HMM із розподілом вихідного стану моделі суміші Гауса (GMM). У другому випадку спостереження залежать від безперервної латентної змінної. Це DBN для HMM з факторно-аналізованими коваріаційними матрицями.

2.2.2 Моделі суміші Гауса

Одним із найпоширеніших розширень стандартних HMM є моделювання розподілу стану-виходу як моделі суміші. В Архітектурі розпізнавача на основі HMM для моделювання розподілу стан-вихід використовувався єдиний розподіл Гауса. Тому ця модель припускає, що спостережувані вектори ознак є симетричними та унімодальними. На практиці це трапляється рідко. Наприклад, розбіжності в мовцях, акцентах і статі зазвичай створюють кілька режимів у даних. Щоб вирішити цю проблему, єдиний гаусівський розподіл стану-виходу можна замінити сумішшю гауссів, яка є дуже гнучким розподілом, здатним моделювати, наприклад, асиметричні та мультимодальні розподілені дані.

Моделі сумішей можна розглядати як додавання додаткової дискретної латентної змінної до системи. Імовірність стану s_j тепер отримується шляхом підсумовування всіх компонентів ймовірностей, зважених за їхніми попередніми:

$$b_j(y) = \sum_{m=1}^{M} c_{jm} N\left(y; \mu^{jm}, \Sigma^{jm}\right). \tag{23}$$

де c_{jm} – апріорна ймовірність для компонента m стану s_j. Ці пріоритети задовольняють стандартні обмеження для дійсної функції ймовірної маси (PMF):

$$\sum_{m=1}^{M} c_{jm} = 1, c_{jm} \geq 0. \tag{24}$$

Це приклад загальної методики моделювання суміші. Кожен із M компонентів моделі суміші є функцією щільності ймовірності Гауса (PDF).

Оскільки до акустичної моделі було додано додаткову приховану змінну, форму описаного раніше ЕМ-алгоритму необхідно змінити. Замість того, щоб розглядати повний набір даних у термінах спостережень за станом, використовуються спостереження за станом/компонентом. Тоді оцінка параметрів моделі виконується за тією ж формою, що й для випадку з одним компонентом. Для середнього гаусового компонента m стану s_j, s_{jm},

$$\hat{\mu}^{jm} = \frac{\Sigma_{r=1}^{R}\Sigma_{t=1}^{T^r}\gamma_t^{rjm} y_t^r}{\Sigma_{r=1}^{R}\Sigma_{t=1}^{T^r}\gamma_t^{rjm}}, \tag{25}$$

де $\gamma_t^{rjm} = P(\theta_t = s_{jm}|Y^r;\lambda)$ є ймовірністю того, що компонент m стану s_j породив спостереження в момент часу t послідовності Y^r. R – кількість тренувальних послідовностей. Пріори компонентів оцінюються подібно до ймовірностей переходу.

При використанні GMM для моделювання розподілу стану-виходу часто застосовують дисперсію. Це запобігає тому, що дисперсії системи стають занадто малими, наприклад, коли компонент моделює дуже малу кількість щільно упакованих спостережень. Це покращує узагальнення.

Використання GMM збільшує витрати на обчислення, оскільки при використанні логарифмічної арифметики для обчислення ймовірності GMM потрібна серія логарифмічних додавання. Для підвищення ефективності включено лише компоненти, які забезпечують «розумний» внесок у загальну ймовірність. Для подальшого зменшення обчислювального навантаження ймовірність GMM

може бути просто апроксимована максимумом за всіма компонентами (зваженими за попередніми значеннями).

Крім того, необхідно визначити кількість компонентів на стан в системі. Для цього було прийнято низку підходів, у тому числі дискримінаційних схем. Найпростішим є використання однакової кількості компонентів для всіх станів системи та використання утримуваних даних для визначення оптимальних чисел. Популярний альтернативний підхід базується на інформаційному критерії Байєса (BIC). Інший підхід полягає в тому, щоб зробити кількість компонентів, призначених стану, функцією кількості спостережень, призначених цьому стану.

2.2.3 Прогнозування даних

В Архітектурі для розпізнавача на основі HMM динамічні перший і другий диференціальні параметри, так звані дельта та параметри дельта-дельта, були додані до параметрів статичних ознак, щоб подолати обмеження припущення умовної незалежності, пов'язаного з HMM. Крім того, передбачалося, що DCT приблизно декорелює вектор ознак, щоб покращити апроксимацію діагональної коваріації та зменшити розмірність вектора ознак.

Також можна використовувати підходи, керовані даними, для декореляції та зменшення розмірності функцій. Стандартним підходом є використання лінійних перетворень, тобто:

$$y_t = A_{[p]}\hat{y}_t, \tag{26}$$

де $A_{[p]}$ – лінійне перетворення $p \times d$, d – розмір вихідного вектора ознак $\hat{y}_t$ – розмір перетвореного вектора ознак y_t.

Детальна форма векторної трансформації ознак, як ця залежить від ряду варіантів. По-перше, необхідно визначити конструкцію $\hat{y}_t$ і, якщо потрібно, мітки класу, які будуть використовуватися, наприклад, мітки телефону, стану або компонента Гауса. По-друге, має бути визначена розмірність p прогнозованих

даних. Нарешті, необхідно вказати критерій, який використовується для оцінки $A_{[p]}$.

Важливим питанням під час оцінювання прогнозу є те, чи слід використовувати мітки класів спостережень, чи ні, тобто чи прогноз слід оцінювати контрольованим чи неконтрольованим способом. Загалом контрольовані підходи дають кращі прогнози, оскільки тоді можна оцінити перетворення для покращення розрізнення між класами. Навпаки, у підходах без контролю можна використовувати лише загальні атрибути спостережень, такі як відхилення. Однак контрольовані схеми зазвичай обчислюються дорожче, ніж неконтрольовані схеми. Для кожної мітки класу потрібні такі статистичні дані, як середні та коваріаційні матриці, тоді як неконтрольовані схеми фактично мають лише один клас.

Об’єкти, які проектуються, зазвичай базуються на стандартних векторах об’єктів MFCC або PLP. Однак обробка параметрів дельта і дельта-дельта може відрізнятися. Один із підходів полягає в тому, щоб з’єднати сусідні статичні вектори разом, щоб сформувати складений вектор із зазвичай 9 кадрів:

$$\hat{y}_t = [y_{t-4}^{sT} \quad \cdots \quad y_t^{sT} \quad \cdots \quad y_{t+4}^{sT}]^T, \tag{27}$$

Інший підхід полягає в розширенні статичних, дельта і дельта-дельта параметрів динамічними параметрами третього порядку (різниця дельта-дельта). Обидва використовувалися для систем розпізнавання мовлення з великим словниковим запасом. Розмірність прогнозованого вектора ознак часто визначається емпірично, оскільки існує лише один параметр для налаштування. У порівнянні ряду можливих визначень класу, телефону, стану або компонента, порівнювалися для контрольованих схем проектування. Вибір класу важливий, оскільки перетворення намагається забезпечити, щоб кожен із цих класів був якомога більш відокремленим від усіх інших. Дослідження показало, що ефективність використання класів рівня стану та компонента була подібною, і обидва були кращими, ніж мітки вищого рівня, такі як телефони або слова. На

практиці переважна більшість систем використовують класи рівня компонентів, оскільки, як обговорюватиметься далі, це покращує припущення діагональних коваріаційних матриць.

$$F_{pca}(\lambda) = \log\left(\left|A_{[p]}\tilde{\Sigma}_g A_{[p]}^T\right|\right), \tag{28}$$

де $\hat{\Sigma}_g$ – загальна або глобальна коваріаційна матриця вихідних даних, $\hat{y}_t$. Це вибирає ортогональні проекції даних, які максимізують загальну дисперсію в прогнозованому підпросторі. Простий вибір підпросторів, які дають великі дисперсії, не обов'язково дає підпростори, які розрізняють класи. Щоб вирішити цю проблему, можна використовувати контрольовані підходи, такі як критерій лінійного дискримінантного аналізу (LDA). Метою LDA є збільшення відношення дисперсії між класами до середньої дисперсії в межах класу для кожного виміру. Цей критерій можна виразити так

$$F_{1da}(\lambda) = \log\left(\frac{A_{[p]}\tilde{\Sigma}_b A_{[p]}^T}{A_{[p]}\tilde{\Sigma}_w A_{[p]}^T}\right), \tag{29}$$

де $\tilde{\Sigma}_b$ – коваріаційна матриця між класами, а $\tilde{\Sigma}_w$ – середня коваріаційна матриця всередині класу, де кожен окремий компонент Гауса зазвичай вважається окремим класом. Цей критерій дає ортонормальне перетворення таким чином, що середня коваріаційна матриця всередині класу є діагоналізованою, що має покращити припущення діагональної коваріаційної матриці.

Критерій LDA може бути додатково уточнений за допомогою фактичних матриць коваріації класів замість використання середніх значень. Однією з таких форм є гетероскедастичний дискримінантний аналіз (HDA), де максимізується наступний критерій:

$$F_{hda}(\lambda) = \sum_{m} \gamma^{m} \log\left(\frac{A_{[p]}\tilde{\Sigma}_{b}A_{[p]}^{T}}{A_{[p]}\tilde{\Sigma}^{m}A_{[p]}^{T}}\right), \tag{30}$$

де γ^{m} – загальна апостеріорна ймовірність заповнення для компонента m, а $\tilde{\Sigma}^{m}$ – його коваріаційна матриця у вихідному просторі, задана $\tilde{y}_{t}$. У цьому перетворенні матриця A не має обмежень бути ортонормальною, і цей критерій також не допомагає декорелювати дані, пов'язані з кожним компонентом Гауса. Тому його часто використовують у поєднанні з глобальним напівзв'язаним перетворенням (також відомим як лінійне перетворення максимальної правдоподібності (MLLT)), описаним у наступному розділі. Альтернативним розширенням LDA є гетероскедастична LDA (HLDA). Це змінює критерій LDA у (рівняння 29) подібно до HDA, але тепер оцінюється перетворення для повного простору ознак, а не лише для розмірів, які потрібно зберегти. Параметри перетворення HLDA можна знайти в стилі ML, як якщо б вони були параметрами моделі, як описано в Архітектурі для розпізнавача на основі HMM. Важливим розширенням цього перетворення є гарантія того, що розподіли для всіх вимірів, які потрібно видалити, мають бути однаковими. Це досягається шляхом зв'язування параметрів, пов'язаних із цими розмірами, щоб забезпечити їх ідентичність. Таким чином, ці розміри не дадуть дискримінаційної інформації, тому їх не потрібно зберігати для розпізнавання. Тоді критерій HLDA можна виразити як:

$$F_{h1da}(\lambda) = \sum_{m} \gamma^{m} \log\left(\frac{|A|^{2}}{diag\left(\left|A_{[d-p]}\tilde{\Sigma}_{g}A_{[d-p]}^{T}\right|\right)diag\left(\left|A_{[p]}\tilde{\Sigma}^{m}A_{[p]}^{T}\right|\right)}\right), \tag{31}$$

звідки A:

$$A = \begin{bmatrix} A_{[p]} \\ A_{[d-p]} \end{bmatrix}, \tag{32}$$

Є дві важливі відмінності між HLDA і HDA. По-перше, HLDA дає найкращу проекцію, одночасно генеруючи найкраще перетворення для покращення апроксимації діагональної коваріаційної матриці. Навпаки, для HDA необхідно додати окреме декорелююче перетворення. Крім того, HLDA дає модель для повного простору функцій, тоді як HDA моделює лише корисні (непрогнозовані) розміри. Це означає, що кілька підпросторових проекцій можна використовувати з HLDA, але не з HDA.

Хоча схеми на кшталт HLDA перевершують за продуктивністю такі підходи, як LDA, вони дорожчі з точки зору обчислень і вимагають більше
пам'ять. Статистичні дані матриці повної коваріації для кожного компонента потрібні для оцінки перетворення HLDA, тоді як для LDA потрібне лише середнє значення всередині та між матрицями коваріації класу. Це робить непрактичними проекції HLDA з просторів об’єктів великого розміру з великою кількістю компонентів. Одним компромісом, який іноді використовується, є дотримання проекції LDA за допомогою декорелюючого глобального напівзв’язаного перетворення.

2.2.4 Коваріаційне моделювання

Однією з причин використання DCT-перетворення та деяких схем проекцій, розглянутих у попередньому розділі, є декореляція вектора ознак таким чином, щоб апроксимація діагональної коваріаційної матриці стала розумною. Це важливо, оскільки загалом використання гаусів повної коваріації у великих словникових системах було б непрактичним через величезний розмір набору моделей. Навіть у малих системах обмеження навчальних даних часто перешкоджають використанню повних коваріацій. Крім того, обчислювальна вартість використання повних коваріаційних матриць становить $O(d^2)$ порівняно з $O(d)$ для діагонального випадку, де d є розмірністю вектора ознак. Щоб вирішити обидві ці проблеми, були розроблені структуровані коваріаційні та прецизійні (обернені коваріаційні) матричні представлення. Це дозволяє покращити коваріаційне

моделювання з дуже невеликими накладними витратами з точки зору пам’яті та обчислювального навантаження.

Структуровані коваріаційні матриці

Стандартна форма структурованої коваріаційної матриці виникає в результаті факторного аналізу, і DBN для цього було показано в рівнянні 25. Тут коваріаційна матриця для кожного гауссового компонента m представлена (залежність від стану вилучено для ясності):

$$\Sigma^m = A_{[p]}^{mT} A_{[p]}^m + \Sigma_{diag}^m, \tag{33}$$

де $A_{[p]}^m$ – специфічна для компонента, $p \times d$, матриця навантаження, а $\Sigma(m)\, diag$ – специфічна для компонента діагональна коваріаційна матриця. Наведений вище вираз ґрунтується на факторному аналізі, який дозволяє оцінювати кожну з компонент Гауса окремо за допомогою ЕМ. Факторно проаналізовані HMM узагальнюють це для підтримки як зв’язування кількох компонентів матриць навантаження, так і використання ГММ для моделювання латентного простору змінних z в рівнянні 25.

Хоча структуровані коваріаційні матриці цієї форми зменшують кількість параметрів, необхідних для представлення кожної коваріаційної матриці, обчислення ймовірностей залежить від оберненої коваріаційної матриці, тобто матриці точності, і це все одно буде повна матриця. Таким чином, HMM, проаналізовані фактором, все ще несуть витрати на декодування, пов’язані з повними коваріаціями.

Матриці структурованої точності

Більш обчислювально ефективний підхід до структуризації коваріації полягає в моделюванні зворотної коваріаційної матриці, а загальна форма для цього:

$$\Sigma^{m-1} = \sum_{i=1}^{B} v_i^m S_i, \tag{34}$$

де v^m – вектор питомої ваги компонента Гауса, який визначає внесок від кожної з B глобальних позитивних напіввизначених матриць, S_i. Однією з бажаних властивостей цієї форми моделі є її обчислювальна ефективність під час декодування, оскільки:

$$\begin{aligned} &\log\big(N(y; \mu^m, \Sigma^m)\big) \\ &\quad = -\frac{1}{2}\log((2\pi)^d|\Sigma^m|) \\ &\quad -\frac{1}{2}\sum_{i=1}^{B} v_i^m (y-\mu^m)^T S_i (y-\mu^m) \\ &\quad = -\frac{1}{2}\log((2\pi)^d|\Sigma^m|) \\ &\quad -\frac{1}{2}\sum_{i=1}^{B} v_i^m (y^T S_i y - 2\mu^{mT} S_i y + \mu^{mT} S_i \mu^m) \end{aligned} \tag{35}$$

Таким чином, замість того, щоб мати обчислювальну вартість повної коваріаційної матриці, можна кешувати такі терміни, як $S_i y$ і $y^T S_i y$, які не залежать від компонента Гауса.

Одним із найпростіших і найефективніших структурованих представлень коваріації є напівзв’язана коваріаційна матриця (STC). Це особлива форма прецизійної матричної моделі, де кількість основ дорівнює числу вимірів ($B = d$), а основи є симетричними та мають ранг 1 (тому можна виразити як $S_i = a_{[i]}^T a_{[i]}$, $a_{[i]}$ – і-й рядок A). У цьому випадку ймовірності компонентів можна обчислити за допомогою:

$$N(y; \mu^m, \Sigma^m) = |A| N\left(A_y; A\mu^m, \Sigma^m_{diag}\right), \quad (36)$$

де Σ^{m-1}_{diag} – діагональна матриця, утворена з v^m і

$$\Sigma^{m-1} = \sum_{i=1}^{d} v_i^m a_{[i]}^T a_{[i]} = A\Sigma^{m-1}_{diag} A^T, \quad (37)$$

Матрицю A іноді називають напівзв'язаним перетворенням. Однією з причин простоти використання систем STC є те, що вага для кожного виміру є просто зворотною дисперсією для цього виміру. Процедура навчання систем STC така (після накопичення стандартної статистики матриці повної коваріації EM):

- (1) ініціалізувати перетворення $A^0 = I$; встановити Σ^{m0}_{diag} рівним коваріаційним матрицям поточної моделі; і встановити $k = 0$;
- (2) оцінка A^{k+1} задана Σ^{mk}_{diag};
- (3) встановити $\Sigma^{m(k+1)}_{diag}$ на дисперсію компонента для кожного виміру за допомогою A^{k+1};
- (4) перейти до (2), якщо не збігається або не досягнуто максимальної кількості ітерацій.

Під час розпізнавання декодування є дуже ефективним, оскільки $A\mu^M$ зберігається для кожного компонента, а перетворені ознаки Ay_t кешуються для кожного моменту часу. Таким чином, майже немає збільшення часу декодування порівняно зі стандартними системами діагональної коваріаційної матриці.

Системи STC можна зробити більш потужними за допомогою кількох напівзв’язаних перетворень. На додаток до STC, інші типи структурованого коваріаційного моделювання включають підпросторову обмежену точність і середнє (SPAM), суміші зворотних коваріацій і розширені лінійні перетворення максимальної правдоподібності (EMLLT).

Моделювання тривалості HMM

Імовірність $d_j(t)$ залишатися в стані s_j для t послідовних спостережень у стандартному HMM визначається як:

$$d_j(t) = a_{jj}^{t-1}(1 - a_{jj}), \tag{38}$$

де a_{jj} – ймовірність самопереходу стану s_j. Таким чином, стандартний HMM моделює ймовірність заповнення стану як експоненціальне зменшення з часом, і це, очевидно, погана модель тривалості.

Враховуючи цю слабкість стандартного HMM, очевидним удосконаленням є введення явної моделі тривалості так, щоб цикли самопереходу в рівні 3 були замінені явним розподілом ймовірностей $d_j(t)$ Відповідними варіантами для $d_j(t)$ є розподіл Пуассона та гамма-розподіл.

Хоча використання цих явних розподілів може, безсумнівно, точніше моделювати тривалість сегментів стану, вони додають дуже значні накладні витрати до складності обчислень. Це пояснюється тим, що отримані моделі більше не мають властивості Маркова, і це перешкоджає ефективному пошуку можливих вирівнювання станів. Наприклад, у цих так званих прихованих напівмарківських моделях (HSMM) пряма ймовірність, наведена в рівні 11, стає:

$$\alpha_t^{rj} = \sum_{\tau} \sum_{i,i \neq j} \alpha_{t-\tau}^{ri} a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(y_{t-\tau+l}^r), \tag{39}$$

Необхідність підсумовувати всі можливі тривалості станів збільшила складність на коефіцієнт $O(D^2)$, де D є максимально дозволеною тривалістю стану, а зворотна ймовірність і декодування Вітербі страждають від такого ж збільшення складності.

На практиці явне моделювання тривалості цієї форми пропонує деякі невеликі покращення продуктивності в малих словникових програмах і мінімальні покращення у великих словникових програмах. Типовий досвід останнього полягає

в тому, що зі збільшенням точності розподілу телефонного виходу покращення, отримані від моделювання тривалості, стають незначними. З цієї причини та його обчислювальної складності довгострокове моделювання рідко використовується в сучасних системах.

Висновки до розділу

В даному розділі було проаналізована архітектура розпізнавання мовлення на основі HMM, де було роз’яснено принципи виділення ознак, акустичних базових-однокомпонентних моделей HMM, модель мови N-грам та декодування з генерацією решітки. Також було проаналізовані розширення для покращення продуктивності ASR, зокрема:

- динамічні байєсовські мережі;
- моделі суміші Гауса;
- проекції функцій;
- коваріаційне моделювання.

3 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір інструментарію для розробки

Основним набором інструментів для створення та перевірки роботи алгоритму було обрано:

- NodeJS;
- Docker;
- Typescript;
- Nx.Js;
- Angular.

3.1.1 Node.Js

Node.js – це кросплатформне середовище виконання JavaScript із відкритим кодом і бібліотека для запуску веб-додатків поза браузером клієнта. Node.js використовується для створення веб-додатків на стороні сервера, і він ідеально підходить для програм, що інтенсивно обробляють дані, оскільки використовує асинхронну подійо-керовану модель.

3.1.2 Docker

Docker – це платформа контейнеризації з відкритим вихідним кодом, яка використовується для розробки, розгортання та керування програмами в легких віртуалізованих середовищах, які називаються контейнерами.

В основному він використовується як платформа розробки програмного забезпечення для розробки розподілених програм , які ефективно працюють у різних середовищах. Зробивши систему програмного забезпечення агностиком, розробникам не доведеться турбуватися про проблеми сумісності. Пакування програм в ізольовані середовища (контейнери) також полегшує розробку, розгортання, підтримку та використання програм.

Оскільки Docker використовує віртуалізацію для створення контейнерів для зберігання програм, концепція може здатися схожою на віртуальні машини. Хоча обидва представляють ізольовані віртуальні середовища, які використовуються для розробки програмного забезпечення, існують важливі відмінності між контейнерами та віртуальними машинами . Найголовнішою відмінністю є те, що контейнери Docker легші, швидші та ресурсоефективніші , ніж віртуальні машини.

3.1.3 TypeScript

TypeScript – це строго типізована, об’єктно-орієнтована, скомпільована мова. Він був розроблений Андерсом Хейлсбергом (розробником C#) у Microsoft. TypeScript – це і мова, і набір інструментів. TypeScript – це типізований наднабір JavaScript, скомпільований у JavaScript. Іншими словами, TypeScript – це JavaScript плюс деякі додаткові функції.

3.1.4 Nx.Js

Nx – це оркестровник, який застосовує інструменти Webpack, Vite, SWC та Vite найефективнішим способом з вбудованою підтримкою монорепозиторіїв.

Nx пропонує повний спектр, дозволяючи поступове та легке впровадження для надання більш повного та попередньо налаштованого досвіду на основі плагінів. Це не лише допомагає налаштувати монорепозиторій та виконувати завдання, але й керує розробниками протягом життєвого циклу розробки.

3.1.5 Angular

Angular – це інтерфейсний фреймворк на стороні клієнта з відкритим вихідним кодом на основі JavaScript , який використовується для створення спеціальних програм у HTML, CSS і Typescript. AngularJS був запущений у 2009 році Міско Гевері та Адамом Абронсом як проект у Google. Це інтерфейсний фреймворк JavaScript, розроблений для спрощення створення web-динамічних додатків завдяки функції MVC (Model-View-Controller).

3.2 Налаштування інструментів розробки

Спочатку потрібно створити docker-compose.yml в котрому додати сервіс з налаштуваннями для додатку.

docker-compose.yml

```yaml
version: '3.9'

services:
  app:
    build:
      dockerfile: ./.docker/app.dockerfile
      args:
        - JAVA_VERSION=11
        - GRADLE_VERSION=7.6
        - ANDROID_SDK_VERSION=6200805
        - ANDROID_BUILD_TOOLS_VERSION=30.0.2
        - ANDROID_PLATFORMS_VERSION=30
        - NODE_VERSION=18.12.1-alpine
    volumes:
      - ./:/home/node/app

```

Рисунок 16 – Вміст файлу docker-compose.yml

Сервіс буде створюватися на основі dockerfile, також за допомогою аргументів в середину контейнера буде передано аргументи і прокинуто робочий каталог з вихідним кодом додатку.

Контейнер app буде створюватися на основі офіційного образу node, версія буде братися з аргументів. Також будуть встановлені необхідні інструменти для розробки додатку.

```
ARG NODE_VERSION

FROM node:${NODE_VERSION}

ARG JAVA_VERSION

RUN yarn global add npm@9.1.3 nx@15.2.4 create-nx-workspace@15.2.4
RUN apk add openjdk${JAVA_VERSION} unzip curl gcompat

USER node
```

Рисунок 17 – Базові інструкції для створення контейнеру додатка

Для можливості компіляції додатку на Android в контейнер буде додано Gradle та Android Sdk Tools.

```
# Install Gradle
ARG GRADLE_VERSION

ENV PATH_GRADLE=${PATH_ANDROID}/gradle
ENV GRADLE_ZIP_FILENAME=gradle-${GRADLE_VERSION}-bin.zip
ENV PATH=$PATH:${PATH_GRADLE}/gradle-${GRADLE_VERSION}/bin

RUN mkdir $PATH_GRADLE &&\
  curl -sL https://downloads.gradle-dn.com/distributions/${GRADLE_ZIP_FILENAME} -o $GRADLE_ZIP_FILENAME &&\
  unzip -d $PATH_GRADLE $GRADLE_ZIP_FILENAME &&\
  rm ${PATH_ANDROID}/${GRADLE_ZIP_FILENAME}

# Install Android SDK tools
ARG ANDROID_SDK_VERSION
ARG ANDROID_BUILD_TOOLS_VERSION
ARG ANDROID_PLATFORMS_VERSION

ENV ANDROID_HOME=/home/node/.android/sdk
ENV PATH=$PATH:${ANDROID_HOME}/tools/bin:${ANDROID_HOME}/platform-tools
ENV ANDROID_SDK_ZIP_FILENAME=commandlinetools-linux-${ANDROID_SDK_VERSION}_latest.zip

RUN curl -sL https://dl.google.com/android/repository/${ANDROID_SDK_ZIP_FILENAME} -o $ANDROID_SDK_ZIP_FILENAME &&\
  unzip $ANDROID_SDK_ZIP_FILENAME &&\
  mkdir $ANDROID_HOME && mv tools $ANDROID_HOME &&\
  yes | $ANDROID_HOME/tools/bin/sdkmanager --sdk_root=$ANDROID_HOME --licenses &&\
  $ANDROID_HOME/tools/bin/sdkmanager --sdk_root=$ANDROID_HOME "platform-tools" \
    "build-tools;${ANDROID_BUILD_TOOLS_VERSION}" "platforms;android-${ANDROID_PLATFORMS_VERSION}" &&\
  rm ${PATH_ANDROID}/${ANDROID_SDK_ZIP_FILENAME}

WORKDIR /home/node/app
```

Рисунок 18 – Встановлення Gradle та Android Sdk Tools в контейнер

Після цього для створення образу додатку необхідно запустити команду *docker compose build*.

Рисунок 19 – Створення контейнера додатку

Тепер можна під єднатися до терміналу додатку за допомогою команди *docker compose run --rm app sh*.

Далі для створення додатку потрібно створити монорепозиторій за допомогою команди *npx create-nx-workspace* в терміналі контейнера додатку.

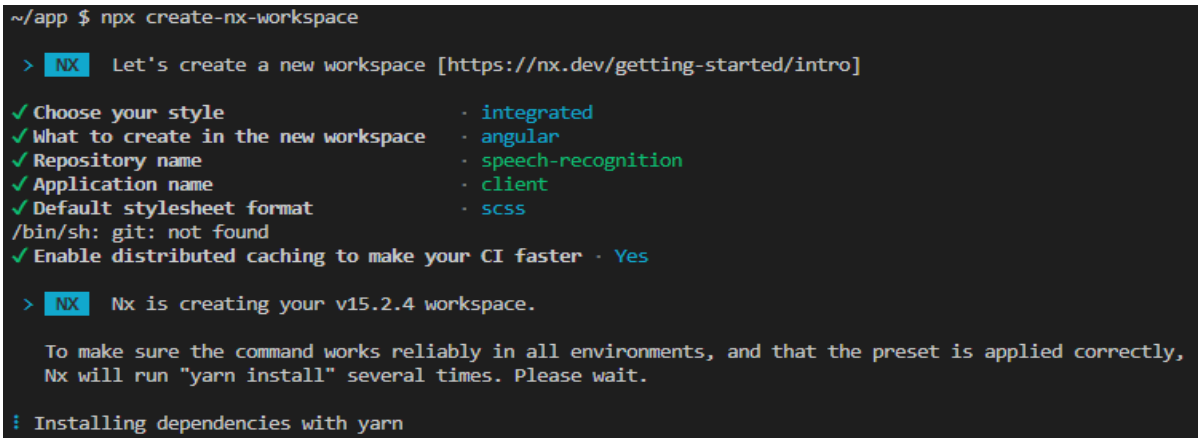


Рисунок 20 – Створення монорепозиторію

Після чого буде створено монорепозиторій з створеним додатком, котрий буде використовувати фреймворк Angular.

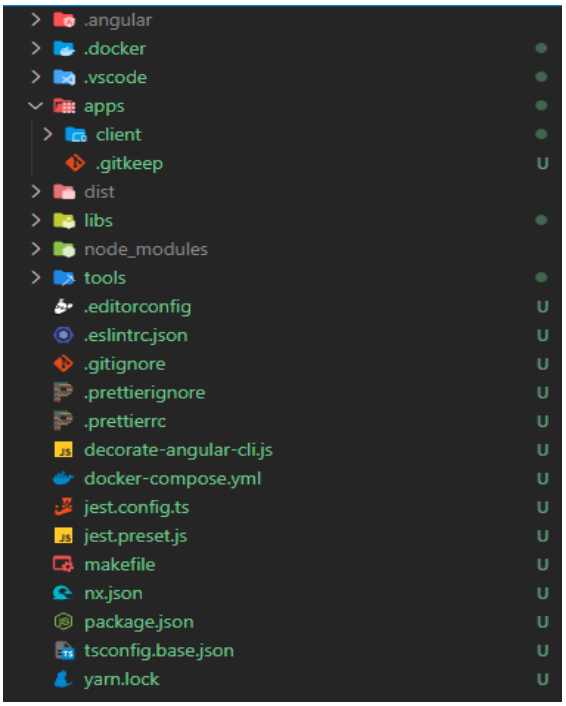


Рисунок 21 – Структура проекту монорепозиторію

Для того щоб створити бібліотеку потрібно запустити наступну команду *nx g lib client* в терміналі контейнера додатку. Щоб запустити web-додаток Angular необхідно запустити команду *docker compose run -p 4200:4200 --rm app nx run client:serve:development -- --host=0.0.0.0*.

3.3 Розробка алгоритму розпізнавання мовлення

Алгоритм повинен на вхід отримувати звуковий сигнал, далі цей звуковий сигнал повинен передаватися в рушії розпізнавання мовлення та паралельно оброблюватися ними. В результаті буде отримано масив строк з розпізнаними командами.

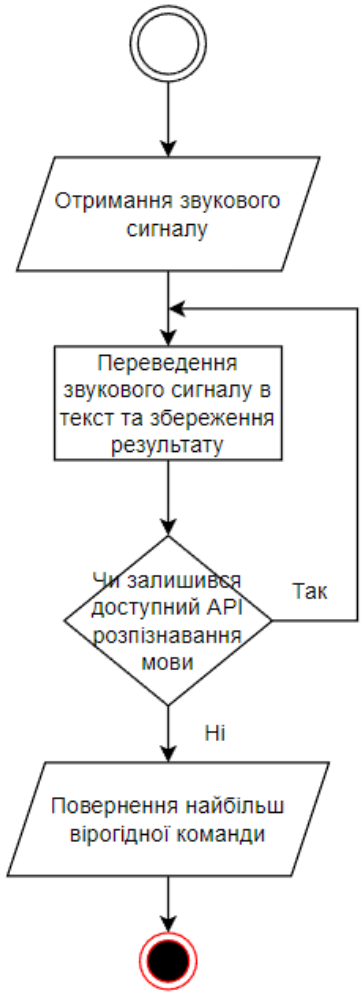


Рисунок 22 – Блок-схема розробленого алгоритму

Для рушія розпізнавання мовлення потрібно створити інтерфейс в котрому будуть описані необхідні поля та методи, а саме поля:

- enabled – поле для перевірки чи увімкнений рушій;
- isSuported – чи підтримується рушій в системі де запущений додаток;
- recognizedPhrase – фраза, котра була розпізнана,

та методи:

- activate() – увімкнути рушій;
- deactivate() – вимкнути рушій;
- start() – запустити процес розпізнавання;
- stop() – зупинити процес розпізнавання.

Інтерфейс реалізовано через абстрактний клас, код інтерфейсу зображено на рис 23.

```
export abstract class SpeechRecognitionCoreRepository {
  protected _enabled = true;
  protected _isSupported = true;
  protected _started = false;
  protected _recognizedPhrase = '';

  public get enabled(): boolean {
    return this._enabled;
  }

  public get isSupported(): boolean {
    return this._isSupported;
  }

  public get recognizedPhrase(): string {
    return this._recognizedPhrase;
  }

  public activate(): void {
    this._enabled = true;
    this.stop();
    if (this._started) {
      this.start();
    }
  }

  public deactivate(): void {
    this._enabled = false;
    this.stop();
  }

  public abstract start(): Promise<boolean>;

  public abstract stop(): Promise<boolean>;
}
```

Рисунок 23 – Код інтерфейсу для рушія розпізнавання мовлення

Наступним кроком йде реалізація сервісу для роботи з рушіями розпізнавання мовлення. В сервісі реалізовано два метода:

- start() – метод для одночасного запуску всіх увімкнених та підтримуваних рушіїв розпізнавання мовлення;
- stop() – метод для припинення процесу розпізнавання мовлення та отримання масиву розпізнаних фраз.

Сервіс також повинен в конструкторі отримувати масив рушіїв розпізнавання мовлення, кожен з яких унаслідується від раніше описаного інтерфейсу, а також мати масив рушіїв котрі вже запущені, для того щоб зручніше працювати з активними рушіями розпізнавання мовлення.

```
import { SpeechRecognitionCoreRepository } from '../repositories';

export class SpeechRecognitionService {
  protected _startedSpeechRecognitionCoreRepositories: SpeechRecognitionCoreRepository[] =
    [];

  public constructor(
    private _speechRecognitionCoreRepositories: SpeechRecognitionCoreRepository[]
  ) {}

  public async start(): Promise<void> {
    await Promise.all(
      this._speechRecognitionCoreRepositories.map(
        (speechRecognitionCoreRepository: SpeechRecognitionCoreRepository) => {
          if (
            speechRecognitionCoreRepository.isSupported &&
            speechRecognitionCoreRepository.enabled
          ) {
            speechRecognitionCoreRepository.start();
            this._startedSpeechRecognitionCoreRepositories.push(
              speechRecognitionCoreRepository
            );
          }
        }
      )
    );
  }

  public async stop(): Promise<string[]> {
    const recognitionPhrases: string[] = await Promise.all(
      this._speechRecognitionCoreRepositories.map(
        async (
          speechRecognitionCoreRepository: SpeechRecognitionCoreRepository
        ) => {
          await speechRecognitionCoreRepository.stop();
          return speechRecognitionCoreRepository.recognizedPhrase.toLowerCase();
        }
      )
    );

    this._startedSpeechRecognitionCoreRepositories = [];

    return Promise.resolve(recognitionPhrases);
  }
}
```

Рисунок 24 – Код сервісу для розпізнавання мовлення

Далі йде реалізація рушіїв розпізнавання мовлення. Спочатку створюємо адаптер для API Webkit Speech Recognition. Для перевірки чи підтримується рушій використовується наступна умова зображена на рис 25.

```
this._isSupported = 'webkitSpeechRecognition' in window;
if (!this.isSupported) {
  return;
}
```

Рисунок 25 – Код перевірки чи підтримується рушій Webkit Speech Recognition

При створені об’єкту webkitSpeechRecognition потрібно вказати мову, котру необхідно розпізнавати, чи потрібно щоб розпізнавання працювало в

безперервному режимі та чи потрібно отримувати проміжні результати. Також необхідно підписатися на події result і end.

```
this._core = new webkitSpeechRecognition();

this._core.lang = 'uk-UA';
this._core.continuous = true;
this._core.interimResults = true;

this._core.addEventListener('result', (event: SpeechRecognitionEvent) => {
  this._recognizedPhrase = Array.from(event.results)
    .map((result: SpeechRecognitionResult) => result[0])
    .map((result) => result.transcript)
    .join('');

  if (event.results[0].isFinal) {
    this._waitForResult = false;
  }
});

this._core.addEventListener('end', () => {
  if (this._isStoppedAutomatically) {
    this._core?.stop();
    this._core?.start();
    this._isStoppedAutomatically = true;
  }
});
```

Рисунок 26 – Код створення об'єкту webkitSpeechRecognition

Для використання Deepgram необхідно зареєструватися на офіційному сайті та отримати API токен, після можна за допомогою WebSocket відправляти аудіо з мікрофону і отримувати результат розпізнавання. Мова розпізнавання вказується в get параметрі.

```
const socket = new WebSocket(
  'wss://api.deepgram.com/v1/listen?language=uk',
  ['token', '<token>']
);

socket.onopen = () => {
  if (this._mediaRecorder) {
    this._mediaRecorder.addEventListener('dataavailable', (event) => {
      if (socket.readyState === WebSocket.OPEN) {
        socket.send(event.data);
      }
    });
  }
};

socket.onmessage = (message) => {
  const recieved = JSON.parse(message.data);
  const transcript = recieved.channel.alternatives[0].transcript;
  if (transcript) {
    this._recognizedPhrase = transcript;
    if (recieved.is_final) {
      this._waitForResult = false;
    }
  }
};
```

Рисунок 27 – Код створення об'єкту Deepgram

Для перевірки роботи алгоритму спочатку потрібно ініціалізувати сервіс та передати в нього створенні адаптери для розпізнавання мовлення. В Angular є зручний інструмент для Dependency Injection, тому буде використано саме його.

```
@NgModule({
  declarations: [RootComponent],
  imports: [BrowserModule],
  providers: [
    WebkitSpeechRecognitionCoreAdapter,
    DeepgramSpeechRecognitionCoreAdapter,
    {
      multi: true,
      provide: SpeechRecognitionCoreRepository,
      useExisting: WebkitSpeechRecognitionCoreAdapter,
    },
    {
      multi: true,
      provide: SpeechRecognitionCoreRepository,
      useExisting: DeepgramSpeechRecognitionCoreAdapter,
    },
    {
      provide: SpeechRecognitionService,
      deps: [SpeechRecognitionCoreRepository],
    },
  ],
  bootstrap: [RootComponent],
})
export class AppModule {}
```

Рисунок 28 – Налаштування залежностей

Після чого можна використовувати сервіс в компоненті та працювати з ним як зображено на рис. 29. Всі залежності вже будуть налаштовані.

```typescript
@Component({
  selector: 'source-root',
  templateUrl: './root.component.html',
  styleUrls: ['./root.component.scss'],
})
export class RootComponent {
  public get webkitRecognizedPhrase(): string {
    return this._webkitSpeechRecognitionCoreAdapter.recognizedPhrase;
  }

  public get deepgramRecognizedPhrase(): string {
    return this._deepgramSpeechRecognitionCoreAdapter.recognizedPhrase;
  }

  public get mdiMicrophoneOutline(): string {
    return mdiMicrophoneOutline;
  }

  public constructor(
    private _s: SpeechRecognitionService,
    private _webkitSpeechRecognitionCoreAdapter: WebkitSpeechRecognitionCoreAdapter,
    private _deepgramSpeechRecognitionCoreAdapter: DeepgramSpeechRecognitionCoreAdapter
  ) {}

  public onPointerUpHandler(): void {
    setTimeout(() => {
      this._s.stop();
    }, 1000);
  }

  public onPointerDownHandler(): void {
    this._s.start();
  }
}
```

Рисунок 29 – Використання сервісу в компоненті

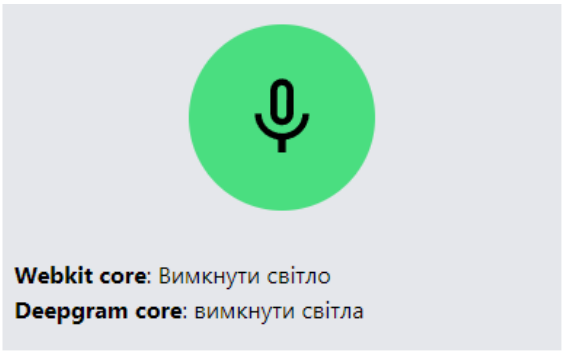


Рисунок 30 – Результат обробки голосової команди "вимкнути світло"

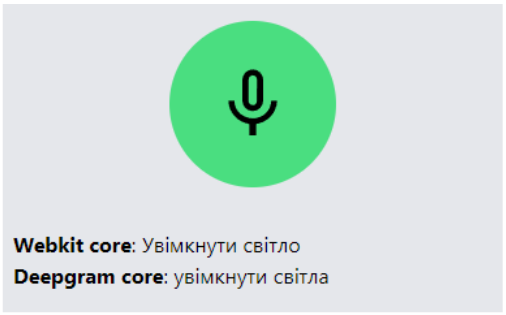


Рисунок 31 – Результат обробки голосової команди "Увімкнути світло"

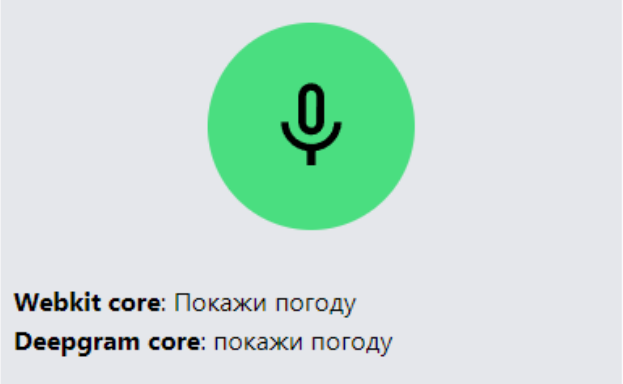


Рисунок 32 – Результат обробки голосової команди "Покажи погоду"

В перших двох перевірках є не критичні відмінності в розпізнаванні, а в третій обидва рушія розпізнавання мовлення однаково розпізнали команду, але навіть з похибками алгоритм за рахунок більшої кількості варіантів розпізнавання команд дає стовідсотковий результат розпізнавання команди, котра буде надіслана до контролера. При умові якщо використовувався тільки один рушій розпізнавання мовлення то команди прийшлося б повторювати, допоки команда не розпізнається правильно.

ВИСНОВКИ

У ході виконання магістерської роботи було запропоноване рішення на основі інтеграції нейронних мереж, що збільшує точність розпізнавання голосових команд.

В першому розділі «Аналіз проблематики та аналогів» було розкрито поняття глибокого навчання, машинного навчання та розпізнавання мовлення. Наведені методи для розпізнавання мовлення та проведено порівняльний аналіз існуючих інструментів розпізнавання мовлення, а саме: Alibaba Cloud Intelligent Speech Interaction, Amazon Transcribe, Nuance Dragon, Deepgram, Google Speech-to-Text API. Вони добре виконують поставлену задачу, проте мають деякі недоліки: проблема розпізнавання української мови, є повністю платні рішення, є рішення котрі залежать від платформи.

В другому розділі «Аналіз інформаційного забезпечення» детально розглянуто архітектуру розпізнавання мовлення на основі прихованих моделей Маркова а також які розширення використовуються для покращення розпізнавання мовлення. Для майбутнього додатку було обрано рекурентну нейронну мережу, побудовану на архітектурі прихованих моделей Маркова.

В третьому розділі «Опис розробки програмного забезпечення» наведені короткі відомості про обраний інструментарій, описано процес налаштування інструментів для розробки та описані основні моменти запропонованого алгоритму розпізнавання мовлення. Рушіями розпізнавання мовлення було обрано Deepgram та Webkit Speech Recognition. Розроблений алгоритм дозволяє інтегрувати необмежену кількість розпізнавачів мовлення котрі будуть паралельно працювати щоб точніше розпізнавати голосові команди. В подальшому цей алгоритм можна перетворити на модуль та інтегрувати в систему керування розумним будинком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kanishka Rao, Ha ̧sim Sak, and Rohit Prabhavalkar, “Exploring architectures, data, and units for streaming end-to-end speech recognition with rnn-transducer,” in2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2017, pp. 193–199.
2. Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu, “Contextnet: Improving convolutional neural networks for automatic speech recognition with global context,”arXiv preprint arXiv:2005.03191, 2020.
3. Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, “Speech recognition with deep recurrent neural networks,” in2013 IEEE international conference on acoustics, speech and signal processing. Ieee, 2013, pp. 6645–6649.
4. Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in2015IEEE international conference on acoustics, speech and signal processing(ICASSP). IEEE, 2015, pp. 5206–5210.
5. Yisen Wang, Xuejiao Deng, Songbai Pu, and Zhiheng Huang, “Residual convolutional ctc networks for automatic speech recognition,”arXiv preprintarXiv:1702.07793, 2017.
6. Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, “Attention-based models for speech recognition,”arXivpreprint arXiv:1506.07503, 2015.↩
7. Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Transactions on audio, speech, and language processing, vol. 22, no. 10, pp. 1533–1545, 2014.
8. Shigeki Karita, Nelson Enrique Yalta Soplin, Shinji Watanabe, Marc Del-croix, Atsunori Ogawa, and Tomohiro Nakatani, “Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language

model integration,”Proc. Interspeech 2019, pp. 1408–1412, 2019.↩

John S Garofolo, “Timit acoustic phonetic continuous speech corpus,” Linguistic Data Consortium, 1993, 1993.

9. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, “Attention is all you need,”arXiv preprint arXiv:1706.03762, 2017.
10. Yangyang Shi, Yongqiang Wang, Chunyang Wu, Christian Fuegen, FrankZhang, Duc Le, Ching-Feng Yeh, and Michael L Seltzer, “Weak-attention suppression for transformer-based speech recognition,”arXiv preprintarXiv:2005.09137, 2020
11. Neil Zeghidour, Qiantong Xu, Vitaliy Liptchinsky, Nicolas Usunier, GabrielSynnaeve, and Ronan Collobert, “Fully convolutional speech recognition,”arXiv e-prints, pp. arXiv–1812, 2018.
12. Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., “State-of-the-art speech recognition with sequence-to-sequence models,” in2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 4774–4778.
13. William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016, pp. 4960–4964.
14. Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert, “Sequence-to-sequence speech recognition with time-depth separable convolutions,”arXivpreprint arXiv:1904.02619, 2019.
15. Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur, “Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus,” inProceedings of the International Workshop on Spoken Language Translation(IWSLT), Heidelberg, Germany, December 2013.

16. Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, “Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss,”inICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 7829–7833.
17. Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Ji-ahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., “Conformer: Convolution-augmented transformer for speech recognition,”arXiv preprint arXiv:2005.08100, 2020.
18. Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde, “Jasper: An end-to-end convolutional neural acoustic model,”arXiv preprintarXiv:1904.03288, 2019.
19. Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, RuomingPang, et al., “Streaming end-to-end speech recognition for mobile devices,”inICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 6381–6385.31
20. Ha ̧sim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, “Fast and accurate recurrent neural network acoustic models for speech recognition,”arXiv preprint arXiv:1507.06947, 2015.
21. Douglas B Paul and Janet Baker, “The design for the wall street journal-based csr corpus,” in Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992, 1992.
22. Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer, “Transformers with convolutional context for asr,”arXiv preprintarXiv:1904.11660, 2019.
23. Linhao Dong, Shuang Xu, and Bo Xu, “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in2018 IEEE Inter-national Conference on Acoustics, Speech and Signal Processing (ICASSP).IEEE, 2018, pp. 5884–5888.

24. Takaaki Hori, Jaejin Cho, and Shinji Watanabe, “End-to-end speech recognition with word-based rnn language models,” in2018 IEEE Spoken LanguageTechnology Workshop (SLT), 2018, pp. 389–396.
25. Chengyi Wang, Yu Wu, Yujiao Du, Jinyu Li, Shujie Liu, Liang Lu, Shuo Ren, Guoli Ye, Sheng Zhao, and Ming Zhou, “Semantic mask for transformer-based end-to-end speech recognition,”arXiv preprintarXiv:1912.03010, 2019.↩

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА

«РОЗРОБКА СИСТЕМИ ГОЛОСОВОГО УПРАВЛІННЯ РОЗУМНИМ БУДИНКОМ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ»

Виконав: студент групи ПДМ-62 Ліщук Ігор Валерійович

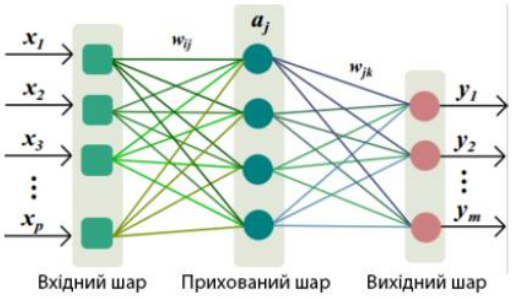
Керівник: к.т.н., доц., доцент кафедри ІПЗ, доктор філософії (PhD) Дібрівний О.А

Київ - 2022

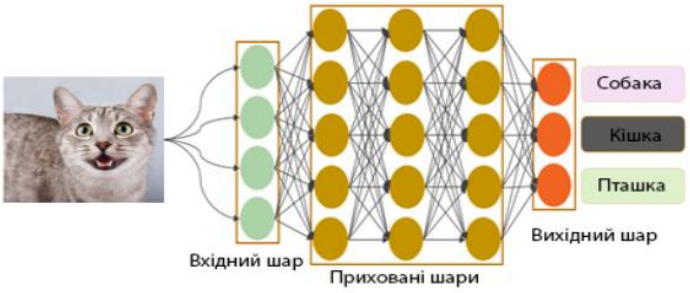
АНАЛІЗ ІТ-РІШЕНЬ

Система автоматизації	Головна перевага	Потрібен хаб?	Голосове керування	Zigbee	Z-wave	Простота використання	Високий рівень захисту	Висока сумісність
SmartThings	Найкраще в цілому	✓	✓	✓	✓	✓	✓	✓
Home Assistant	Найкраще без хаба	-	✓	✓	з ключем	-	✓	✓
Apple HomeKit	Найкраще для користувачів Apple	✓	✓	-	-	✓	-	-
Amazon Alexa	Найкраще для голосового керування	✓	✓	✓	-	-	-	✓
Google Home	Найкраще для екосистеми Google	✓	✓	-	-	✓	-	-
IFTTT	Найкраще для автоматизації	-	✓	-	-	✓	-	✓

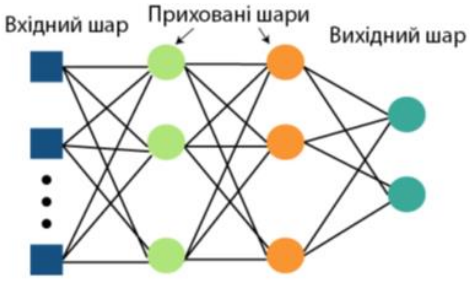
ІСНУЮЧІ МОДЕЛІ



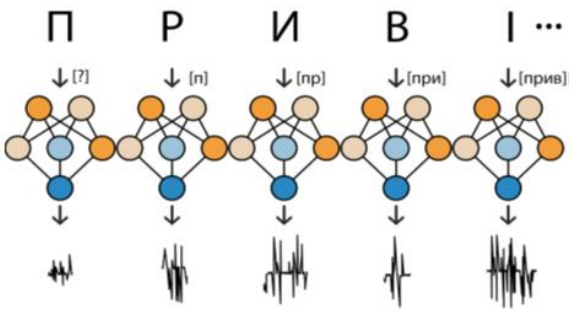
Одношарова нейронна мережа



Згорткова нейронна мережа



N-шарова нейронна мережа

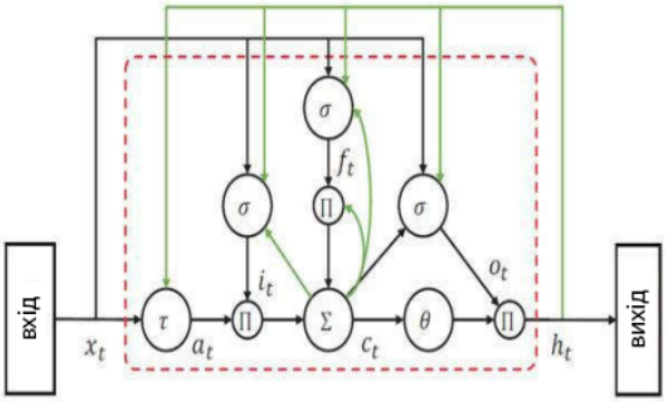


Рекурентна нейронна мережа

ОБРАНА НЕЙРОННА МЕРЕЖА

Було обрано рекурентну нейронну мережу, а саме довгу короткочасну пам’ять (LSTM).

Загальноприйняті LSTM можна визначити наступним чином:
Враховуючи вхідну послідовність $x = (x_1, x_2, \cdots, x_T)$, звичайна рекурентна нейронна мережа обчислює приховану векторну послідовність $h = (h_1, h_2, \cdots, h_T)$ і вихідну векторну послідовність $y = (y_1, y_2, \cdots, y_T)$ від t = 1 до T наступним чином:



Мережа LTSM з одним блоком пам’яті

$$h_t = H(W_{xh}x_t + W_{hh}h_t - 1 + b_h)$$

$$y = W_{hy}h_t + b_y$$

Де, *W* позначає вагові матриці, *b* позначає вектори зміщення, а *H(·)* є рекурентною функцією прихованого шару.

ПРОЦЕС РОЗПІЗНАВАННЯ МОВИ

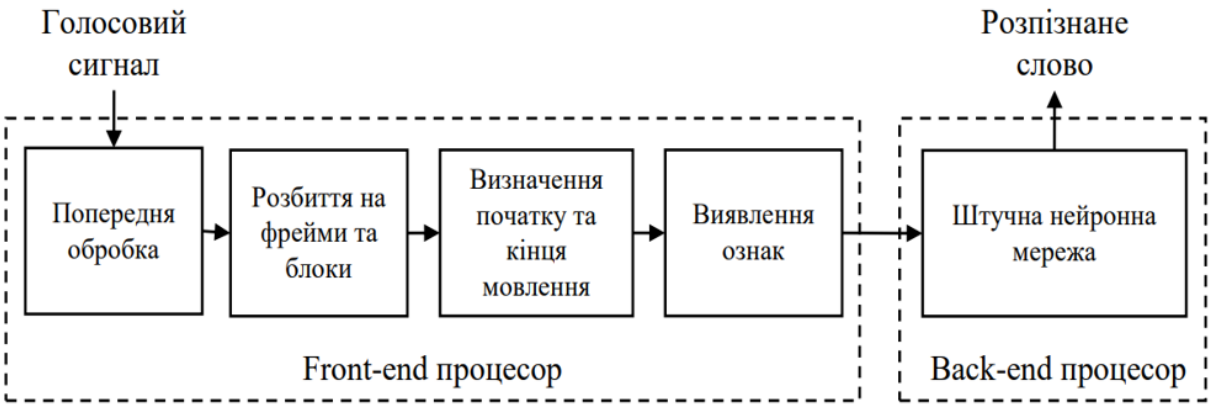


СХЕМА СИСТЕМИ РОЗУМНОГО БУДИНКУ

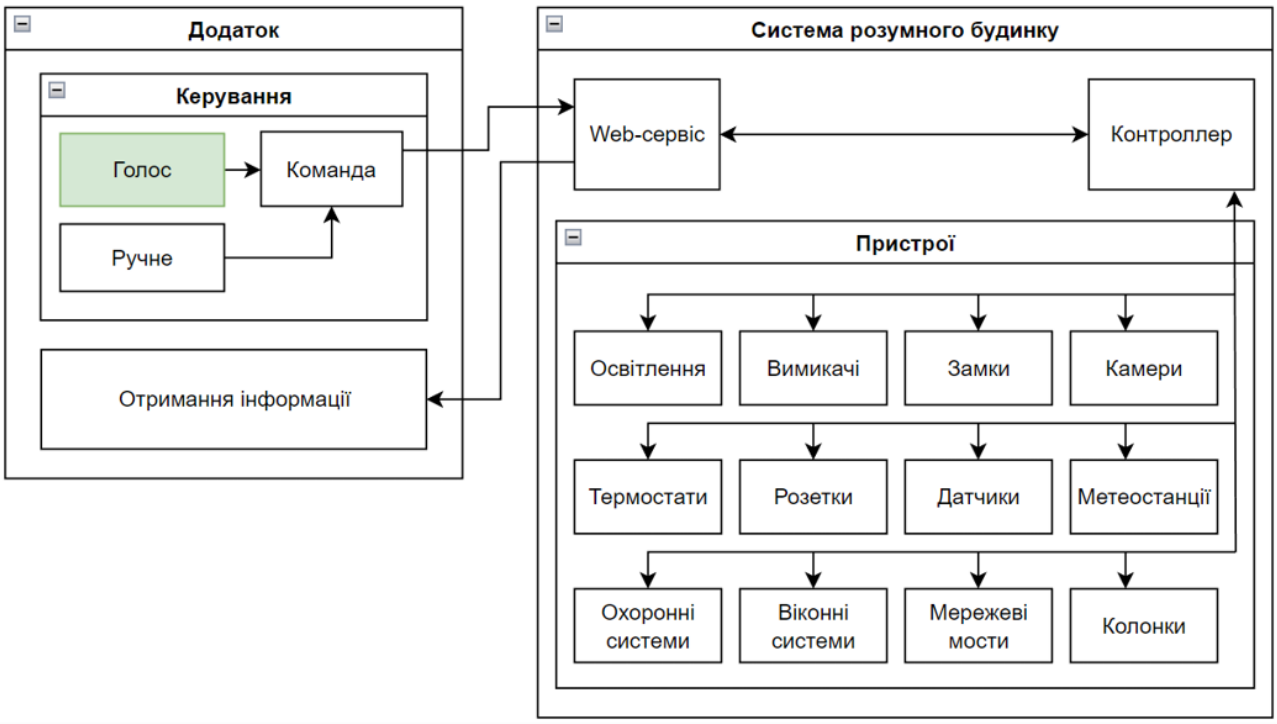
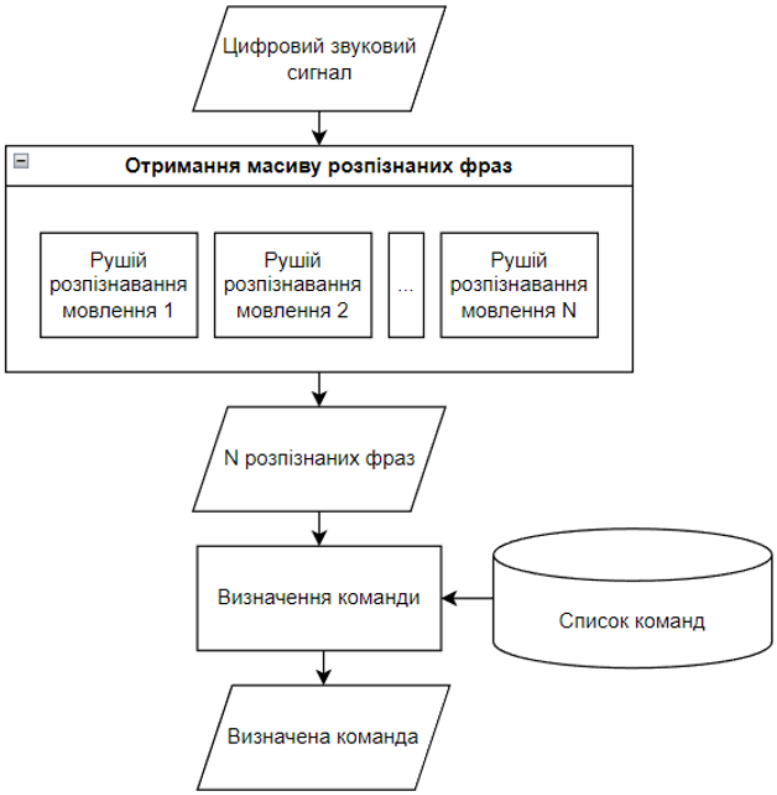


СХЕМА РОЗРОБЛЕНОГО АЛГОРИТМУ



ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБЛЕНОГО АЛГОРИТМУ

МЕТОДИ ВИПРОБУВАННЯ	ТЕСТУВАННЯ ФАКТОРІВ	РЕАКЦІЯ	ТОЧНІСТЬ
АМПЛІТУДА / ГОЛОС	нормальна розмова 60 дБ	відповідає 9 з 10 разів	90%
	шепіт 35 дБ	відповідає 6 раз з 10	60%
КІЛЬКІСТЬ СЛІВ	мінімум 2 слова	точно відповідає 9 із 10 разів	90%
	максимум 5 слів	точно відповідає 8 з 10 разів	100%
ВІДСТАНЬ ВІД МІКРОФОНУ	менша відстань - 12 см	точна відповідь 10 разів з 10	100%
	більша відстань - 30 см	точна відповідь 9 разів з 10	90%
СЕРЕДОВИЩЕ	тихо	точна відповідь 10 разів з 10	100%
	шумно	правильна відповідь 6 з 10 разів	60%
КІЛЬКА ДИНАМІКІВ	декілька спікерів	точно відповідає 4 раз з 10	40%
	індивідуальний спікер	точно відповідає 8 із 10 разів	80%
РОЗМІР КІМНАТИ	меленька кімната	точна відповідь 10 раз з 10	100%
	велика кімната	точна відповідь 10 раз з 10	100%

ВИСНОВКИ

1. Проаналізовано предметну область розпізнавання мовлення, її задачі та застосування у системах розумного будинку.
2. Проаналізовано довгу короткочасну пам’ять та приховані моделі Маркова (і розширення для покращення продуктивності розпізнавання мовлення).
3. Розроблено алгоритм кращого розпізнавання мовлення для системи керування мовлення, в котрий можна інтегрувати різні рушії розпізнавання мовлення. Для інтеграції в алгоритм було обрано Deepgram та Webkit (Google) Speech Recognition.
4. Налаштовано інструментарій для розробки алгоритму та клієнтського додатку.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Ліщук І.В., Дібрівний О.А. Система голосового управління розумним будинком з використанням штучного інтелекту // XV Науково-технічна конференція «Сучасні інфокомунікаційні технології» – Київ: ДУТ, 2022.
2. Ліщук І.В., Дібрівний О.А. Система голосового управління розумним будинком з використанням штучного інтелекту // Науково-практична конференція «Проблеми комп’ютерної інженерії». – Київ: ДУТ, 2022.

ДЯКУЮ ЗА УВАГУ!