КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Інформаційне забезпечення проекту створення додатку голосового управління засобом IoT»

Студента 2-го курсу групи ІАВ-21

Єщенкова Владислава Сергійовича
(прізвище, ім'я, по батькові)

(підпис студента)

Науковий керівник:

Кандидат економічних наук, доцент
(науковий ступінь, вчене звання)

Мезенцева Ольга Олексіївна
(прізвище, ім'я, по батькові)

(дата) (підпис)

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління _____ _____ _____
(підпис) (прізвище, ініціали) (дата)

Київ – 2021 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.

_____
«____» _____20___року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: Єщенков Владислав Сергійович

Група: ІАВ-21

1. Тема кваліфікаційної роботи

«Інформаційне забезпечення проекту створення додатку голосового управління засобом IoT»

Затверджена наказом по від «09» листопада 2020 р. № 4.

2. Строк подання студентом готової роботи – «20» травня 2021 р.

3. Цільова установка та вихідні дані до роботи: розширення інформаційного забезпечення створення додатку голосового управління засобом IoT; модель машинного навчання для пробудження пристрою інтернету речей за допомогою ключового слова.

4. Зміст роботи: ознайомитися з основами створення додатків голосового управління; сформувати вимоги до інформаційного забезпечення створення додатку голосового управління; визначити особливості голосового управління пристроями інтернету речей; скласти характеристику управління засобами IoT; проаналізувати існуючі алгоритми для створення додатків голосового управління; розглянути взаємодію засобів IoT до команд додатків голосового управління; проаналізувати потоки даних у системах IoT з голосовим управлінням; розробити модель для удосконалення інформаційного забезпечення створення додатку голосового управління пристроєм IoT; оцінити ефективність розробленої моделі.

5. Перелік графічного матеріалу (слайдів): 45 рисунків, 2 таблиці, 21 формула, 1 додаток.

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.10.20	01.10.20
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	09.11.20	09.11.20
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	08.12.20	08.12.20
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.21	18.01.21
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	20.01.21	19.01.21
6.	Підготовка розділу 1 «Теоретичні засади проекту створення додатку голосового управління IoT»	15	12.02.21	12.02.21
7.	Підготовка розділу 2 «Оцінка існуючих підходів до створення додатків голосового управління засобом IoT»	18	10.03.21	10.03.21
8.	Підготовка розділу 3 «Аналіз інформаційного забезпечення створення додатку голосового управління засобом IoT»	16	15.04.21	15.04.21
9.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	29.04.21	29.04.21
10.	Передача кваліфікаційної роботи науковому керівникові	2	04.05.21	04.05.21
11.	Передача кваліфікаційної роботи рецензенту для рецензування	2	05.05.21	05.05.21
12.	Попередній захист кваліфікаційної роботи	5	11.05.21	11.05.21
13.	Подача готової кваліфікаційної роботи на кафедру	2	20.05.21	20.05.21

Дата видачі завдання «01» жовтня 2020 р.

Керівник роботи: к.е.н, доц., Мезенцева Ольга Олексіївна _____
(підпис)

Завдання прийняв до виконання студент групи ІАВ-21:

Єщенков Владислав Сергійович _____
(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Єщенкова Владислава Сергійовича.

Тема роботи – «Інформаційне забезпечення проекту створення додатку голосового управління засобом IoT».

Мета дипломної роботи магістра – розширення інформаційного забезпечення створення додатку голосового управління засобом IoT.

Об'єкт дослідження – процес голосового управління пристроєм IoT

Предмет дослідження – є інформаційні засоби та технології управління у сфері інтернету речей.

Наукова новизна роботи полягає в удосконаленні інформаційного забезпечення створення додатку голосового управління засобом IoT, що на відміну від існуючих, дозволяє використовувати механізм пробудження за ключовим словом на малопотужних комп'ютерах та мікроконтролерах в закритих мережах без доступу до Інтернету.

У роботі досліджуються підходи та алгоритми створення додатків розпізнавання мовлення. Розробляється модель для удосконалення інформаційного забезпечення створення додатку голосового управління засобом IoT. Надається оцінка розробленої моделі та рекомендації по вдосконаленню та використанню.

Дипломна робота складається зі вступу, основної частини, яка включає три розділи, висновків та списку використаних джерел. Всього налічує 88 сторінок основного тексту, перелік посилань з 64 джерел на 6 сторінках та додатки на 3 сторінках.

Ключові слова: інтернет речей, голосове управління, глибоке навчання, нейронна мережа.

ВСТУП

Актуальність дослідження. Сфера інтернету речей набула швидкого розвитку за останнє десятиліття. Обсяг ринку інтернету речей у 2017 році складав 110 млрд доларів США. До 2020 року відбулося зростання ринку майже у 2,5 рази, до 250 млрд. При чому сфера невпинно продовжує розширятися. Експерти очікують темпи зростання до 1300 млрд доларів до 2027 року.

Інтернет речей - це пристрої, які люди використовують кожного дня, такі як розумні годинники, голосові помічники, системи розумного будинку і навіть RFID чіпи для відстеження інвентарю в магазинах.

Багато і виробничих галузей використовують IoT, щоб зрозуміти потреби споживачів у режимі реального часу, покращити якість функціонування машин та якості продукції, відкривати інноваційні способи керування виробництвом.

У сфері медицини - це носимі пристрої, що слідкують за показниками стану організму, сном, звичками. Такі пристрої можуть діагностувати відхилення та сповіщати про це лікарів.

У сфері логістики засоби IoT використовуються для відслідковування геолокації, побудови оптимальних логістичних маршрутів, цим самим підвищуючи ефективність та швидкість постачання.

Системи розумного міста включають в себе розумні пристрої для керування світлофорами, вуличними ліхтарями, забезпечують безпеку на вулицях з використанням систем камерного спостереження.

Керування пристроями IoT відбувається за допомогою різних способів, від мобільного застосунку на смартфоні до жестів та голосового управління.

Голосове управління, в свою чергу почало активно розвиватися не дуже давно. Значний приріст у дослідженні цієї технології зумовлений розвитком машинного навчання та початком використання технологій глибокого навчання. Це призвело до значного покращення якості розпізнавання мовлення і дало поштовх до активного розвитку технології великими корпораціями, такими як Google, Apple, Amazon, Yandex. Розроблені алгоритми успішно використовуються у роботі голосових помічників, які зараз можуть вести

розмову в контексті, хоча і обмеженому. Голосове управління також активно використовується у системах розумного дому для керування освітленням, побутовими приладами, безпекою. Сфера автоматичного розпізнавання мовлення є досить перспективним напрямком на сьогоднішній день.

Метою роботи є розширення інформаційного забезпечення створення додатку голосового управління засобом IoT.

Об'єктом дослідження є процес голосового управління пристроєм IoT.

Предметом дослідження є інформаційні засоби та технології управління у сфері інтернету речей.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

1. Ознайомитися з основами створення додатків голосового управління.
2. Сформувати вимоги до інформаційного забезпечення створення додатку голосового управління.
3. Визначити особливості голосового управління пристроями інтернету речей.
4. Скласти характеристику управління засобами IoT.
5. Проаналізувати існуючі алгоритми для створення додатків голосового управління.
6. Розглянути взаємодію засобів IoT до команд додатків голосового управління.
7. Проаналізувати потоки даних у системах IoT з голосовим управлінням.
8. Розробити модель для удосконалення інформаційного забезпечення створення додатку голосового управління пристроєм IoT.
9. Оцінити ефективність розробленої моделі.

Методами дослідження є опис та узагальнення щодо існуючих підходів створення додатків голосового управління, методи аналізу та синтезу при дослідженні алгоритмів розпізнавання мовлення, методи вимірювання для оцінювання якості розробленої моделі.

Наукова новизна полягає в удосконаленні інформаційного забезпечення створення додатку голосового управління засобом IoT, що на відміну від існуючих, дозволяє використовувати механізм пробудження за ключовим словом на малопотужних комп'ютерах та мікроконтролерах в закритих мережах без доступу до Інтернету.

Практична значимість полягає у дослідженні та розширенні інформаційного забезпечення створення додатку голосового управління засобом IoT, розробці моделі машинного навчання для пробудження пристрою інтернету речей за допомогою ключового слова.

Публікації. Основні наукові положення, висновки і результати магістерської кваліфікаційної роботи знайшли відображення у 1 тезах доповіді на конференції [1].

Структура та обсяг роботи. Дипломна робота складається зі вступу, основної частини, яка включає три розділи, висновків, переліку використаних джерел з 64 пунктами на 6 сторінках та додатків на 3 сторінках. Загальний обсяг кваліфікаційної роботи становить 103 сторінки, із них 88 сторінок основного тексту.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ ПРОЕКТУ СТВОРЕННЯ ДОДАТКУ ГОЛОСОВОГО УПРАВЛІННЯ ІОТ

1.1 Основи проекту створення додатку голосового управління

Мова є основним засобом спілкування між людьми. Дослідження в області автоматичного розпізнавання та синтезу мови за допомогою машини та бажання автоматизувати прості завдання, що вимагають взаємодії людини і машини останні шість десятиліть привертають увагу вчених.

Сьогодні мовні технології комерційно доступні для обмеженого кола завдань. Ці технології дозволяють машинам правильно і надійно реагувати на людський голос, а також надавати корисні та цінні послуги. Поки ми ще далекі від машин, які розмовляють з людьми на будь-які темі, як з іншою людиною, але ми вже досягнули деяких успіхів, які наближають нас до створення таких пристроїв. Розглянемо ключові історичні моменти, що вплинули на цей прогрес.

Ранні спроби створення систем автоматичного розпізнавання мови в основному ґрунтувалися на теорії акустико-фонетики, що описує фонетичні елементи (основні звуки) мови і намагається пояснити, як вони акустично реалізуються в усному висловлюванні. Ці елементи включають фонеми і відповідну манеру артикуляції, що використовуються для відтворення звуку в різних фонетичних контекстах. Наприклад, щоб отримати стійкий голосний звук, голосові зв'язки повинні вібрувати, і повітря, яке поширюється через голосовий тракт, дає звук з природними режимами резонансу, подібними тому, що відбуваються в акустичній трубці. Ці природні форми резонансу, форманти або формантні частоти, проявляються як основні області концентрації енергії в спектрі потужності мовлення. У 1952 р. Девіс, Біддульф і Балашек з Bell Laboratories створили систему для розпізнавання ізольованих цифр для одного диктора [2], використовуючи формантні частоти, виміряні в регіонах голосних звуків при вимові кожної цифри. В інших ранніх системах розпізнавання 1950-х років Олсон і Белар з RCA Laboratories створили систему розпізнавання 10

складів одного диктора і в MIT Lincoln Lab, Джеймс Форгі побудував систему розпізнавання 10 голосних, що не залежить від мовця [3]. У 1960-х роках кілька японських лабораторій продемонстрували свою здатність створювати устаткування спеціального призначення для виконання задач розпізнавання мовлення. Найбільш значимими були прилади для розпізнавання голосних, створені в радіодосліджувальній лабораторії в Токіо , система розпізнавання фонем Сакаї і Дошіта в Університеті Кіото і розпізнавач цифр від NEC Laboratories [4]. У роботі Сакаї і Дошіта вперше застосовано мовний сегментатор для аналізу і розпізнавання мовлення в різних частинах вхідного висловлювання. А ізольований прилад для розпізнавання цифр працював за принципом припущення, що невідоме висловлювання містить повну цифру (і ніяких інших звуків мови або слова) і, отже, не потребували явного сегментування. Робота Кіотського університету може вважатися попередником системи розпізнавання злитого мовлення.

Одна із складностей розпізнавання мови, що існувала, полягала в нерівномірності розподілу мовленевої події на часовій шкалі. У 1960-х Мартін і його колеги з RCA Laboratories розробили набір елементарних методів нормалізації часу, заснованих на здатності надійно визначати початок мовлення і закінчення, що значно знизило мінливість оцінок розпізнавання.

Наприкінці 1960-х та на початку 1970-х років дослідники розпізнавання мовлення досягли низки важливих етапів. У 1968 році в Радянському Союзі Тарас Вінцюк запропонував використовувати методи динамічного програмування для вирівнювання часу пари висловлювань (загальновідомої як Dynamic Time Warping (DTW)), включаючи алгоритми для розпізнавання пов'язаних слів. Однак його робота була в основному невідомою в інших країнах до 1980-х років. У той же час, незалежними зусиллями в Японії, Сакое та Чіба в лабораторіях NEC також почали використовувати метод динамічного програмування для вирішення проблеми нерівномірності людської вимови. Сакое та Чіба вдосконалили свою техніку в 1970-х. З кінця 1970-х років динамічне програмування у численних варіантних формах, включаючи алгоритм

Вітербі [6], що походить від загальної теорії комунікацій, стало незамінною технікою автоматичного розпізнавання мовлення.

Область розпізнавання ізольованих слів або висловлювань стала придатною для використання технологією, заснованою на фундаментальних дослідженнях в Радянському Союзі та Японії. Величко та Загоруйко у Радянському Союзі вдосконалили використання ідей розпізнавання шаблонів у розпізнаванні мовлення. Працюючи в Bell Laboratories, Ітакура показав, як лінійне прогностичне кодування (LPC) може бути застосовано до систем розпізнавання мовлення за допомогою відповідної міри відстані на основі спектральних параметрів LPC.

Наприкінці 1960-х років Редді з Університету Карнегі-Меллона провів новаторські дослідження в галузі безперервного розпізнавання мовлення за допомогою динамічного відстеження фонем.

IBM Labs: Дослідники розпочали вивчення розпізнавання мовлення з великим словником для трьох різних задач, а саме: використання мови New Raleigh для простих запитів до баз даних, транскрипція для опису лазерних патентів та для задачі з кореспонденційного листування під назвою Tangora для надиктовування простих пам’яток.

AT&T Bell Labs: Дослідники розпочали серію експериментів, спрямованих на створення незалежних від мовця систем розпізнавання мовлення. Для досягнення цієї мети було використано широкий спектр складних алгоритмів кластеризації для визначення кількості різних шаблонів, необхідних для представлення всіх варіацій різних слів у широкій сукупності користувачів.

DARPA program: Амбіційний проект з розуміння мови був профінансований Агентством оборонних дослідницьких проектів оборони (DARPA) у США, що призвело до розроблення багатьох основних систем та технологій. Однією з перших демонстрацій розпізнавання мовлення було досягнуто Університетом Карнегі-Меллона в 1973 р. Їхня система Harpy змогла використовувати семантичну інформацію для значного зменшення кількості альтернатив, що розглядалися системою. Показано, розпізнає мовлення за

допомогою словника з 1011 слів із достатньою точністю. Одним із особливих внесків системи Harpy стала концепція пошуку по графу, словник представлений як пов'язана мережа, в якій відображалися правила лексичних подань слів, синтаксичні правила словотворення та розмежування. Система Harpy була першою, яка використала мережу кінцевих станів (FSN), щоб зменшити обчислення та ефективно визначити найближчий рядок відповідності.

Проблема створення надійної системи, здатної розпізнавати низку сполучених слів або цифр при вільній розмові була основною темою досліджень у 1980-х роках. Було сформульовано та впроваджено широкий спектр алгоритмів, заснованих на відповідності об’єднаних шаблонів окремих слів, включаючи дворівневий підхід динамічного програмування Сакое з NEC, однопрохідний метод Брідла та Брауна у Спільному дослідницькому підрозділі. (JSRU) у Великобританії, підхід Level-building Майерса та Рабінера та підхід Framesynchronous Level-building Лі та Рабінера у Bell Labs [7]. Кожна з цих процедур мала свої переваги щодо впровадження.

Дослідження розпізнавання мовлення у 1980-х роках характеризувалося зміщенням методології від більш інтуїтивного підходу, заснованого на шаблоні (прямолінійна парадигма розпізнавання зразків), до більш суворої системи статистичного моделювання. Сьогодні більшість практичних систем розпізнавання мовлення базуються на статистичній системі, розробленій у 1980-х роках, та їх результатах, із значними додатковими вдосконаленнями, які були зроблені в 1990-х роках.

Однією з ключових технологій, розроблених у 1980-х роках, є підхід прихованої моделі Маркова (HMM) [8]. Це подвійний стохастичний процес, оскільки він має в основі стохастичний процес, який не можна спостерігати (звідси термін «прихований»), але, що досягається за допомогою іншого стохастичного процесу, який виробляє послідовність спостережень. Хоча HMM був добре відомий і зрозумілий у декількох лабораторіях (насамперед IBM, Інституті оборонного аналізу (IDA) та Dragon Systems), лише в середині 80-х років розповсюджена публікація методів та теорії HMM широко застосовується

практично у кожній дослідницькій лабораторії розпізнавання мовлення по всьому світу.

Фуруї запропонував використовувати комбінацію миттєвих кепстральних коефіцієнтів та їх поліноміальних коефіцієнтів першого та другого порядку, як основні спектральні особливості для розпізнавання мови. Він запропонував цей метод наприкінці 1970-х, але ніхто навіть не намагався застосувати його для розпізнавання мовлення протягом багатьох років. Але зарай цей метод широко використовується майже у всіх системах розпізнавання мови.

Основним напрямком діяльності IBM була розробка структури мовної моделі (граматики), яка була представлена статистичними синтаксичними правилами, що описують, наскільки ймовірно, в імовірнісному сенсі, послідовність мовних символів (наприклад, фонем або слів) могли б з'являтися в мовному сигналі. Була введена N-грам модель, яка визначала ймовірність впорядкованої послідовності N слів, і з тих пір використання мовних моделей N-грам та її варіантів стало незамінним у розпізнаванні мовлення у системах з великим словником.

У 1980-х роках була знову введена ідея застосування нейронних мереж для розпізнавання мови. Нейронні мережі вперше були представлені в 1950-х роках, але вони не виявились корисними через практичні проблеми. У 1980-х роках було досягнуто більш глибоке розуміння переваг та обмежень технології, а також розуміння зв'язку цієї технології із класичними класифікаційними методами [9].

Спільнота DARPA провела дослідження з використанням великого словника в системі безперервного розпізнавання мовлення, спрямованої на досягнення високої точності розпізнавання слів для завдання управління базами даних із 1000 слів. Основні внески в дослідження були результатом зусиль в CMU із системою SPHINX, BBN із системою BYBLOS , SRI із системою DECIPHER, у Lincoln Labs, MIT та AT&T Bell Labs [10]. Система SPHINX успішно інтегрувала статистичний метод HMM із мережевим пошуком попередньої системи Harpy.

У 1990-х роках у галузі розпізнавання мовлення відбувся ряд нововведень. Проблема розпізнавання шаблонів, яка традиційно слідувала рамкам Байєса і вимагала оцінки розподілу даних, перетворилася на задачу оптимізації, що передбачає мінімізацію емпіричної помилки розпізнавання. Ця фундаментальна парадигматична зміна була викликана визнанням того факту, що функції розподілу мовного сигналу не можуть бути точно вибрані чи визначені, і що теорія рішення Байєса стає непридатною за цих обставин. По суті, метою проекту створення системи-розпізнавача повинно бути досягнення найменшої помилки розпізнавання, а не забезпечення найкращого способу використання функції розподілу до існуючого набору даних, як відстоюється критерієм Байєса. Ця концепція мінімізації помилок створила ряд методів, таких як дискримінаційне навчання та методи, засновані на ядрі.

Як приклад дискримінаційного навчання було запропоновано критерій мінімальної похибки класифікації (MCE) разом із відповідним алгоритмом навчання узагальненого ймовірнісного спуску (GPD), щоб мінімізувати цільову функцію, яка діє для наближення рівня похибки [11]. Іншим прикладом був критерій максимальної взаємної інформації (MMI). Під час навчання MMI взаємна інформація між акустичним спостереженням та його правильним лексичним значенням, усередненим по навчальному набору, максимізується. Незважаючи на те, що цей критерій не заснований на прямій мінімізації рівня помилок класифікації і цілком відрізняється від підходу, заснованого на MCE, він добре обґрунтовується в теорії інформації та має добрі теоретичні властивості. І MMI, і MCE можуть призвести до ефективності розпізнавання мови, що перевершує підхід, заснований на максимальній імовірності [11].

Програма DARPA продовжувалась і в 1990-х роках, при цьому акценти переходили до завдання розпізнавання природного мовлення. Центральна увага також була перенесена на завдання отримання інформації про авіаперевезення від служби авіаційних подорожей (ATIS). Пізніше акцент було зміщено на цілий ряд різних областей, що розуміють мовлення, у поєднанні з транскрипцією мовних новин (БН) та розмовну мову. Задача комутатора є однією із

найскладніших, запропонованих DARPA; у цьому завданні мова є розмовною і спонтанною, з багатьма випадками так званих розбіжностей, таких як частково вимовлені слова, вагання. Технологія транскрипції BN була інтегрована з технологією вилучення та пошуку інформації, і було розроблено багато прикладних систем, таких як автоматичне індексування та пошук голосових документів. Ряд проектів технологій дослідження людської мови, що фінансувалися DARPA у 1980-х та 1990-х роках, ще більше прискорили прогрес.

З початку 2000-х років були досліджені різні методи для підвищення стійкості систем розпізнавання мови проти невідповідності між умовами навчання та тестування, спричинених фоновими шумами, індивідуальністю голосу, мікрофонами, каналами передачі, реверберацією кімнати тощо. Основні методи включають максимальну ймовірнісну лінійну регресію (MLLR), розкладання моделі, паралельний складання моделі (PMC) та апостеріорний метод структурного максимуму (SMAP)[12].

Технологія розпізнавання мовлення все частіше застосовується в телефонних мережах для автоматизації та покращення послуг оператора.

Було започатковано програму EARS для розробки технології автоматичної транскрипції з метою досягнення значно багатших та набагато точніших результатів, ніж раніше. Завдання включають виявлення меж речень, розділових знаків. Програма була зосереджена на природному, біглому мовленні людини в ефірі та спільній розмові декількох іноземців на різних мовах. Метою було зробити для машин можливість набагато кращої роботи з виявлення, вилучення, узагальнення та перекладу важливої інформації, таким чином даючи можливість людям зрозуміти сказане, читаючи транскрипції замість прослуховування звукових сигналів.

Хоча читання тексту і подібні типи мовлення можуть бути розпізнані з точністю вище 95% з використанням сучасної технології розпізнавання мовлення, точність розпізнавання для спонтанного мовлення різко знижується. Розширення застосування розпізнавання мови вирішальним чином залежить від підвищення ефективності розпізнавання спонтанної мови. З метою підвищення

ефективності розпізнавання спонтанної мови було проведено кілька проектів. В Японії було проведено 5-річний національний проект «Спонтанна мова: корпус та технології обробки» . Був побудований найбільший у світі "Корпус спонтанної японської мови" (CSJ), що складається приблизно з 7 мільйонів слів, що відповідає 700 годинам мовлення для дослідження різних нових методів. Ці нові техніки включають гнучке акустичне моделювання, виявлення меж речень, моделювання вимови, а також адаптацію мовної моделі та автоматичне узагальнення мови [13].

Для подальшого підвищення надійності систем розпізнавання мови, особливо для спонтанного мовлення, інтенсивно досліджуються заходи перевірки висловлювання та міри довіри. Щоб мати інтелектуальну або подібну до людини взаємодію в діалогових додатках, важливо приєднати до кожної розпізнаної події число, яке вказує, наскільки впевнено система ASR може приймати розпізнані події. Міра довіри служить довідкою для діалогової системи, щоб надати належну відповідь своїм користувачам. Для виявлення семантично значущих частин та відкидання нерелевантних частин у спонтанних висловлюваннях нещодавно було досліджено новий підхід виявлення [14]. Ця комбінована стратегія розпізнавання та перевірки добре працює, особливо для неправильно сформованих висловлювань. Для побудови акустичних моделей, більш складних, ніж звичайні HMM, досліджено динамічну байєсівську мережу.

Люди використовують так зване «мультимодальне» спілкування, коли розмовляють між собою. Дослідження розпізнавання мовлення показали, що наявність як візуальної, так і звукової інформації збільшує швидкість успішної обробки інформації, особливо коли повідомлення складне або коли спілкування відбувається в гучному середовищі. Досліджено використання візуальної інформації про обличчя, зокрема артикуляції, для розпізнавання мовлення, і результати показують, що використання обох типів інформації дає кращі результати розпізнавання, ніж використання лише аудіо чи лише візуальних образів, особливо в гучних умовах.

1.2 Вимоги до інформаційного забезпечення додатку голосового управління

У зв'язку з удосконаленням мовленевих і мовних технологій протягом останніх декількох десятиліть потреба в оцінці та оцінці таких технологій значно зросла. Починаючи з оцінки окремих компонентів системи, таких як автоматичне розпізнавання мови (ASR) або синтез тексту в мову (TTS), тепер потрібні методи оцінки для звернення до системи і заснованої на ній послузі в цілому. Тут варто розглянути як оцінку окремих компонентів, так і оцінку всієї системи, в залежності від завдання, для якого призначена та чи інша технологія.

Далі термін оцінка буде використовуватися для вимірювання продуктивності системи (компонента) по одному або декількох критеріях, встановлених оцінювачем та для визначення придатності системи для певної задачі, поставленої потенційним користувачем системи. Ця класифікація подібна до "оцінки ефективності" та "оцінки адекватності", запропонованої Гіршманом та Томпсоном [15]. Обидва типи оцінки можуть мати аналітичний або утилітарний характер, тобто вони можуть забезпечувати діагностичний профіль якості системи або продуктивності (іноді її називають "діагностичною оцінкою") або загальний показник, пов'язаний із загальною якістю, зручністю використання або прийнятністю.

Оцінювання є процесом вимірювання у більш широкому сенсі: метою є отримання кількісних описів продуктивності системи або якості, що сприймається користувачем. Такі процеси вимірювання відрізняються від, наприклад, фізичних, тим, що вимірювальний прилад повинен доповнюватися контролюючим органом вимірювання, а саме людиною - учасником випробування. У цьому сенсі вони є «суб’єктивними» вимірами, на відміну від інструментальних вимірювань, які покладаються лише на фізичний вимірювальний прилад. Поняття «суб'єктивність», не означає, що такі методи будуть менш надійними. Навпаки, достовірні та надійні вимірювання якості можна отримати лише за допомогою учасників тестів. Як суб'єктивні, так і

інструментальні вимірювання повинні відповідати загальним вимогам до вимірювань, таким як:

• валідність (метод повинен мати можливість вимірювати те, що призначено для вимірювання),

• надійність (метод повинен забезпечувати стабільні результати при повторному виконанні операції для того самого вимірювання),

• об'єктивність,

• чутливість (метод повинен мати можливість реагувати на невеликі варіації того, для чого призначене вимірювання),

• стійкість (метод повинен мати можливість забезпечувати результати, незалежні від змінних, які є сторонніми для вимірюваної конструкції).

Завдання оцінювання - визначити, наскільки послуга та система, на яку вона спирається, відповідають вимогам користувача або розробника системи. Для цього зазвичай використовують два терміни: продуктивність та якість.

Що стосується окремих компонентів системи, то доречніше говорити про продуктивність, тобто про здатність модуля забезпечувати виконання поставлених функцій, для яких він розроблений. Ефективність може бути легко визначена кількісно, коли існує міра, яка тісно пов’язана з реалізованим функціоналом. Наприклад, функціонал розпізнавання мовлення полягає в транскрипції вимовленого висловлювання у послідовність слів, а кількісно вимірюваним показником, пов’язаним із цим, є, рівень помилки цієї транскрипції. Поняття ефективності є менш очевидним, коли функціонал не до кінця зрозумілий, або коли з ним не пов'язаний «натуральний» показник. Розглянемо, наприклад, синтезатор TTS: його функція полягає в забезпеченні генерування усного висловлювання з послідовності написаних слів; однак успішність виконання завдання може бути пов'язаним з різними мірками, такими як сегментарна зрозумілість, зрозумілість в цілому, зусилля при прослуховуванні, природність вимови, виразність та адекватність.

Для інтерактивної системи в цілому також не просто визначити одну точну функцію. Хоча ефективність інформаційної системи для зачитування розкладу

руху поїздів може в першу чергу визначатися з точки зору якості виконання завдання (тобто, чи забезпечує вона кінцевого користувача бажаною інформацією), існують інші важливі функції та відповідні заходи, такі як її продуктивність (виміряна, наприклад, через час, необхідний для виконання завдання, або завдяки зусиллям, що вимагаються від користувача), задоволення користувача (пов'язане з відчутим комфортом, приємністю чи радістю від використання системи), його корисність (з урахуванням витрат), а також прийнятність (тобто, чи готовий потенційний користувач використовувати систему). Заходи для більшості цих аспектів якості можуть бути визначені лише на основі суб'єктивних експериментів з учасниками випробувань. Якість мовної інтерактивної системи визначається сприйняттям її користувачів. Це є результатом сприйняття та процесу судження, коли суб'єкт, що користується системою, встановлює взаємозв'язок між сприймаючою подією та тим, що він / вона очікує чи бажає від цієї системи. Цей результат також можна назвати «якісною подією», оскільки він визначається у просторі, часі та характері (як і всі події). Така орієнтована на користувача точка зору відображається у визначенні якості, даному Єкошем [16]: «Результат оцінки сприйманого складу сутності по відношенню до його бажаного складу».

Оскільки саме користувач оцінює якість, такі фактори, як ставлення, емоції, досвід та знання про поставлену задачу, впливатимуть на сприйняття якості.

Важливим аспектом якості мовної інтерактивної системи є її зручність використання. Зручність визначається як «ступінь, за якої продукт може бути використаний зазначеними користувачами для досягнення визначених цілей з ефективно, продуктивно та із задоволенням у певному контексті використання»[17]. Її можна виміряти за трьома показниками ефективності (тобто точності та повноти, за допомогою якої зазначені користувачі можуть досягти визначених цілей у визначених середовищах), продуктивності (тобто витрачених ресурсів щодо точності та повноти досягнутих цілей) та задоволення

(комфорт та прийнятність системи для її користувачів та інших людей, які її використовують).

Для розмежування безлічі аспектів якості встановлено таксономію [18, 19] (Рис. 1.1).

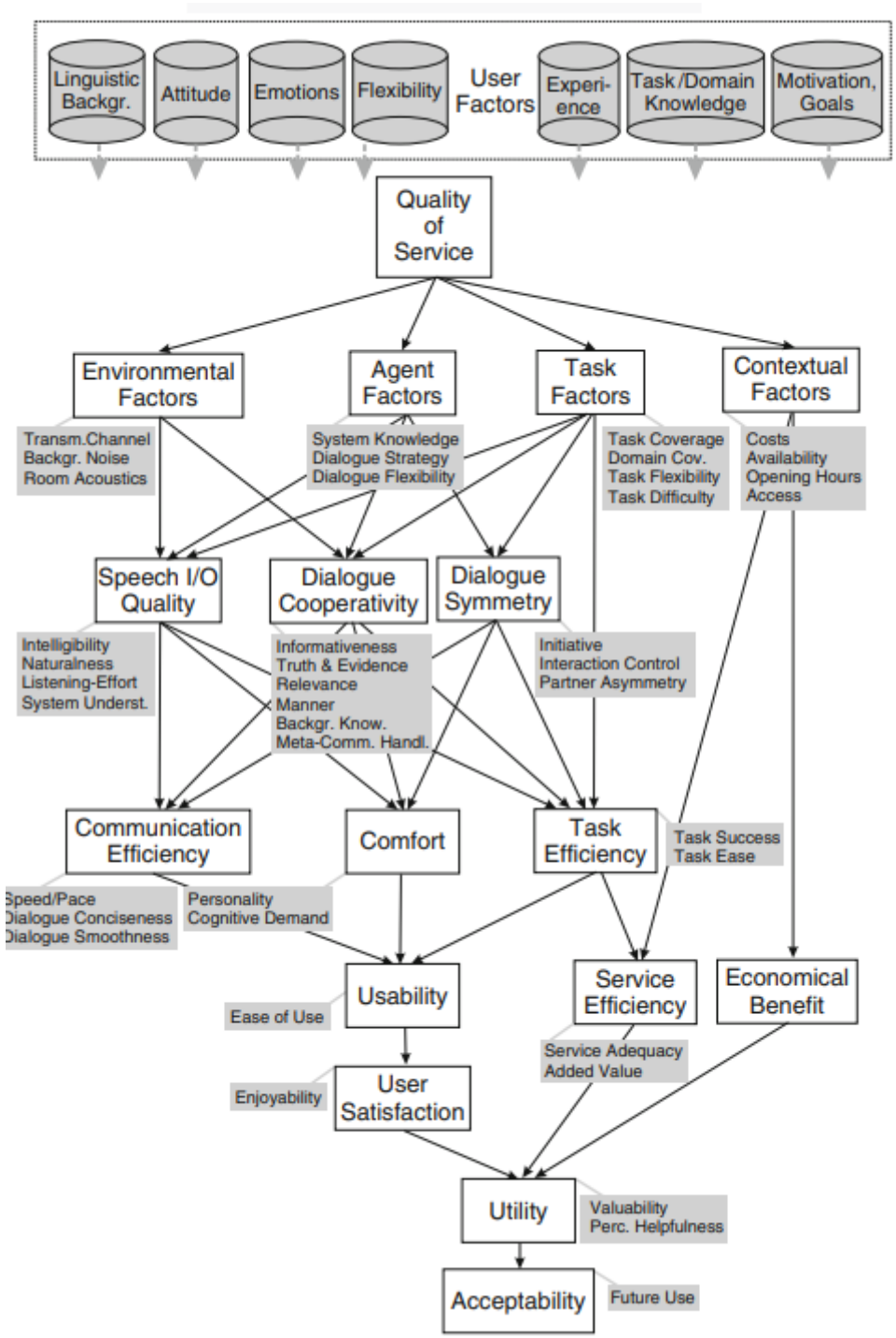


Рисунок 1.1 – Таксономія аспектів якості системи розпізнавання мовлення [18, 19]

У верхній частині малюнка показано фактори системи та контекст використання, які впливають на якість (фактори якості). У нижній частині він відображає аспекти якості, тобто категорії якості, та їхні перцептивні виміри, особливості якості з точки зору користувача. Таксономія тут далі обговорюватися не буде, а зацікавлений читач посилається на розширений опис у [18]. Тим не менше, картинка виявляє багатовимірність якості як з точки зору розробника системи, так і з точки зору користувача. Ці виміри слід враховувати в процесі оцінки та оцінки. Універсальної архітектури мовних інтерактивних систем не існує.

Більшість систем містять компоненти для розпізнавання мовлення, розуміння природної мови, управління діалогом та виведенням генерованого мовлення. Додаткові (необов'язкові) компоненти включають телефонні інтерфейси, ідентифікацію користувачів, доступ до бази даних, генерацію відповідей та синтез мовлення.

Для порівняння гіпотези рядка слів розпізнавача безперервного мовлення та цільової транскрипції, гіпотезу та транскрипцію спочатку потрібно вирівняти, наприклад, використовуючи алгоритм динамічного програмування, який призначає різні ваги видаленим, заміненим або вставленим словам[20]. На підставі вирівнювання підраховується кількість правильно ідентифікованих, замінених, видалених та вставлених слів і визначається частота помилок або точність слів. Для ізольованого розпізнавача слів кількість вставок замінюється швидкістю помилкової тривоги.

Подібні показники також можуть бути визначені на рівні речення, що впливає на рівень помилок у реченнях або точність розпізнавання речень. Альтернативні показники на рівні речення включають середню кількість помилкових речень та помилкових слів у реченні. Зазвичай точність речення нижча за точність слова, оскільки одне неправильно розпізнане слово може вплинути на ціле речення. Однак багато помилок розпізнавання насправді не впливають на потік діалогу; їх можна виключити, застосувавши надійні методи

часткового розбору, або повністю ігнорувати, якщо жодне «ключове слово», необхідне для семантичної інтерпретації системи, не позначається.

Кращим підходом для мовленнєвих інтерактивних систем є безпосереднє порівняння семантичних понять, які можна отримати з висловлювання користувача. Найчастіше семантичні поняття можна формалізувати з точки зору пар атрибут-значення (AVP). Подібно до підходу, застосованого для оцінки розпізнавання мовлення, підраховуються семантичні концепції, які були замінені, видалені або вставлені, а потім обчислюється точність або рівень помилок концепції. В якості альтернативи кожне висловлювання користувача може бути позначене, чи було воно правильно, частково правильно чи неправильно проаналізовано, а потім обчислюється точність розуміння (щодо загальної кількості висловлювань користувачів).

1.3 Особливості голосового управління IoT

Поняття Інтернету речей (IoT) з’явилося на ще початку 2000-х років. Інтернет речей розглядається як нова революція мережевих технологій. Кевін Ештон, візіонер технологій IoT, вказав, що IoT може підключати всі пристрої до мережі, від щоденно використовуваних девайсів до промислового обладнання. А можливість збирати всю інформацію та керувати всіма пристроями в одній мережі може значно покращити ефективність та зручність.

Рання концепція Інтернету речей полягає в управлінні та інвентаризації об’єктів за допомогою радіочастотної ідентифікації (RFID). Одним із прикладів є «Електронний код товару», який представляє ідею схеми ідентифікації об’єктів із використанням Інтернету та технології RFID [21].

Концепція Інтернету речей також базується на іншій ідеї додатків Machine-to-Machine (M2M) - "машина має більшу цінність, коли вона з'єднана в мережу, і що мережа стає більш цінною, оскільки більше машин підключено" [22] . На самому початку Інтернет був мережею для комп'ютерів, які з'єднувались для обміну даними. З удосконаленням та зближенням таких технологій, як бездротовий зв’язок, розвинення різного роду датчиків та вбудованих систем, типи девайсів, які можуть підключатися до Інтернету, значно розширилися.

У наші часи усю сукупність програм, машин та всього, що пов’язано з обміном інформацією, автоматизацією називають Інтернетом речей. І для підключення всіх цих пристроїв до Інтернету виникає потреба створення мільйонів і навіть мільярдів адрес. Протокол IPv6, який призначений замінити стару версію IPv4, було створено для надання приблизно $3{,}4 \times 10^{38}$ адрес.

Після підключення до мережі повинна з’являтися можливість у пристроїв обмінюватися інформацією між собою. Існує кілька бездротових мережевих технологій, таких як ZigBee, NFC та Wi-Fi. Розробники можуть вибирати різні технології для конкретного середовища відповідно до цілей та переваг і недоліків самих технологій зв’язку.

ZigBee - це бездротова технологія, яка може передавати дані через сітчасті мережі. Його пропускна здатність і швидкість передачі набагато нижчі, ніж Wi-Fi, але він ідеально підходить для додатків, які вимагають низької потужності та невисокої вартості. NFC досить сильно відрізняється від ZigBee та Wi-Fi. Оскільки його ефективний діапазон дуже малий, його можна використовувати лише в деяких особливих випадках, таких як безконтактна оплата та доступ до картки. Типовою технологією зв’язку є Wi-Fi для мережі IoT, оскільки вона має значну зону покриття та високу швидкість передачі даних, а єдиним недоліком є високе споживання електроенергії. Щоб вирішити проблему енергоспоживання Wi-Fi, Qualcomm представила свою нову технологію платформи Wi-Fi для IoT, яка врегульовує цей недолік шляхом оптимального управління електроспоживання [23].

Окрім технологій підключення, датчики та процесори є іншими важливими елементами IoT. Нові технології датчиків і мікропроцесорів дозволяють створювати нові види IoT направлених на вирішення різних робочих процесів.

Сенсори стають меншими, що дозволяє зменшувати розміри самих пристроїв. Крім цього підвищується стійкість самих сенсорів до умов навколишнього середовища, таких як тиск, висока температура, водонепроникність.

Для процесорів, з одного боку, виробники додають більше ядер або процесорів в один чіп для підвищення продуктивності; з іншого боку, інновації в дизайні та мікроархітектурі процесорів продовжують здійснюватися виробниками для забезпечення більш високої ефективності. Видатним прикладом є чіп SyNAPSE від IBM, який є електронною нейроморфною машинною технологією, що забезпечує дуже потужні обчислювальні можливості [24]. Покращення продуктивності процесора надає можливість виконувати складні цілі для систем IoT. Окрім продуктивності, спеціальні процесори спеціального призначення також допомагають розширити сферу IoT. Мережеві процесори Bluetooth Smart від STM - це спеціальні процесори, які забезпечують рішення Bluetooth із наднизьким енергоспоживанням [25].

У майбутньому п’ять ключових сфер стануть найбільш привабливими та перспективними для розвитку IoT (Рис. 1.2):

1. Носимі пристрої, які в основному збирають інформацію від тіла та забезпечують і описують певні процеси життєдіяльності.
2. Пристрої для автомобілів. Вбудовані системи забезпечують зручність керування автомобілем. Наприклад, навігація, сповіщення про трафік та аварії, зв’язок з іншими автомобілями.
3. Системи розумного дому, що призначені для домашньої автоматизації. Вони включають управління безпекою, освітленням, різними видами побутової техніки.
4. Smart cities, або розумні міста. Включають в себе міське освітлення, сполучення будинків, управління енергетикою, міським трафіком тощо.
5. Промислові IoT девайси. Поєднує в собі велику кількість пристроїв для автоматизації виробництва, вирощування польових культур, збору та аналізу даних про навколишнє середовище.

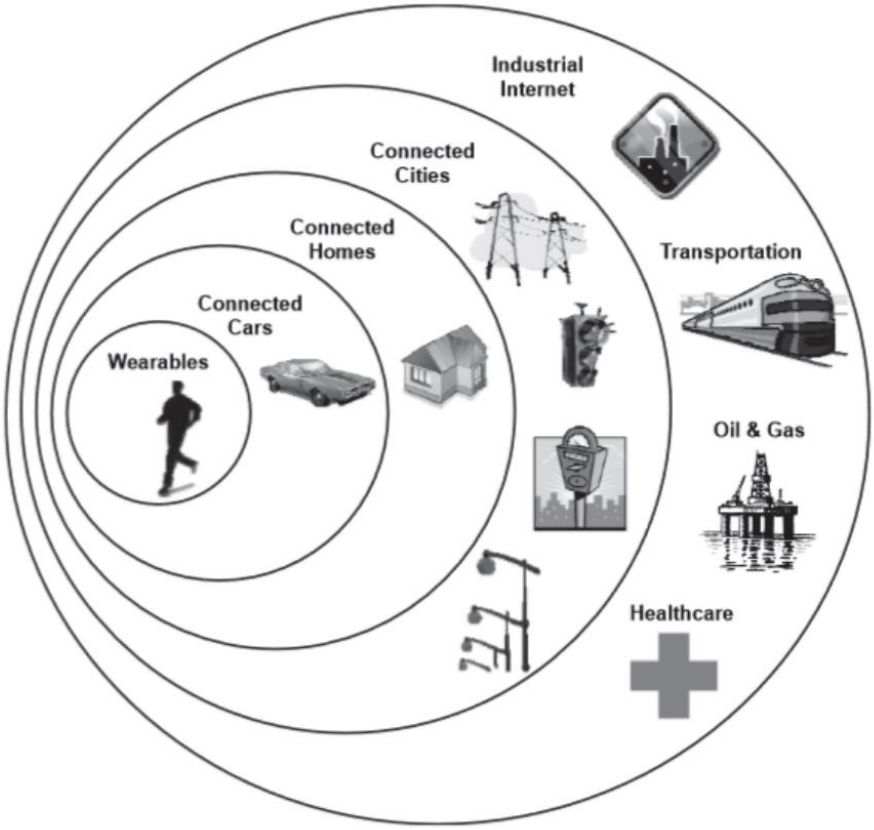


Рисунок 1.2 – Основні сфери IoT [25]

Інтернет речей продовжує розвиватися з величезною кількістю суміжних технологій, таких як хмарні обчислення, Big Data, робототехніка та комунікаційні технології [26]. Розпізнавання голосу та управління жестами - це дві популярні комунікаційні технології в Інтернеті речей.

Розпізнавання голосу широко використовується у пристроях IoT, і існує безліч програм для розпізнавання голосу. Оскільки обмеження розміру та сфера використання може накладати обмеження, де типовий метод введення не може бути виконаний, там голосове управління є найкращою заміною. Зростає інтерес до використання людського голосу для взаємодії з обчислювальними пристроями. Форрестер каже, що "голосове управління буде наступним полем битви" для технологічних компаній [27].

У цій галузі існує безліч методів та алгоритмів розпізнавання голосу для акустичного моделювання та моделювання мови, таких як Динамічне перекручування часу (DTW), Глибокі нейронні мережі (DNN) та Приховані

моделі Маркова (HMM). В даний час більшість систем розпізнавання голосу базуються на прихованих Марковських моделях, які легко піддаються навчанню і прості у використанні.

Розпізнавання голосу може широко використовуватися для багатьох пристроїв в Інтернеті речей. Одним із найпоширеніших застосувань є автомобільні системи. Прості голосові команди можна використовувати для здійснення телефонних дзвінків, перемикання радіо чи музики та налаштування навігації.

Майже всі відомі компанії надають своє програмне забезпечення для розпізнавання голосу. Візьмемо для прикладу систему голосових команд мобільного телефону. Це дозволяє користувачам викликати певні команди, просто розмовляючи з ним. Це може допомогти користувачам надсилати повідомлення, занотовувати певні події та здійснювати дзвінки. Голосове управління - це ключова технологія для носимих пристроїв, оскільки більшість з них приймають голос як основний підхід до введення даних. Деякі щоденно використовувані носимі пристрої, такі як розумні годинники, можуть збирати інформацію та організовувати її, наприклад, отримувати повідомлення, оновлювати інформацію про погоду та відстежувати фізичну форму користувача.

Системи розпізнавання голосу можна поділити на три типи:

1. Ізольоване мовлення.
2. Розривне мовлення.
3. Безперервне мовлення.

Розпізнавання ізольованого мовлення - це розпізнавання одного слова користувачів, і це відносно простий підхід. Кожне вимовлене слово має свою інформацію у вигляді звукових сигналів. На рисунку 1.3 показаний приклад голосового сигналу в MatLab для слова «Hi» на амплітудно-часовому графіку.

Існують як корисні звуки, так і «шум» голосових сигналів. Щоб відокремити ці дві частини людської мови, використовується метод кепстрального аналізу [28]. MFC (Mel-частота кепстра) - класичний метод

перетворення голосового сигналу в кепстральний. Коефіцієнтами MFC є MFCC (коефіцієнти mel-частоти) [28]. Цей метод винайдено ще у 1970-х роках і використовував MFCC (коефіцієнти кепстральної mel-частоти) та DTW (динамічне викривлення часу) для розпізнавання ізольованої мови. Алгоритм DTW - це підхід для розрахунку подібності між двома часовими рядами, які можуть змінюватися за часом або швидкістю [29].

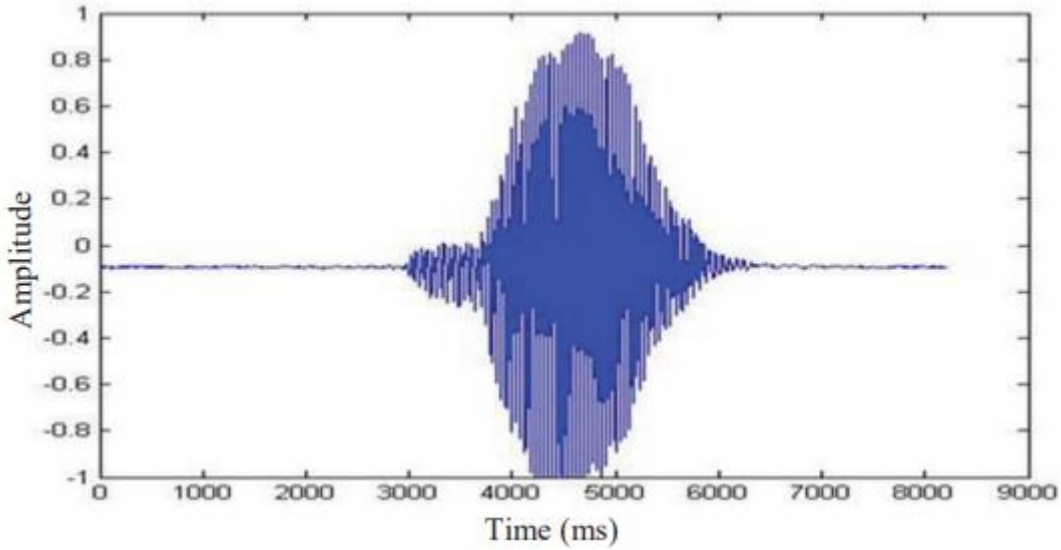


Рисунок 1.3 – Амплітудно-часовий графік представлення слова «Ні»

Розривне мовлення подібне до ізольованої мови. Серед речень є навмисні паузи, і в розмові може бути більше однієї людини. Ця методологія визначає зміни слів у вхідному повідомленні відповідно до розривів між реченнями, а також коли змінюється мовець. І відповідно до цих змін можна виділити невпинні дискурсивні коливання [30].

Безперервне мовлення являє собою інтерпретацію природної розмови людей. Спеціальної паузи немає. Розпізнати таку мову - найскладніша задача. Технологія в цій області ще недостатньо розвинена, а існуючі методи явно не володіють високою точністю для обширних словників.

Схематичне представлення процесу розпізнавання мовлення показаний на рисунку 1.4. У процесі є шість етапів.

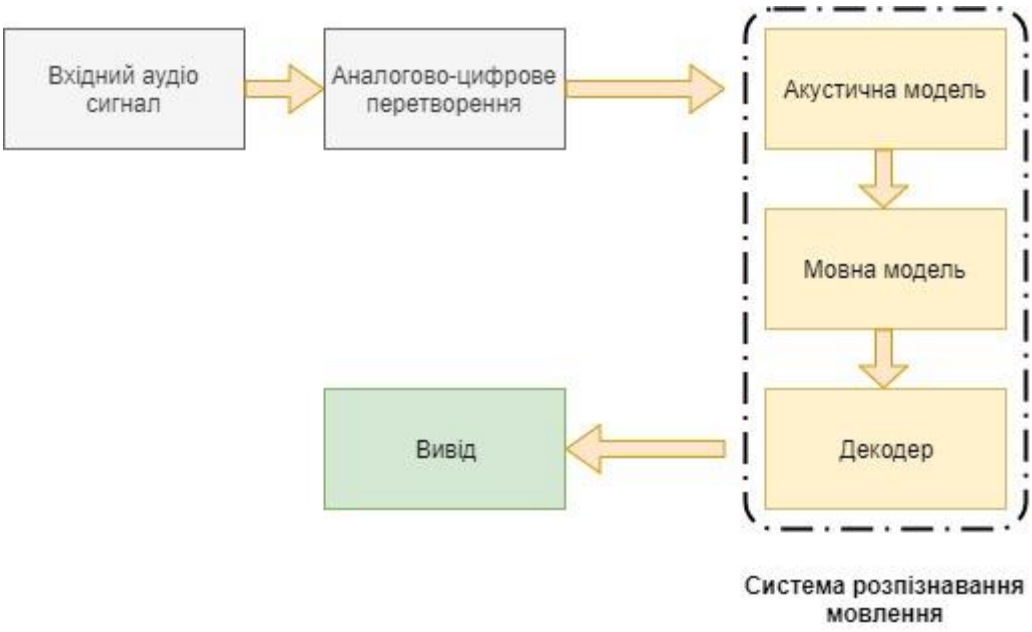


Рис.1.4 – Схематичне представлення процесу розпізнавання мовлення

Отримання вхідного аудіо сигналу. Першим кроком є отримання звукових сигналів через пристрої введення звуку. Мікрофон - це найпоширеніший пристрій введення звуку, який дозволяє користувачам розмовляти з програмним забезпеченням розпізнавання мовлення.

Аналогово-цифрове перетворення. На цьому етапі зібрані аналогові сигнали перетворюються в цифрову форму. Є два етапи: дискретизація та квантування. Дискретизація - це процес передачі безперервного сигналу в дискретні сигнали, тоді як квантування - процес відображення великого набору значень сигналу в малий набір (Рис.1.5).

Акустична модель. Основним завданням акустичного моделювання є створення файлу, що містить статистичне представлення мовлення. Метод прихованих моделей Маркова є широко застосовуваним методом у цій галузі. Це основний статистичний алгоритм розпізнавання за допомогою мовного механізму.

Мовна модель. Моделювання мови використовується для фіксації властивостей певного виду мови. Тоді виходячи з цього, можна передбачити

наступне слово в послідовності, що може допомогти розрізнити слова, які звучать схоже та вибрати те, яке більш імовірно підходить.

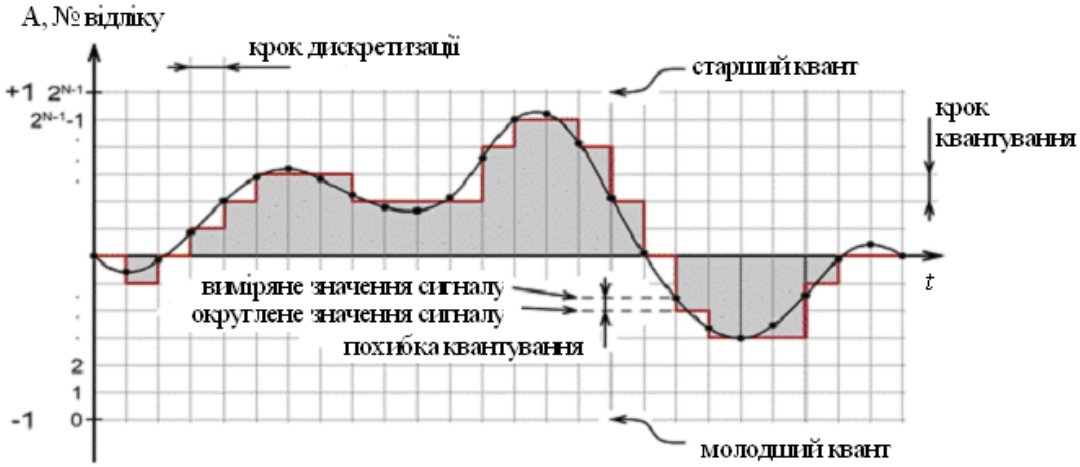


Рис.1.5 – Дискретизація та квантування звукового сигналу [31]

Декодер. Типовий мовленнєвий механізм складається з акустичного моделювання, моделювання мови та декодера. Після перетворення вхідного аудіо у належний формат, наступним кроком є його декодування у відповідний текст [31].

Виведення. Останнім етапом є виведення декодованого аудіо повідомлення. Для пристроїв IoT це може бути переведення повідомлення на інший модуль для визначення команди або транскрипція.

Завданням даного дослідження є удосконалення інформаційного забезпечення створення додатку голосового управління засобом IoT.

Для виконання його необхідно більш детально дослідити архітектуру мережі інтернету речей, охарактеризувати управління засобами IoT. Крім цього необхідно проаналізувати алгоритми, які використовуються для виконання завдання розпізнавання мовлення. Після цього потрібно визначити критичні точки існуючих архітектурних рішень побудови додатків голосового управління засобами IoT та обрати алгоритм та технологію для удосконалення системи в проблемних моментах.

Після цього необхідно побудувати модель для удосконалення інформаційного забезпечення створення додатку голосового управління та оцінити її ефективність, надати поради щодо використання.

Висновки до першого розділу

Було розглянуто основи та історичні особливості створення додатків голосового управління. Робота в напрямку розпізнавання мовлення розпочалася ще в 1950х роках. Переломним моментом стало дослідження статистичних методів розпізнавання мовлення на основі прихованих моделей Маркова. Такі методи використовуються і зараз, але з деякими відмінностями. За останнє десятиліття здобувають популярність методи з використанням нейронних мереж.

Крім цього було складено вимоги до інформаційного забезпечення створення додатку голосового управління. Розглянуто критерії оцінки та прийому алгоритмів і систем розпізнавання мовлення.

Також було розглянуто особливості сфери інтернету речей і особливості голосового управління пристроями IoT. Названо типові процеси розпізнавання мовлення. Сформульовано завдання дослідження.

РОЗДІЛ 2
ОЦІНКА ІСНУЮЧИХ ПІДХОДІВ ДО СТВОРЕННЯ ДОДАТКІВ ГОЛОСОВОГО УПРАВЛІННЯ ЗАСОБОМ IOT

2.1 Характеристика управління засобами IoT та їх типи

Для виконання поставлених задач та взаємозв'язку пристрої IoT оснащені вбудованими датчиками, виконавчими механізмами, процесорами та трансиверами. IoT не є єдиною технологією; скоріше це агломерація різноманітних технологій, які працюють разом у тандемі. Датчики та виконавчі механізми - це пристрої, які допомагають взаємодіяти з фізичним середовищем. Дані, зібрані датчиками, повинні зберігатися та оброблятися розумно, щоб отримати з них користь. Термін датчик визначено широко - мобільний телефон або навіть мікрохвильова піч може вважатися датчиком, якщо він надає дані про його поточний стан (внутрішній стан + середовище). Виконавчий механізм - це пристрій, який використовується для зміни навколишнього середовища, наприклад, регулятор температури кондиціонера.

Зберігання та обробка даних може здійснюватися на кінцях самої мережі або на віддаленому сервері. Якщо можлива будь-яка попередня обробка даних, тоді це зазвичай робиться або самим датчиком, або на якомусь іншому наближеному пристрої. Потім оброблені дані зазвичай надсилаються на віддалений сервер. Можливості зберігання та обробки об'єкта IoT також обмежуються наявними ресурсами, які часто дуже обмежені через обмеження розміру, електропостачання, потужності та обчислювальних можливостей. Як результат, головним завданням дослідження є забезпечення того, щоб на виході було отримано правильний тип даних із бажаним рівнем точності. Поряд із проблемами збору та обробки даних існують проблеми і у взаємодії елементів мережі. Зв'язок між пристроями IoT в основному бездротовий, оскільки вони, як правило, встановлюються в географічно розподілених місцях. Бездротові канали часто мають високі показники спотворень і є ненадійними. У цьому сценарії важлива проблема надійного обміну даними без занадто великої кількості

повторних передач, тому комунікаційні технології є невід'ємною частиною дослідження пристроїв IoT. Тепер, після обробки отриманих даних, потрібно виконати певні дії на основі отриманих висновків. Характер дій може бути різноманітним. Можна безпосередньо модифікувати фізичне оточення за допомогою виконавчих механізмів. Або можна щось робити віртуально, наприклад, надсилати деяку інформацію іншим IoT пристроям. Процес зміни фізичного оточення часто залежить від його стану на той момент часу. Це називається контекстною обізнаністю. Кожна дія здійснюється з урахуванням контексту, оскільки програма може поводитися по-різному в різних контекстах. Наприклад, людині може не сподобатися повідомлення зі свого кабінету, щоб перервати її, коли вона перебуває у відпустці. Датчики, виконавчі механізми, обчислювальні сервери та комунікаційна мережа утворюють основну інфраструктуру фреймворку IoT. Однак існує багато аспектів програмного забезпечення, які потрібно враховувати.

Не існує єдино точного та універсального уявлення щодо архітектури Інтернету речей.

Найбільш базовою архітектурою є тришарова архітектура [32, 33], як показано на рисунку 2.1. Вона була представлена на ранніх етапах досліджень у цій галузі. Він має три шари, а саме - рівень сприйняття, мережу та прикладний рівень.

1. Рівень сприйняття - це фізичний рівень, який має датчики для дослідження та збору інформації про навколишнє середовище. Він розпізнає деякі фізичні параметри або визначає інші IoT пристрої в навколишньому середовищі.
2. Мережевий рівень відповідає за підключення до інших IoT девайсів, мережевих пристроїв та серверів. Його функції також використовуються для передачі та обробки даних датчиків.
3. Рівень застосування відповідає за надання користувачеві певних послуг. Він визначає різні програми, в яких може бути розгорнутий

Інтернет речей, наприклад, розумні будинки, розумні міста та розумне здоров'я.

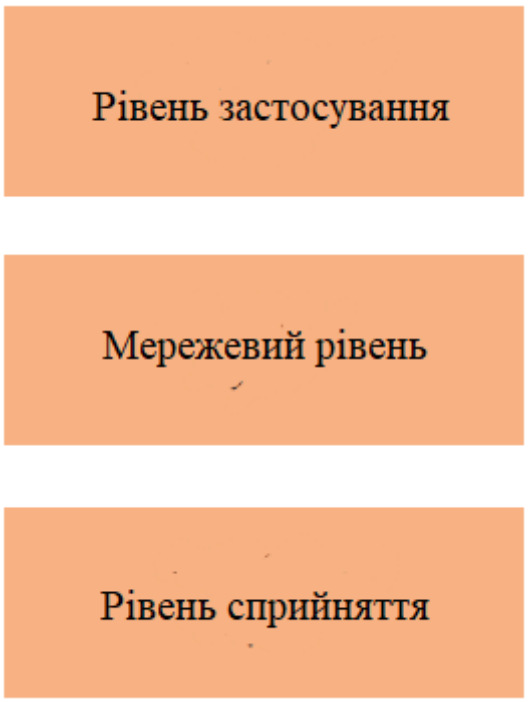


Рисунок 2.1 – Тришарова архітектура Інтернету речей

Тришарова архітектура визначає головну ідею Інтернету речей, але її недостатньо для досліджень IoT, оскільки дослідження часто зосереджуються на більш поглиблених аспектах Інтернету речей.

Іншим представленням є п’ятишарова архітектура, яка додатково включає обробний та бізнес рівні [34]. П’ять шарів - це рівні сприйняття, транспортування, обробки, застосування та бізнесу (Рис. 2.2). Роль шарів сприйняття та застосування така ж, як і архітектура з трьома шарами. Окреслимо функцію решти трьох шарів.

1. Транспортний рівень передає дані датчика з рівня сприйняття на рівень обробки і навпаки через такі мережі, як бездротові мережі, 3G, LAN, Bluetooth, RFID та NFC.
2. Шар обробки також відомий як шар проміжного програмного забезпечення. Він зберігає, аналізує та обробляє величезні обсяги даних, що надходять із транспортного рівня. Він може керувати та

надавати різноманітний набір послуг нижчим шарам. Він використовує безліч технологій, таких як бази даних, хмарні обчислення та модулі обробки великих даних.

3. Бізнес-рівень керує всією системою IoT, включаючи додатки, бізнес-моделі, прибутку та конфіденційність користувачів.

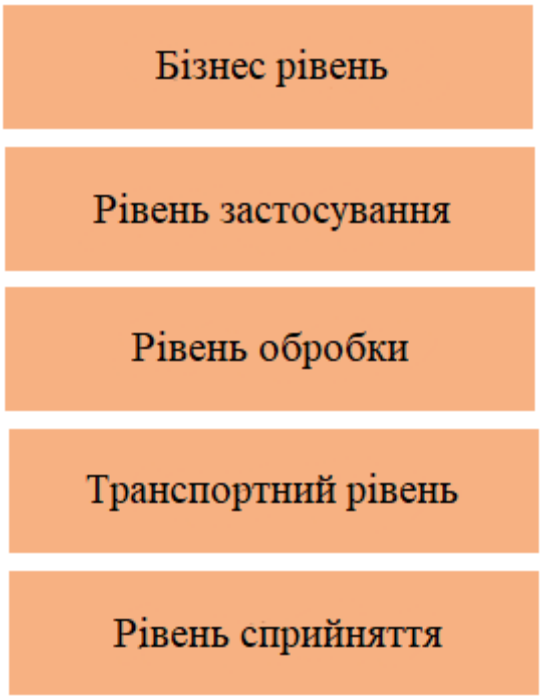


Рисунок 2.2 – П’ятишарова архітектура Інтернету речей

Архітектура на основі хмарних та «туманних» технологій.

У деяких системних архітектурах обробка даних здійснюється централізовано за допомогою хмарних технологій. Така хмарно-орієнтована архітектура має хмарне середовище в центрі, програми над нею та мережу розумних речей під нею. Хмарним обчисленням надається першість, оскільки вони забезпечують велику гнучкість і масштабованість. Вони пропонують такі послуги, як основна інфраструктура, платформа, програмне забезпечення та сховище. Розробники можуть надавати свої засоби зберігання, програмні засоби, засоби обробки даних та інструменти машинного навчання та засоби візуалізації через хмарне середовище.

Останнім часом спостерігається рух до іншої системної архітектури, а саме до «туманних обчислень» [35], де датчики та мережеві шлюзи виконують частину обробки даних та аналітики. Архітектура туману [36] представляє багаторівневий підхід (Рис. 2.3), який включає рівні моніторингу, попередньої обробки, зберігання та захисту між фізичним та транспортним рівнями. Рівень моніторингу контролює потужність, ресурси, відповіді та послуги. Шар попередньої обробки виконує фільтрацію, обробку та аналіз даних датчиків. Тимчасовий рівень зберігання забезпечує такі функції зберігання, як реплікація, розподіл та зберігання даних. Нарешті, рівень безпеки виконує шифрування / дешифрування та забезпечує цілісність та конфіденційність даних. Моніторинг та попередня обробка виконуються на межі мережі перед відправкою даних у хмару.

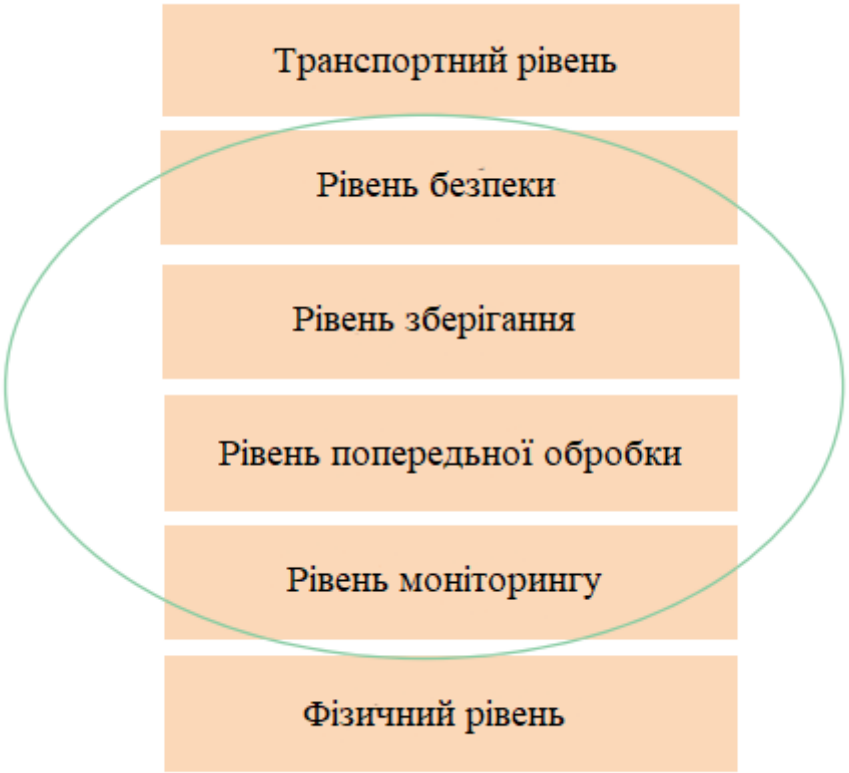


Рисунок 2.3 – Багаторівнева архітектура «туманних технологій»

Часто терміни "туманних обчислень" та "обчислень на кінцях мережі" використовуються як взаємозамінні. Останній термін передує першому і

тлумачиться як більш загальний. «Туманні обчислення», як їх спочатку називали Cisco, відносяться до розумних шлюзів та розумних датчиків, тоді як обчислення на краях мережі мають трохи більш проникливий характер. Ця парадигма передбачає додавання можливостей інтелектуальної попередньої обробки даних до фізичних пристроїв, таких як двигуни, насоси або ліхтарі. Мета полягає в тому, щоб зробити якомога більше попередньої обробки даних на цих пристроях, які називаються кінцями мережі. Така архітектура не відрізняється від архітектури «туманних обчислень».

Основні компоненти.

У типовій архітектурі IoT, що складається з багатьох девайсів, ми розглядаємо пристрої як ботів, де вони можуть встановлювати зв’язки між собою та змінювати їх з часом. Це дозволить нам безперешкодно дозволити пристроям взаємодіяти між собою та досягти складного завдання. Щоб така модель працювала, нам потрібно мати багато взаємодіючих компонентів. Давайте розглянемо деякі основні компоненти такої системи.

1. ID: нам потрібен унікальний метод ідентифікації об’єкта. Ідентифікатор може бути присвоєний об'єкту на основі традиційних параметрів, таких як MAC ID, IPv6 ID, універсальний код продукту або інший спеціальний метод.
2. Метаінформація: поряд із ідентифікатором нам потрібна деяка метаінформація про пристрій, що описує його форму та роботу. Це потрібно для встановлення відповідних взаємозв’язків з пристроєм, а також належного розміщення його у загальній мережі IoT.
3. Контроль безпеки. Власник пристрою може встановити обмеження щодо типів пристроїв, які можуть до нього підключатися.
4. Виявлення сервісів: такий тип системи схожий на хмару послуг, де нам потрібно мати спеціальні каталоги, які зберігають деталі пристроїв, що надають певні види послуг. Дуже важливо постійно оновлювати ці каталоги, щоб пристрої могли дізнаватися про інші пристрої.

5. Управління міжкомпонентною взаємодією: цей модуль керує взаємозв'язками з іншими пристроями. У ньому також зберігаються типи пристроїв, з якими певний пристрій повинен спробувати підключитися залежно від типу наданих послуг. Наприклад, для сенсора світла має сенс встановити взаємозв'язок із датчиком світла.
6. Композиція сервісів. Цей модуль дозволяє об'єднувати набір пристроїв у рівні.

Кінцевою метою наявності такої системи є надання кращих інтегрованих послуг користувачам. Наприклад, якщо людина має датчик потужності зі своїм кондиціонером, і цей пристрій встановлює взаємозв'язок з аналітичним механізмом, тоді ансамбль може надати багато даних про схеми використання кондиціонера. Якщо «соціальна» модель є більш розширеною, а пристроїв набагато більше, то можна порівняти дані зі схемами використання інших користувачів і отримати ще більш значущі дані.

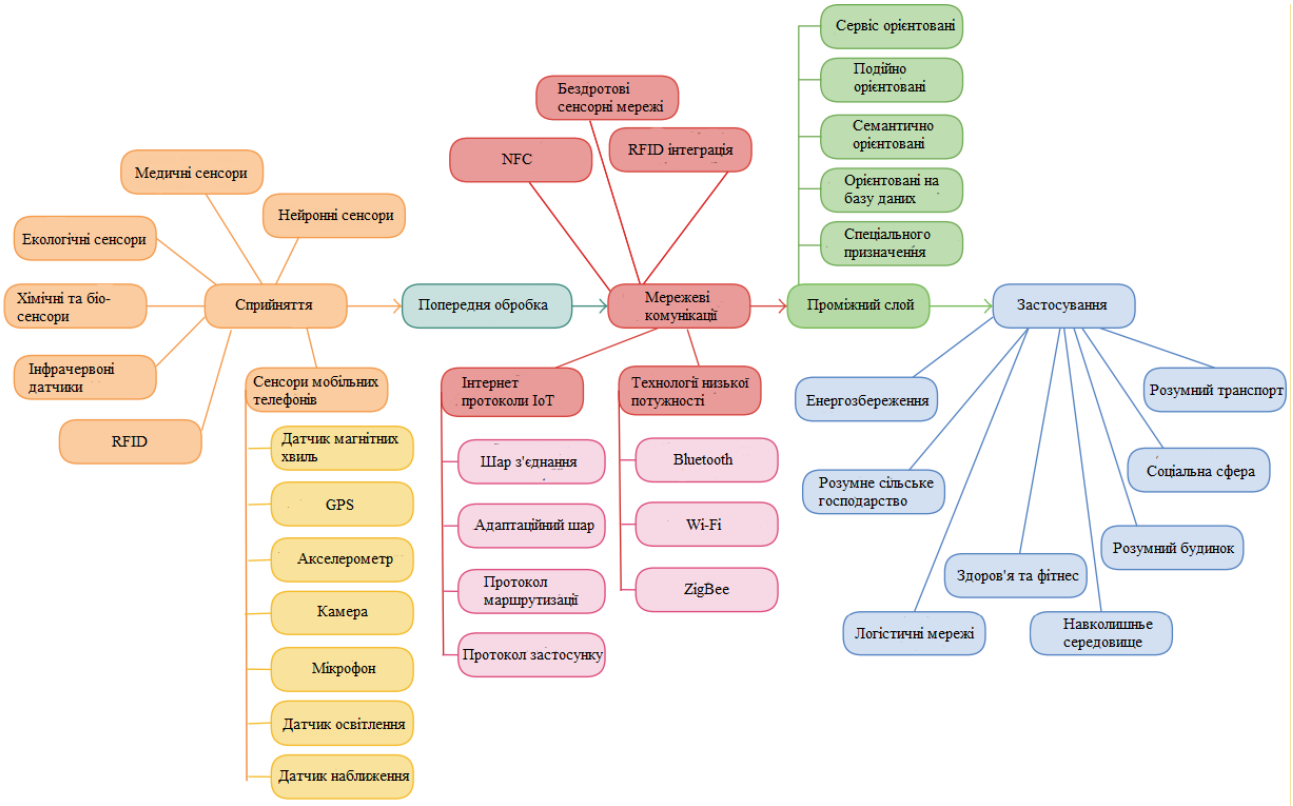


Рисунок 2.4 – Таксономія архітектури інтернету речей [37]

Усі програми IoT повинні мати один або кілька датчиків для збору даних із навколишнього середовища. Датчики є важливими компонентами розумних об'єктів. Одним з найважливіших аспектів Інтернету речей є розуміння контексту, що неможливо без сенсорної технології. Датчики IoT в основному мають невеликі розміри, мають низьку вартість і споживають менше електроенергії. Вони обмежені такими факторами, як ємність акумулятора та простота розгортання. Шмідт та Ван Лаерховен [38] надають огляд різних типів датчиків, що використовуються для побудови інтелектуальних додатків.

Датчики на базі мобільного телефону.

Смартфон - це дуже зручний пристрій, який має безліч вбудованих функцій зв'язку та обробки даних. Зі зростанням популярності смартфонів серед людей, дослідники виявляють інтерес до створення розумних рішень IoT за допомогою смартфонів завдяки вбудованим датчикам [39]. Деякі додаткові датчики також можуть бути використані залежно від вимог. Програми можна будувати на смартфоні, який використовує дані датчиків для отримання значущих результатів. Основні датчики всередині сучасного смартфону:

1. Акселерометр. Він відчуває рух та прискорення мобільного телефону. Зазвичай він вимірює зміни швидкості смартфона в трьох вимірах. Шаблони даних, зафіксовані акселерометром, можуть бути використані для виявлення фізичних навантажень користувача, таких як біг, ходьба та їзда на велосипеді.

2. Гіроскоп дуже точно визначає орієнтацію телефону. Орієнтація вимірюється за допомогою ємнісних змін, коли сейсмічна маса рухається в певному напрямку.

3. Камера та мікрофон є дуже потужними датчиками, оскільки вони фіксують візуальну та звукову інформацію, яку потім можна аналізувати та обробляти для виявлення різних типів контекстної інформації. Наприклад, ми можемо зробити висновок про поточне середовище користувача. Щоб зрозуміти аудіодані, можна використовувати такі технології, як розпізнавання мовлення та акустичні характеристики.

4. Магнітометр виявляє магнітні поля. Це можна використовувати як цифровий компас і в додатках для виявлення присутності металів.

5. GPS (система глобального позиціонування) визначає місце розташування телефону, що є однією з найважливіших контекстних відомостей для розумних додатків.

6. Сенсор світла визначає інтенсивність навколишнього світла. Він може бути використаний для встановлення яскравості екрану та інших застосувань, в яких слід виконати певні дії залежно від інтенсивності навколишнього світла. Наприклад, ми можемо керувати освітленням у кімнаті.

7. Датчик наближення використовує інфрачервоний (ІЧ) світлодіод, який випромінює ІЧ-промені. Ці промені відскакують, коли потрапляють на якийсь предмет. Виходячи з різниці в часі, ми можемо розрахувати відстань. Таким чином можна виміряти відстань до різних предметів від телефону.

Медичні сенсори.

Інтернет речей може бути справді корисним для медичних програм. Ми можемо використовувати датчики, які можуть вимірювати та контролювати різні медичні параметри в організмі людини [40]. Ці програми можуть бути спрямовані на моніторинг стану здоров’я пацієнта, коли він не перебуває в лікарні або коли він один. Згодом вони можуть надавати зворотній зв'язок у режимі реального часу лікареві, родичам або пацієнту. McGrath та Scanaill [41] докладно описали різні датчики, які можна носити на тілі для контролю здоров’я людини.

На ринку доступно багато носимих приладів. Вони оснащені медичними датчиками, які здатні вимірювати різні параметри, такі як частота серцевих скорочень, пульс, артеріальний тиск, температура тіла, частота дихання та рівень глюкози в крові [42]. Ці носимі засоби включають розумні годинники, браслети, моніторингові патчі та розумний текстиль.

Більше того, розумні годинники та фітнес-трекери стають досить популярними на ринку, оскільки такі компанії, як Apple, Samsung та Sony, пропонують дуже інноваційні функції. Наприклад, розумні годинники

включають такі функції, як підключення до смартфона, датчики, такі як акселерометр, та монітор серцевого ритму.

Нейронні сенсори.

Сьогодні можна зрозуміти нервові сигнали мозку, зробити висновок про стан мозку та навчити його кращій увазі та зосередженню. Це називається нейровідгуком [43] (Рис. 2.5).

Рисунок 2.5 – Прилад з нейронними сенсорами [43]

Технологія, що використовується для зчитування мозкових сигналів, називається ЕЕГ (електроенцефалографія) або комп'ютерний інтерфейс мозку. Нейрони всередині мозку взаємодіють за допомогою електронних імпульсів і створюють електричне поле, яке можна виміряти ззовні за частотою. Мозкові хвилі можна розділити на альфа-, бета-, гамма-, тета- та дельта-хвилі залежно від частоти. Виходячи з типу хвилі, можна зробити висновок, спокійний мозок чи заблокований думками. Цей тип нейрофідбеку можна отримати в режимі

реального часу, і його можна використовувати для тренування мозку до фокусування, приділення кращої уваги речам, управління стресом та покращення психічного самопочуття.

Екологічні та хімічні датчики.

Датчики навколишнього середовища використовуються для визначення параметрів у фізичному середовищі, таких як температура, вологість, тиск, забруднення води та повітря. Такі параметри, як температура та тиск, можна виміряти за допомогою термометра та барометра. Якість повітря можна виміряти за допомогою датчиків, які визначають присутність газів та інших твердих частинок у повітрі [44]. Хімічні датчики використовуються для виявлення хімічних та біохімічних речовин. Ці датчики складаються з елемента розпізнавання та перетворювача. «Електронний ніс» та «електронний язик» - це технології, які можуть бути використані для відчуття хімічних речовин відповідно до запаху та смаку [45]. Такі сенсори складаються з масиву хімічних датчиків у поєднанні з попереднім програмним забезпеченням для розпізнавання шаблонів та образів.

Виконавчі механізми.

Виконавчий механізм - це пристрій, який може впливати на зміну навколишнього середовища, перетворюючи електричну енергію в певну форму корисної енергії. Деякі приклади - нагрівальні або охолоджувальні елементи, динаміки, ліхтарі, дисплеї та двигуни. Приводи, що викликають рух, можна класифікувати на три категорії, а саме на електричні, гідравлічні та пневматичні приводи, залежно від їх роботи. Гідроприводи полегшують механічний рух за допомогою рідини або гідравлічної сили. Пневмоприводи використовують тиск стисненого повітря, а електричні - електричну енергію.

2.2 Алгоритми реалізації проекту створення додатку голосового управління

Розпізнавання мови - один із основних напрямків обробки мовлення, також відоме як автоматичне розпізнавання мовлення (ASR). Це процес перетворення

мовленнєвого сигналу в послідовність слів (тобто вимовлених слів у текст) за допомогою алгоритму, реалізованого комп'ютерною програмою [46].

Основною метою системи ASR є висунення гіпотези про найімовірнішу дискретну послідовність символів з усіх дійсних послідовностей мови L із заданого акустичного входу O [47]. Як зазначено вище, вхідні дані розглядаються як сукупність дискретних спостережень, таких що (2.1):

$$O = o_1, o_2, o_3, \dots, o_t \quad (2.1)$$

Аналогічно послідовність символів, яку слід розпізнати, визначається як (2.2):

$$W = w, w_2, w_3, \dots, w_n \quad (2.2)$$

Основна мета системи ASR може бути виражена у вигляді (2.3):

$$\widehat{W} = argmax\ P(W|O)\ , \text{де}\ W\ \in L \quad (2.3)$$

З цього рівняння випливає, що для даної послідовності W та акустичної вхідної послідовності O слід визначити ймовірність P (W | O). Можна застосувати теорему Байєса до цієї ймовірності, щоб отримати наступне рівняння (2.4):

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)} \quad (2.4)$$

Величини в правій частині рівняння легше обчислити, ніж P (W | O). P (W) визначається як попередня ймовірність самої послідовності. Це обчислюється з використанням попередніх знань про випадки послідовності W. Оскільки P(O) однаковий для кожного кандидата W, то рівняння можна спростити до (2.5):

$$\widehat{W} = argmax\ \frac{P(O|W)P(W)}{P(O)} = argmax\ P(O|W)P(W), \text{де}\ W\ \in L \quad (2.5)$$

Імовірність P (O | W), яка є ймовірністю акустичного входу O, враховуючи послідовність W, визначається як ймовірність спостереження, може бути названа акустичною оцінкою. Цю величину можна визначити за допомогою статистичного методу, прихованої моделі Маркова, використанням нейронних мереж та глибокого навчання.

Типова система розпізнавання мовлення розроблена з основними компонентами, які включають акустичний вхідний модуль (фронтенд), базу

знань , що містить акустичну модель, словник та мовну модель, декодер який об’єднує обчислювальні модулі та реалізовує основний функціонал (Рис. 2.6). Акустичний вхідний модуль забезпечує перетворення мовленнєвого сигналу у відповідні функції, що забезпечує отримання вхідного повідомлення для розпізнавання.

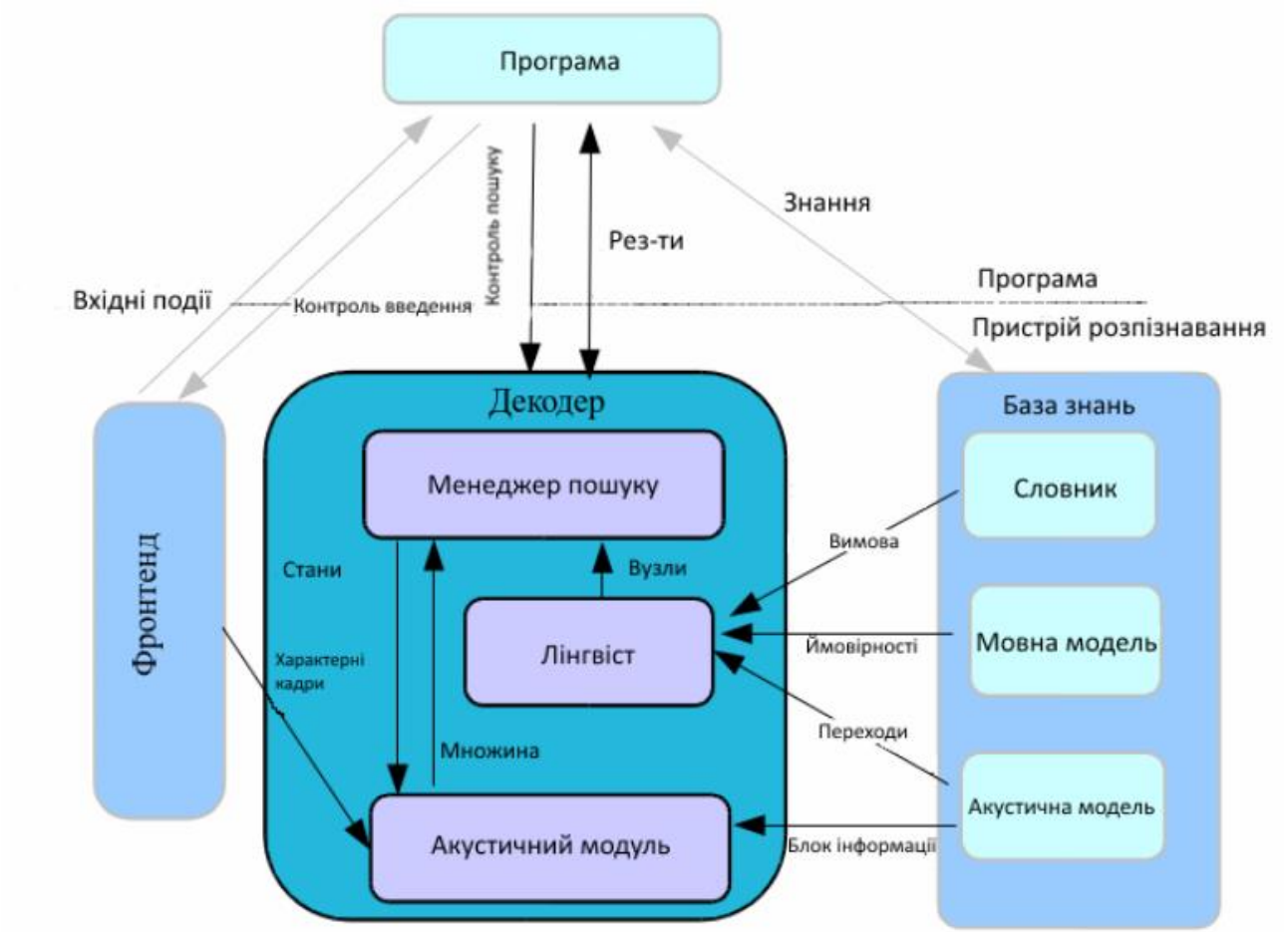


Рисунок 2.6 – Архітектура системи розпізнавання мовлення

Форма вхідного звукового сигналу від мікрофона перетворюється в послідовність акустичних векторів фіксованого розміру - це процес, який називається виділенням ознак. Параметри моделей слів / телефонів оцінюються на основі акустичних векторів навчальних даних. Декодер функціонує шляхом пошуку у всіх можливих послідовностях слів, щоб знайти таку послідовність, яка, імовірніше за все, може бути сгенерована. Ймовірність визначається як акустична модель P (O | W), а P (W) визначається мовною моделлю.

Функціональність автоматичної системи розпізнавання мовлення може бути описана як вилучення ряду параметрів мовлення з акустичного мовленнєвого сигналу для кожного слова або одиниць, що створюють слова. Параметри мови описують слово або підслово шляхом їх варіації в часі, і вони разом створюють шаблон, що характеризує слово або підслово. На етапі навчання оператор прочитає всі слова з словника поточної програми. Шаблони слів зберігаються, і пізніше, коли слово слід розпізнати, його образ порівнюється із шаблонами, що зберігаються, і вибирається слово, яке найкраще йому відповідає. Цей прийом зазвичай називають розпізнаванням зразків.

Акустичний інтерфейс передбачає обробку сигналу та вилучення ознак. При розпізнаванні мовлення головною метою етапу вилучення ознак є обчислення невеликої послідовності векторів ознак, що забезпечує компактне представлення даного вхідного сигналу [48]. Вилучення ознак зазвичай виконується у три етапи. Перший етап називається аналізом мовлення або акустичним інтерфейсом. Він вибирає якийсь спектр часового сигналу і генерує необроблені характеристики, що описують огинаючу потужність спектра коротких інтервалів мовлення. На другому етапі складається розширений векторний об’єкт, що складається із статичних та динамічних об’єктів. Нарешті, останній етап (який не завжди присутній) перетворює ці розширені вектори функцій у більш компактні та надійні вектори, які потім подаються в декодер.

Немає універсальної функції, придатної для конкретного застосування, але вибір функцій оброблення повинен відповідати наступним властивостям: вони повинні дозволяти автоматичній системі розрізняти різницю у схожих за звучанням мовленнєвих сигналів, повинні дозволяти автоматичне створення акустичних моделей для цих звуків без надмірної кількості навчальних даних, і вони повинні демонструвати статистичні дані, які в значній мірі є незмінними для мовців та навколишнього середовища.

Для того, щоб знайти деяку статистично релевантну інформацію з вхідних даних, важливо мати механізми зменшення інформації кожного сегмента в звуковому сигналі на порівняно невелику кількість параметрів або ознак. Ці

ознаки повинні описувати кожен сегмент таким характерним чином, щоб інші подібні сегменти можна було згрупувати разом, порівнюючи їх ознаки. Є велика кількість способів опису мовленнєвого сигналу з точки зору параметрів. Деякі з методів вилучення ознак - це Принциповий компонентний аналіз (PCA), Лінійний дискримінантний аналіз (LDA), Незалежний аналіз компонентів (ICA), Лінійне інтелектуальне кодування (LPC), Кепстральний аналіз, Аналіз частоти за частотою, Аналіз банку фільтрів, Коефіцієнти Mel-частоти кепстра (MFCC), вилучення функцій на основі ядра, динамічне вилучення характеристик, характеристики на основі вейвлетів, спектральне віднімання та віднімання середнього кепстру (CMS) [49].

У шумному середовищі для розпізнавання мовлення існує багато методів вилучення акустичних ознак, таких як амплітуда пікового перетину нуля (ZCPA), середнє локалізоване виявлення синхронії (ALSD), мінімальна дисперсія сприйняття без спотворень (PMVDR), нормовані кепстральні коефіцієнти (PNCC) ефективно застосовуються інваріантні функції інтеграції (IIF), спектрограма амплітудної модуляції, кепстральні коефіцієнти частоти гамматону, розріджене слухове відтворююче ядро (SPARK) та функції банку фільтрів Габора [49].

Найпоширенішим методом вилучення мовленнєвих ознак є набір характеристик Mel-частотного кепстрального коефіцієнта (MFCC). Процес вилучення характеристик MFCC має багато етапів (Рис. 2.7).

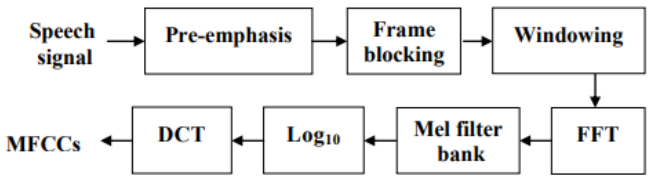


Рисунок 2.7 – MFCC метод для вилучення мовленнєвих ознак [49]

Етапи MFCC:

- Попереднє акцентування (Pre-emphasis) - Цей етап використовується для посилення енергії на високих частотах вхідного мовленнєвого

сигналу. Це дозволяє інформації в цих регіонах бути більш впізнаваною під час навчання та розпізнавання моделі HMM.

- Windowing - цей етап нарізає вхідний сигнал на дискретні часові сегменти, фрейми. Це робиться за допомогою вікна шириною N мілісекунд і зі зміщенням по довжині M мілісекунд. Вікно Хеммінга зазвичай використовується для запобігання крайовим ефектам, пов’язаними з різкими змінами всередині прямокутного вікна.
- Дискретне перетворення Фур'є (DFT) - застосовується до віконного мовленнєвого сигналу, що призводить до подання сигналу за величиною та фазою.
- Mel Filter Bank - Хоча отриманий спектр DFT містить інформацію на кожній частоті, людський слух менш чутливий на частотах вище 1000 Гц. Ця концепція також має прямий вплив на продуктивність систем ASR; тому спектр викривляється за допомогою логарифмічної шкали Мела. Частоту Мела можна розрахувати за рівнянням (2.6):

$$mel(f) = 1127\ln\left(1 + \frac{f}{700}\right) \qquad (2.6)$$

  Для того, щоб створити цей ефект на спектрі DFT, побудований банк фільтрів, відомий як трикутні фільтри, з фільтрами, розподіленими однаково нижче 1000 Гц і розташованими логарифмічно вище 1000 Гц. Вихідні дані фільтрації сигналу DFT кожним фільтром Мела відомі як спектр Мела.
- Log – логарифмування спектру забезпечує створення коефіцієнтів спектра Мела.
- Особливості Delta MFCC - Для того, щоб вловити зміни мовлення від фрейму до фрейму, також обчислюється і використовується перша та друга похідна коефіцієнтів MFCC.

Акустична модель - одне з найважливіших джерел знань для автоматичної системи розпізнавання мовлення, яка представляє акустичні особливості для

фонетичних одиниць, що розпізнаються. При побудові акустичної моделі одним з фундаментальних і важливих питань є вибір основних модельних одиниць. Взагалі кажучи, коли вказано цільову мову виступу, для акустичного моделювання можна використовувати декілька типів одиниць. Різні блоки акустичного моделювання можуть суттєво змінити ефективність системи розпізнавання мовлення.

Акустичне моделювання мови, як правило, відноситься до процесу встановлення статистичних уявлень для послідовностей векторних ознак, обчислених з мовленнєвої звукової хвилі. Приховані моделі Маркова (HMM) - одна з найбільш часто використовуваних статистичних моделей для побудови акустичних моделей. Інші акустичні моделі включають сегментарні моделі, суперсегментарні моделі (включаючи приховані динамічні моделі), нейронні мережі, моделі максимальної ентропії та (приховані) умовні випадкові поля тощо. Кожному з цих статистичних подань присвоюється мітка, яка називається фонематичною акустичною моделлю, створюється шляхом створення великої бази даних мови, що називається мовленнєвим корпусом, та використання спеціальних алгоритмів навчання для створення статистичних подань для кожної фонеми на мові. Кожна фонема має свій власну модель HMM. Мовний декодер прослуховує чіткі звуки, що промовляються користувачем, а потім шукає відповідну модель HMM в акустичній моделі. Кожне вимовлене слово w розкладається на послідовність основних звуків, які називаються базовими звуковими одиницями. Акустична модель описує ймовірність конкретного спостереження за базовою звуковою одиницею.

Мовна модель - це сукупність обмежень щодо послідовності слів, прийнятних для даної мови. Ці обмеження можуть бути представлені, наприклад, правилами генеративної граматики або просто статистикою щодо кожної пари слів, оціненої в навчальному корпусі. Хоча існують слова, що мають подібне звучання, людям, як правило, не важко розпізнати це слово. Це головним чином тому, що вони знають контекст, а також досить добре уявляють, які слова або фрази можуть траплятися в контексті. Надання цього контексту системі

розпізнавання мовлення є метою мовної моделі. Мовна модель визначає, які дійсні слова в мові та в якій послідовності вони можуть відбуватися.

Мовні моделі зазвичай навчаються, тобто ймовірності n-грам оцінюються шляхом спостереження за послідовностями слів в сукупностях текстів, що містять, як правило, мільйони словесних лексем, і шляхом зменшення заплутаності на навчальних даних [47]. Однак було відмічено, що зменшення заплутаності не обов'язково призводить до кращих результатів розпізнавання мовлення. Тому алгоритми, які покращують мовні моделі на основі їх впливу на розпізнавання мови, особливо привабливі. Мовна модель визначає розподіл ймовірності слів, які диктор може вимовити наступними, з огляду на історію вимовлених слів. Поширеними мовними моделями є біграмні і триграмні моделі. Ці моделі містять обчислені ймовірності групування двох чи трьох конкретних слів у послідовності, відповідно.

На етапі декодування завдання полягає у пошуку найбільш вірогідної послідовності слів W з урахуванням послідовності спостереження O та акустико-фонетичної мовної моделі. Проблему декодування можна вирішити за допомогою алгоритмів динамічного програмування. Замість оцінки ймовірностей усіх можливих модельних шляхів, що генерують O, основна увага приділяється пошуку єдиного шляху через мережу, що дає найкращий збіг з O. Для оцінки найкращої послідовності стану для даної послідовності спостереження часто використовується алгоритм Вітербі [50]. У випадку більших словників було б складно розглянути всі можливі слова під час рекурсивної частини алгоритму Вітербі. Для вирішення цієї проблеми для ітерації Вітербі можна використовувати збалансований пошук, лише слова з шляховими імовірностями вище порогового значення враховуються при розширенні шляхів до наступного кроку. Такий підхід пришвидшує процес пошуку, але зменшує точність декодування. Алгоритм Вітербі передбачає, що кожен з найкращих шляхів в момент часу t повинен бути продовженням кожного з найкращих шляхів, що закінчується в момент часу t - 1, що, як правило, не відповідає дійсності. Шлях, який на початку здається менш вірогідним, ніж інші,

може перетворитися на найкращий шлях для послідовності в цілому. Наприклад, найбільш вірогідна послідовність фонем не завжди має відповідати найбільш вірогідній послідовності слів. Цю проблему вирішують розширені алгоритми Вітербі та двосторонні алгоритми [47].

2.2.1 Прихована модель Маркова для розпізнавання мовлення

Прихована модель Маркова (HMM) – статистичний алгоритм. HMM інтерпретує процес, аналізуючи шаблон послідовності спостережуваних символів. HMM складається з подвійно стохастичного процесу, в якому базовий (або прихований) стохастичний процес можна опосередковано зробити шляхом аналізу послідовності спостережуваних символів іншого набору стохастичних процесів. HMM містить (приховані) стани, які представляють прихований атрибут модельованого процесу. Підходи, засновані на HMM, широко використовуються для аналізу ознак або спостережень, для прогнозування найбільш вірогідної послідовності станів. HMM - це стохастична модель дискретних подій і варіація ланцюга Маркова, ланцюг пов'язаних станів або подій, в яких наступний стан залежить лише від поточного стану системи. Стани HMM приховані (такі, що не можна зробити висновок лише із спостережуваних символів) [52].

Ланцюг Маркової моделі містить усі можливі стани системи та ймовірність переходу з одного стану в інший (Рис. 2.8).

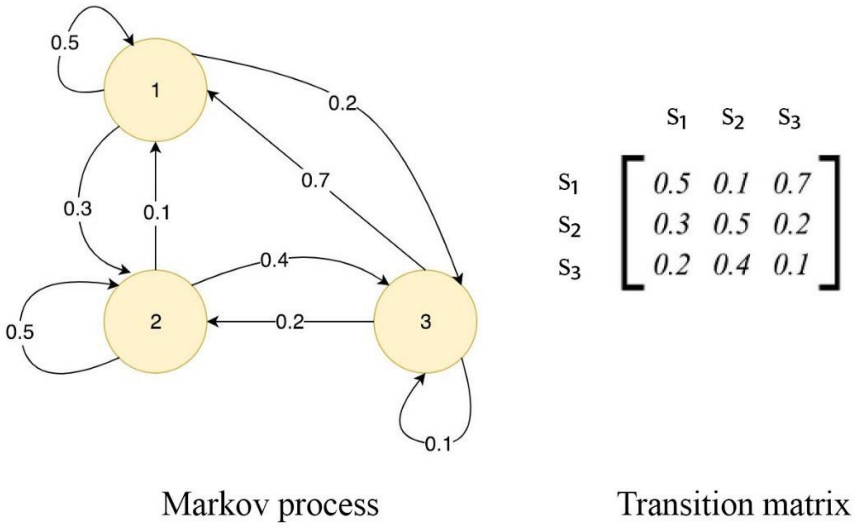


Рисунок 2.8 – Ланцюг Маркової моделі [51]

Ланцюг Маркова першого порядку передбачає, що наступний стан залежить лише від поточного стану. Спрощено це називається ланцюгом Маркова (Формула 2.7).

$$P\left(X_{n+1} = x \middle| X_1 = x_1, X_2 = x_{2,\ldots,} X_n = x_n\right) = P(X_{n+1} = x | X_n = x_n) \quad (2.7)$$

З цією моделлю буде набагато легше впоратися. Однак у багатьох системах ML не всі стани можна спостерігати, і ці стани називаються прихованими або внутрішніми станами. Деякі можуть розглядати їх як приховані фактори для вхідних даних. Наприклад, може бути нелегко дізнатися, щаслива людина чи ні. Внутрішній стан буде {H або S}. Але можна отримати деякі підказки із спостережуваних факторів. Наприклад, коли людина щаслива, у неї є 0,2 шансу подивитися фільм, але коли їй сумно, цей шанс сягає 0,4 (Рис.2.9). Ймовірність спостереження з урахуванням внутрішнього стану називається ймовірністю емісії. Ймовірність переходу з одного внутрішнього стану в інший називається ймовірністю переходу.

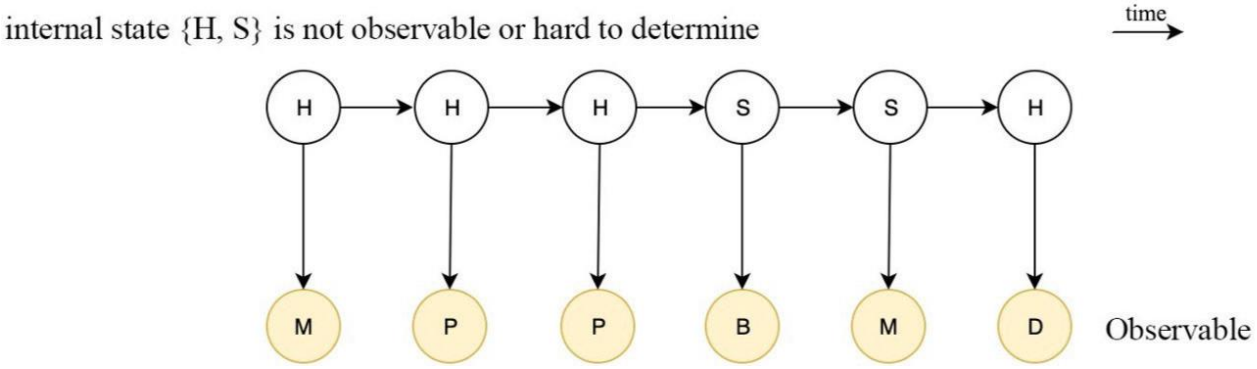


Рисунок 2.9 – Ланцюг Маркова спостереження станом людини [51]

Для розпізнавання мовлення спостереження - це вміст кожного аудіо сигналу. Для його представлення використовуються параметри MFCC.

Враховуючи вивчення моделі HMM, можна використовувати прямий алгоритм для обчислення ймовірності спостережень. Мета цього етапу - підсумувати ймовірності спостережень для всіх можливих послідовностей станів (Формула 2.8):

$$p(X) = \sum_S p(X, S) = \sum_S p(X \mid S) p(S), \quad (2.8)$$

де X – досліджувані події,

$\sum_s$ – сума всіх можливих послідовностей внутрішніх станів,

p(X | S) – розрахована ймовірність емісії,

p(S) - розрахована ймовірність переходу.

Але підводження усіх можливих послідовностей станів по одному має експоненціальну складність.

Якщо виражати обчислення рекурсивно, можна розбити задачу на проміжні етапи. У HMM проблема вирішується в момент часу t, використовуючи результат часу t-1 та / або t + 1. Коло внизу представляє HMM прихований стан j в момент часу t [52, 53]. Тож навіть кількість послідовностей станів з часом збільшується експоненційно, можна розв'язати її лінійно, якщо виразити обчислення рекурсивно з часом (Рис. 2.10).

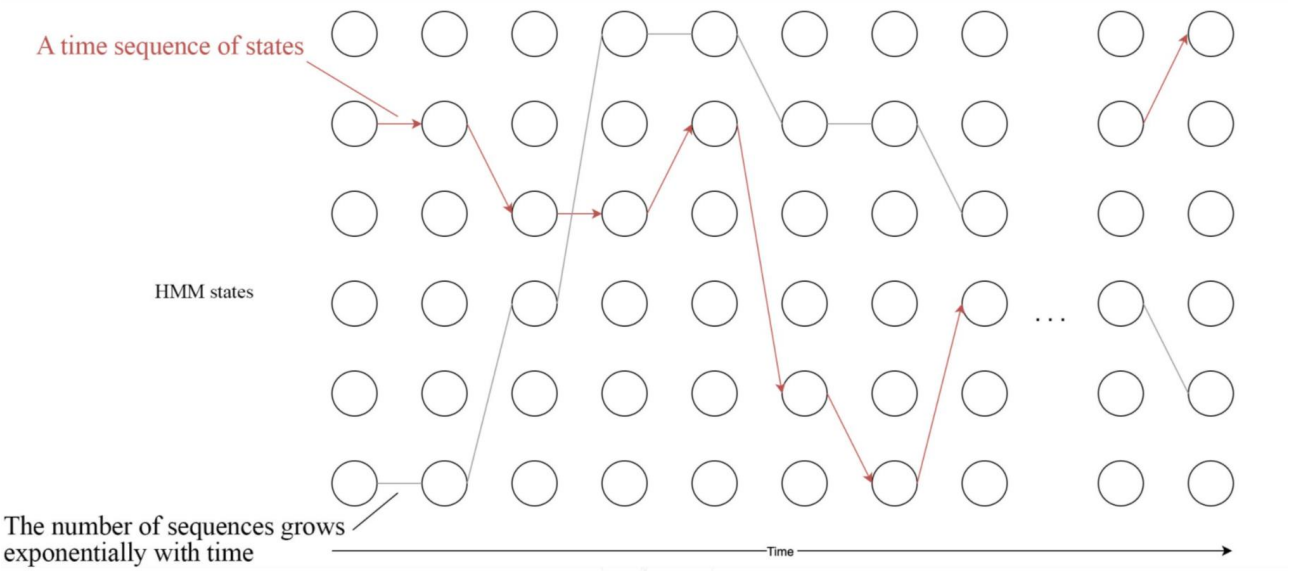


Рисунок 2.10 – проходження по всіх станах ланцюга Маркова [51]

Це принцип динамічного програмування, який розбиває експоненціальну складність. У момент часу t ймовірність спостережень до часу t становить (2.9):

$$P(x_1, x_2, \dots, x_t|\lambda) = \sum_J P\,(x_1, x_2, \dots, x_t, S_t = s_j|\lambda), \qquad (2.9)$$

Де x_1 – спостереження до часу t,

j – сума через усі внутрішні стани,

λ – параметри прихованої моделі Маркова.

Оскільки поточне спостереження залежить лише від поточного стану, α можна виразити як (2.10):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(x_t), \quad (2.10)$$

де t – час,

j – стан j,

$\sum_{i=1}^{N} P(\dots)$ – сума через усі внутрішні стани,

a_{ij} – ймовірність переходу від стану i до стану j,

b_j – ймовірність емісії.

Тож він має рекурсивні відносини. Нижче наведені етапи підрахунку ймовірності спостережень за моделлю λ за допомогою рекурсії. Замість того, щоб підсумовувати кожну послідовність станів самостійно, α обчислюється від першого кроку часу до кінця (час T). Якщо існує k внутрішніх станів, складність буде лише O (k^2T), а не експоненціальною.

1. Ініціалізація (Формула 2.11):

$$\alpha_1(j) = \pi_j b_j(x_1), \quad (2.11)$$

де π_j – початковий стан дистрибуції,

b_j – ймовірність спостереження даного стану j

2. Для кожного часового етапу (Формула 2.12):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(x_t), \quad (2.12)$$

де t – час,

a_{ij}– ймовірність переходу,

$\sum_{i=1}^{N} P(\dots)$ – сума через усі внутрішні стани,

$a_{t-1}(i)$ – ймовірність, що всі попередні спостереження дадуть стан i,

b_j – ймовірність, що поточне спостереження дасть стан j.

3. Результат (Формула 2.13):

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i), \quad (2.13)$$

де $P(X|\lambda)$ – усі спостереження,

$\sum_{i=1}^{N}$ - сума усіх можливих станів,

$\alpha_T(i)$ – останній крок.

На рисунку 2.11 зображено приклад, який починається з початкового розподілу стану зліва. Потім відбувається поширення значення α праворуч. Обчислення α проводиться для кожного стану і повторюється для кожного кроку часу (Рис. 2.11).

Далі, враховуючи модель HMM, можна знайти внутрішні стани з урахуванням послідовності спостережень. Цей процес називається декодуванням. Це основне призначення алгоритму HMM в процесі розпізнавання мовлення. Внутрішні стани аудіо сигналів представляють звуки. Розпізнавання мовлення можна розглядати як пошук цих внутрішніх станів, враховуючи аудіо сигнали.

x_{t+1}

x_t	Happy	Sad
Happy	0.99	0.01
Sad	0.1	0.9

$P(y_t|x_t)$:

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

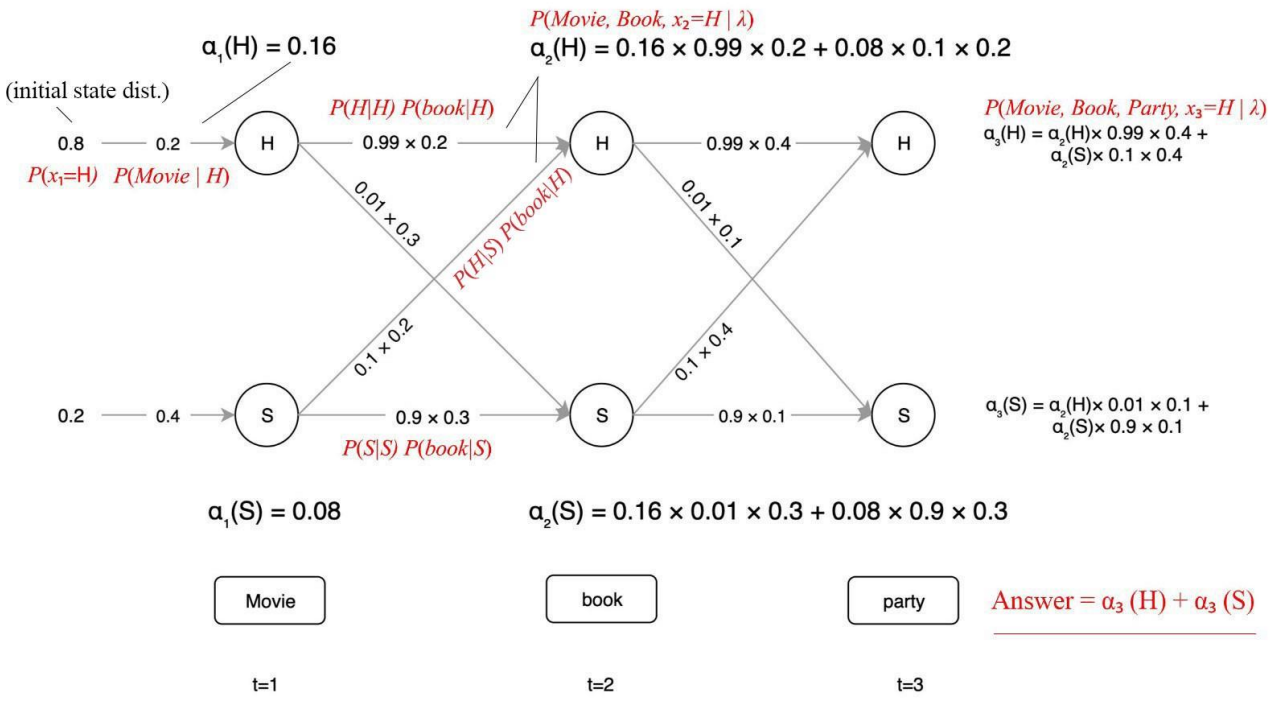


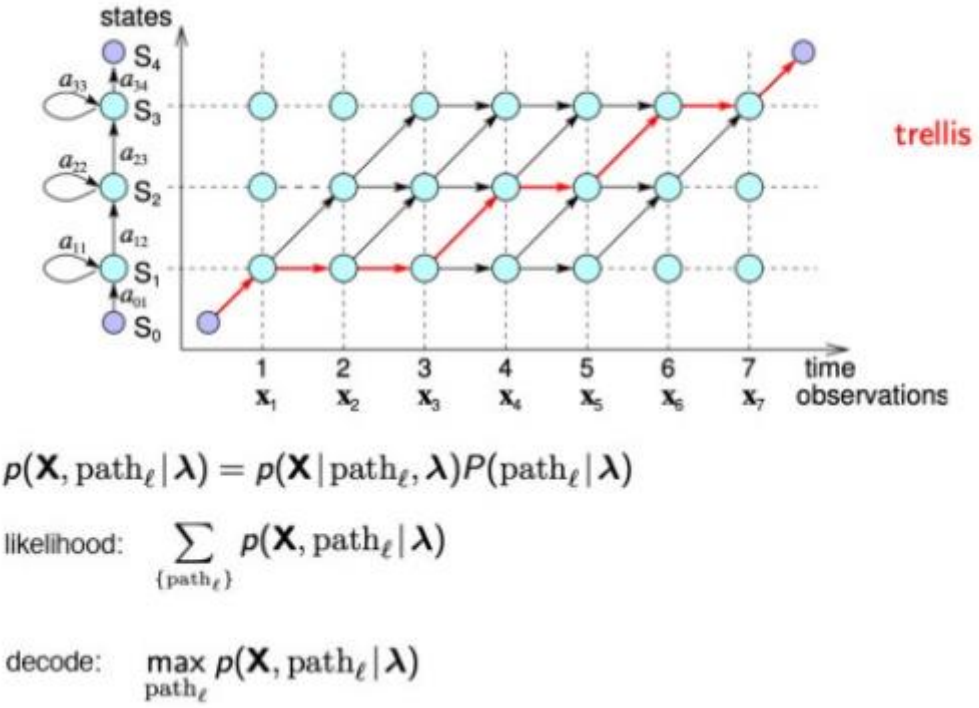
Рисунок 2.11 – Приклад дистрибуції станів та розрахунку ймовірностей [51]

Враховуючи стан j у момент часу t, vt (j) - спільна ймовірність послідовності спостереження з оптимальною послідовністю стану (2.14):

$$\nu_t = \max_{s_0, s_1, \ldots, s_{t-1}} P(S_0, S_1, \ldots, S_{t-1}, x_1, x_2 \ldots x_t, S_t = s_j | \lambda) \qquad (2.14)$$

$$\nu_t = \max_{i=1}^{N} \nu_{t-1}(i) a_{ij} b_j(x_t)$$

Рівняння подібне до прямого алгоритму, за винятком того, що підсумовування замінено функцією максимуму. Замість того, щоб підсумовувати всі можливі послідовності станів у прямому алгоритмі, алгоритм Вітербі бере найбільш вірогідний шлях (Рис. 2.12).



$$p(\mathbf{X}, \text{path}_\ell | \boldsymbol{\lambda}) = p(\mathbf{X} | \text{path}_\ell, \boldsymbol{\lambda}) P(\text{path}_\ell | \boldsymbol{\lambda})$$

likelihood: $\sum_{\{\text{path}_\ell\}} p(\mathbf{X}, \text{path}_\ell | \boldsymbol{\lambda})$

decode: $\max_{\text{path}_\ell} p(\mathbf{X}, \text{path}_\ell | \boldsymbol{\lambda})$

Рисунок 2.12 – Алгоритм Вітербі для визначення послідовності спостережень [51]

Пошук внутрішніх станів, які максимізують ймовірність спостережень, подібний до методу вірогідності. Різниця в тому, що підсумовування замінюється на максимальну функцію (Рис. 2.13).

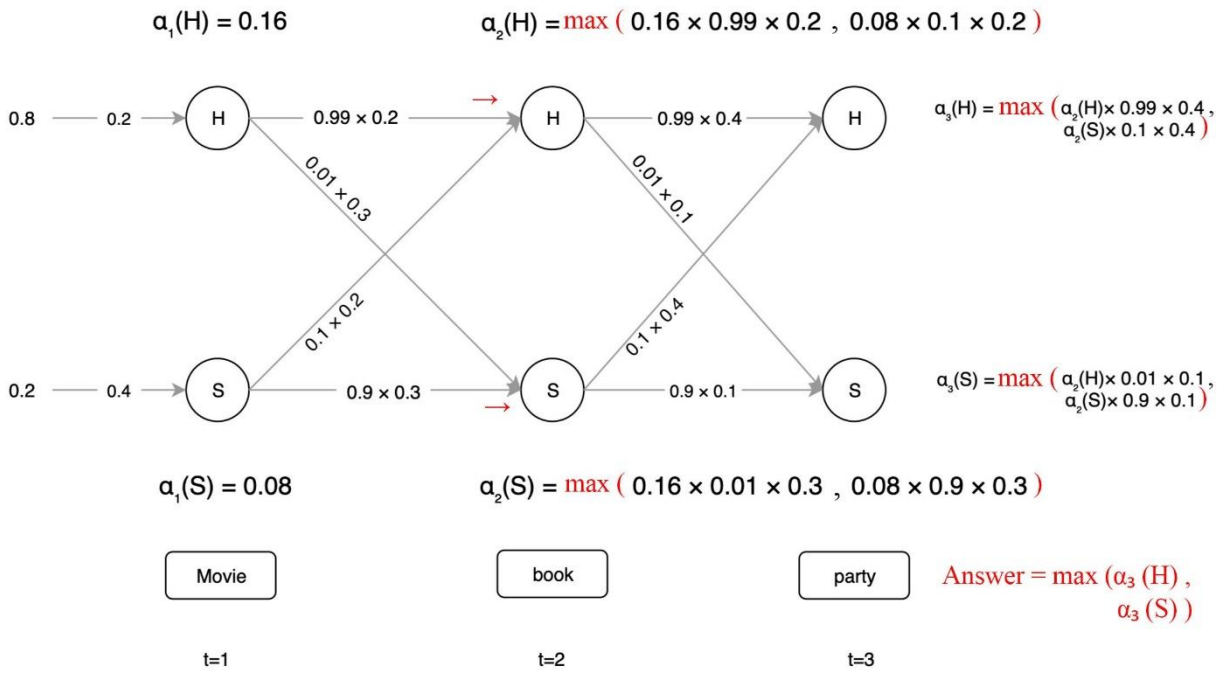


Рисунок 2.13 – Алгоритм пошуку внутрішніх станів [51]

У цьому алгоритмі також записується максимальний шлях, який веде до кожного вузла в момент часу t (червона стрілка вище), тобто задача полягає у відстеженні оптимального шляху для кожного вузла. Наприклад, із щасливого стану H при t = 1 відбувається перехід до щасливого стану H при t = 2 (Рис.2.14).

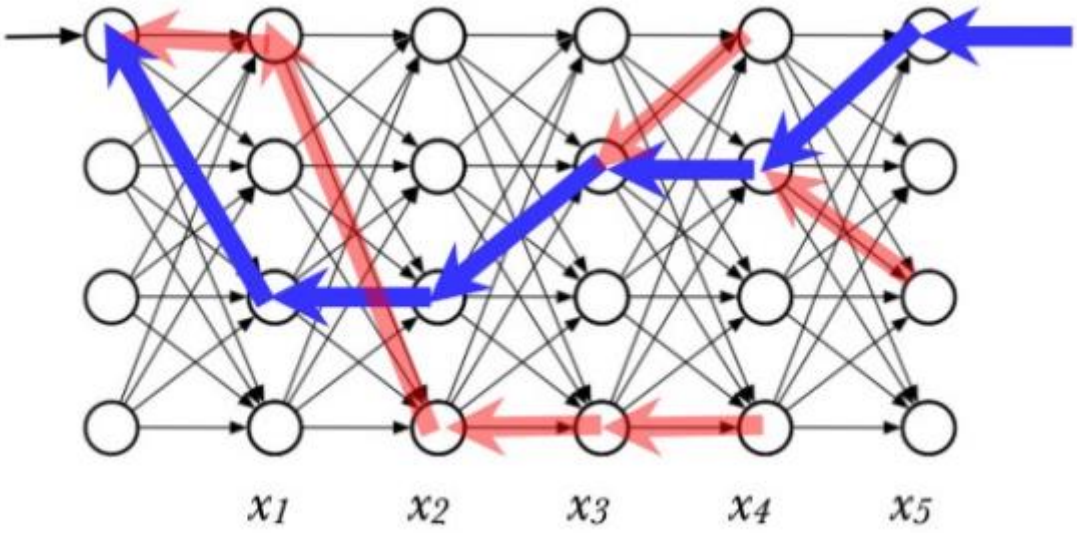


Рисунок 2.14 – Пошук оптимальної послідовності спостережень [51]

Процес навчання НММ відбувається за допомогою алгоритму Баума – Велча (алгоритм «вперед-назад») для вивчення переходу та ймовірності емісії. Це завдання звучить складним, оскільки обидві ймовірності дуже заплутані в розрахунках. Але з певної точки зору, знаючи ймовірність стану в момент часу t, можна отримати ймовірність емісії та ймовірність переходу. Знаючи ці дві ймовірності, можна отримати розподіл стану в момент часу t. На кожному кроці відбувається оптимізація однієї прихованої змінної, одночасно фіксуючи інші. Навіть для неперервного простору працювати доводиться з обмеженою точністю, і тому існують скінченні стани для дослідження та вдосконалення. Отже, якщо зберегти знання про всі ітерації, рішення буде сходитися.

Враховуючи момент часу t із внутрішнім станом i, ймовірність побачити всі спостереження до цього часу (2.15).

$$\alpha_i(t) = P(Y_1 = y_1, \dots, Y_t = y_t, X_t = i \mid \theta) \tag{2.15}$$

В заданий в момент часу t з внутрішнім станом i, ймовірність побачити всі майбутні спостереження (2.16).

$$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \dots, Y_T = y_T \mid X_t = i, \theta) \tag{2.16}$$

Ймовірність емісії в даному стані i в момент часу t, враховуючи спостереження (2.17).

$$\gamma_i(t) = P(X_t = i \mid Y, \theta) \tag{2.17}$$

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j \mid Y, \theta)$$

Ймовірність переходу зі стану i у стан j в момент часу t, враховуючи спостереження.

β (зворотна ймовірність) - це схожий алгоритм до прямої імовірності, але у зворотному напрямку (ймовірність побачити всі майбутні спостереження, задані станом i в момент часу t). Це можна виразити рекурсивно, подібно до α, але у зворотному напрямку (тобто зворотний алгоритм) (2.18).

$$\beta_i(T) = 1 \tag{2.18}$$

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_j(t+1)$$

$$P(Y \mid \lambda) = \sum_{j=1}^{N} \pi_j b_j(y_1)\beta_j(1)$$

Щоб вивчити модель HMM, потрібно знати, в яких станах знаходиться система, щоб найкраще пояснити спостереження. Це буде ймовірність зайнятого стану γ - ймовірність стану i в момент часу t, враховуючи всі спостереження.

Враховуючи фіксовані параметри моделі HMM, можна застосувати прямий та зворотний алгоритм для обчислення α та β на основі спостережень. γ можна обчислити, просто помноживши α на β, а потім перенормувавши добуток (2.19).

$$\begin{aligned} \gamma_i(t) = P(X_t = i \mid Y, \theta) &= \frac{P(X_t=i,Y|\theta)}{P(Y|\theta)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \end{aligned} \tag{2.19}$$

ξ - ймовірність переходу зі стану i в j через час t, враховуючи всі спостереження. Це можна обчислити за допомогою α та β аналогічним чином (2.20).

$$\begin{aligned} &\gamma_i(t) = P(X_t = i \mid Y, \theta) \\ &\xi_{ij}(t) = P(X_t = i, X_{t+1} = j \mid Y, \theta) \end{aligned} \tag{2.20}$$

Після уточнення розподілу γ та ξ (θ_2) можна виконати точкову оцінку того, які ймовірності переходу та емісії будуть найкращими(θ_1: a, b) (2.21).

$$\begin{aligned} a_{ij}^* &= \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \\ b_i^*(v_k) &= \frac{\sum_{t=1}^{T} 1_{y_t=v_k}\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \end{aligned} \tag{2.21}$$

Далі відбувається фіксація одного набору параметрів, щоб покращити інші, і продовжується ітерація, поки рішення не сходиться (Рис.2.15).

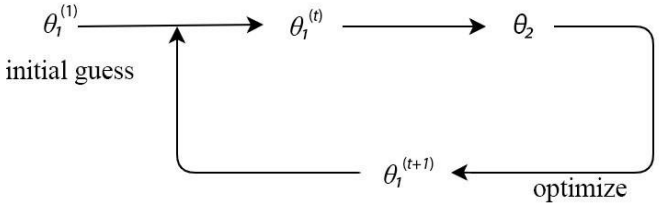


Рисунок 2.15 – Ітеративний процес оптимізації [51]

Отже, враховуючи всі спостереження в навчальних даних, алгоритм Баума – Велча може навчити модель HMM. Однак слід пам’ятати, що слід дотримуватися відкритості. У розпізнаванні мовлення проблема набагато складніша, і багато рішень іноді не вдається масштабувати.

2.2.2 Нейронні мережі та глибоке навчання для розпізнавання мовлення

Для задачі розпізнавання мовлення на кожному етапі використовуються різні статистичні моделі.

- Раніше мовними моделями, як правило, були N-грамові моделі, які дуже добре працювали для простих проблем з обмеженими вхідними даними мовлення. Вони, по суті, є таблицями, що описують ймовірності послідовностей лексем.
- Моделі вимови являли собою прості пошукові таблиці з ймовірностями, пов'язаними з вимовою. Ці таблиці могли бути дуже великими таблицями з різними типами вимов.
- Акустичні моделі будуються з використанням Гаусових змішаних моделей з дуже специфічними архітектурами, пов'язаними з ними.
- Обробка мовлення була попередньо визначена.

Після того, як такі типи моделей побудовано, можна виконати розпізнавання, роблячи висновок про отримані дані. Отже, після отримання форми хвилі, необхідно обчислити для неї характеристики (X) і виконати пошук Y, що дає найбільші ймовірності X.

З часом дослідники почали помічати, що кожен із цих компонентів може працювати ефективніше, якщо використати нейронні мережі (Рис.2.16).

- Замість N-грамових моделей можна будувати нейронні мовні моделі та подавати їх у систему розпізнавання мови, щоб відновити особливості, які були створені системою розпізнавання мови першого шляху.

- Розглядаючи моделі вимови, можна з’ясувати, як робити вимову для нової послідовності символів, яких ніколи не було раніше за допомогою нейронної мережі.
- Для акустичних моделей можна будувати глибокі нейронні мережі (такі як моделі на основі LSTM), щоб отримати набагато кращі показники точності класифікації характеристик для поточного звукового сигналу.
- Цікаво, що навіть етапи попередньої обробки мовлення були замінені згортковими нейронними мережами на необроблених мовних сигналах.

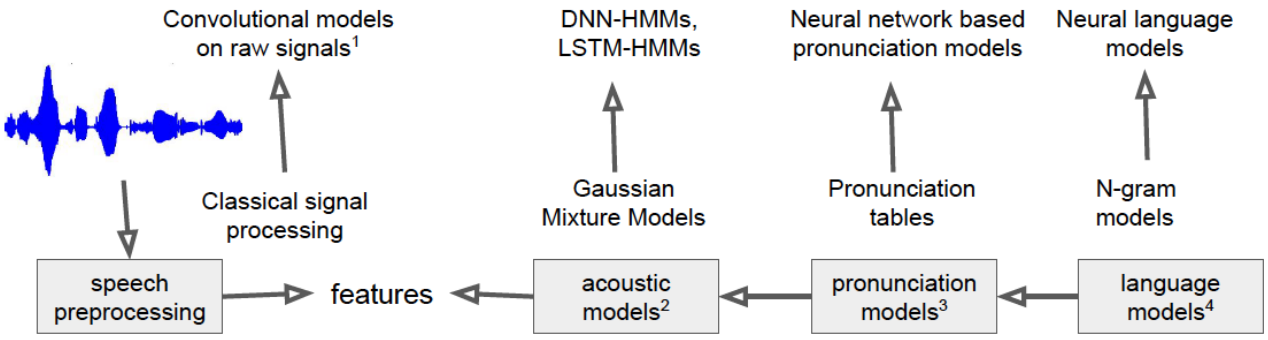


Рисунок 2.16 – Використання нейронних мереж на різних етапах розпізнавання мовлення [53]

Однак деякі нюанси залишилися. У кожному компоненті є нейронні мережі, але вони навчаються самостійно з різними цілями. Через це помилки одного компонента можуть не поводитися належним чином з помилками іншого компонента. Отже, це основна мотивація для розробки процесу, коли є можливість навчити всю модель як одну велику систему.

Ці так звані наскрізні моделі охоплюють все більше і більше компонентів, розглянутих вище. 2 найпопулярніші з них (1) Connectionist Temporal Classification (CTC), яка сьогодні широко використовується в Baidu та Google, але для цього необхідний довгий процес тренування; та (2) Послідовність до послідовності (Seq-2-Seq), яка не вимагає ручного налаштування.

Основна мотивація полягає в тому, що кінцева мета - робити наскрізне розпізнавання мови. Дано звук X - це послідовність сигналів від x1 до xT, і відповідний вихідний текст Y - який є послідовністю від y1 до yL. Y - це просто текстова послідовність (розшифровка), а X - оброблена аудіо спектрограма. Необхідно виконати розпізнавання мови шляхом вивчення імовірнісної моделі p (Y | X): починаючи з даних і прогнозуючи самі цільові послідовності.

Перша з цих моделей називається Connectionist Temporal Classification (CTC) ([1], [2], [3]). X - послідовність кадрів даних довжиною T: x1, x2,…, xT, а Y - вихідні маркери довжини L: y1, y2,…, yL. Через спосіб побудови моделі потрібно, щоб T було більше L (Рис.2.17).

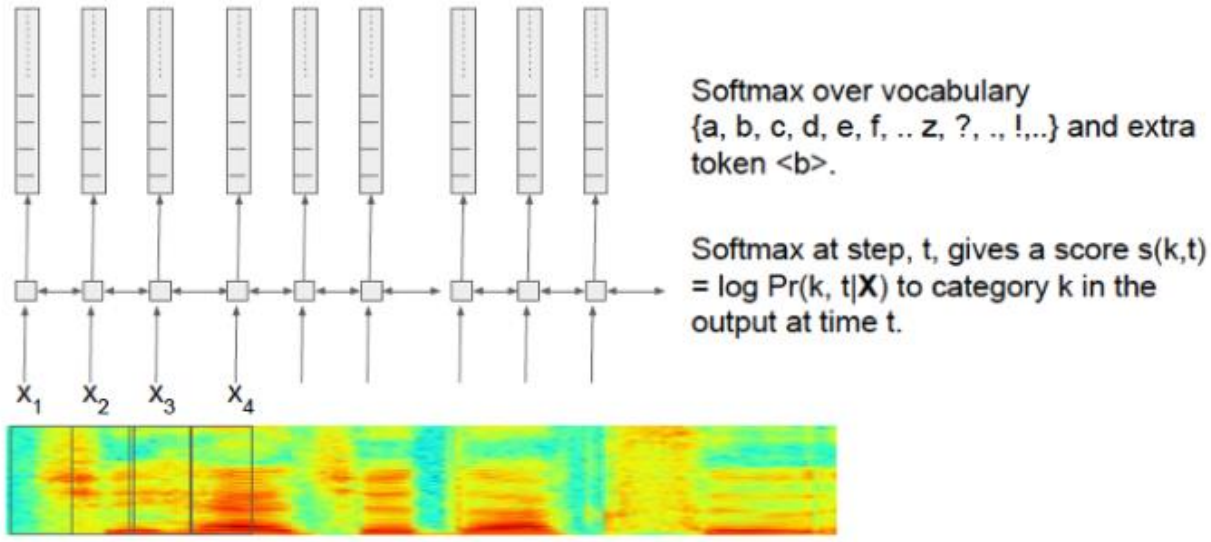


Рисунок 2.17 - Connectionist Temporal Classification модель [53]

Ця модель має дуже специфічну структуру, яка робить її придатною для мовлення:

- На виході отримується спектрограма звукового сигналу внизу (X). Необхідно подати його у двонаправлену рекурентну нейронну мережу, і в результаті стрілка, що вказує на будь-якому кроці, залежить від цілісності вхідних даних. Таким чином, він може обчислити досить складну функцію всіх даних X.

- Ця модель у верхній частині має функції softmax на кожному часовому інтервалі, що відповідає вводу. Функція softmax застосовується до словника з певною довжиною. У цьому випадку є малі літери від a до z та деякі розділові символи. Тож словниковий запас для CTC був би цим самим, а додатковий маркер називався пустим маркером.
- Кожен кадр прогнозування в основному створює вірогідність журналу для іншого класу маркерів на тому етапі часу. У наведеному вище випадку оцінка s (k, t) є часовою вірогідністю категорії k на часовому кроці t, враховуючи дані X.

У моделі CTC, якщо розглянути лише функції softmax, які генеруються повторюваною нейронною мережею протягом усього кроку часу, можна з часом знайти ймовірність розшифровки за допомогою цих окремих функцій softmax.

Модель CTC може представляти всі ці шляхи через весь простір функцій softmax і розглядати лише символи, що відповідають кожному з часових кроків. Модель CTC пройде через 2 символи C, потім через порожній символ, потім створить символи 2 A, потім створить інший порожній символ, потім перейде до символу T і, нарешті, знову створить порожній символ (Рис. 2.18).

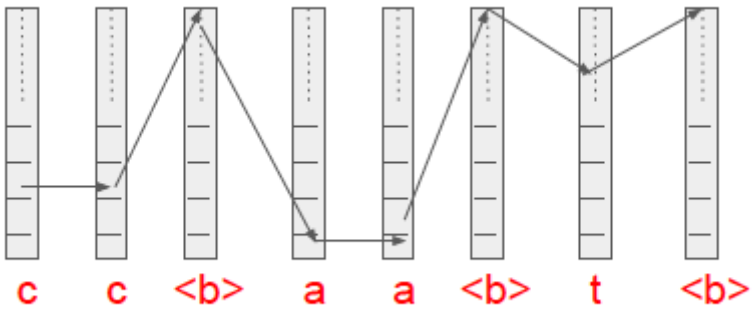


Рисунок 2.18 – приклад побудови шляхів у моделі CTC [53]

Отже, при проходженні цих шляхів з обмеженнями, можна лише переходити від однієї і тієї ж фонеми одного кроку до наступного. Тому на виході різні способи представлення вихідної послідовності.

У наведеному вище прикладі наведено послідовність cc <b> aa <b> t <b> або cc <b> <b> a <b> t <b> або cccc <b> aaaa <b> tttt <b>. З огляду на ці обмеження, виявляється, що, хоча існує експоненціальна кількість шляхів, за допомогою яких можна створити той самий вихідний символ, можна насправді зробити це правильно, використовуючи алгоритм динамічного програмування. Завдяки динамічному програмуванню можна точно обчислити і вірогідність журналу p (Y | X), і його градієнт. Цей градієнт може бути розповсюджений до нейронної мережі, параметри якої потім можна регулювати оптимізатором.

Нижче на рисунку 2.19 наведено деякі результати для CTC, які показують, як модель функціонує на заданому аудіо. Невихідна форма сигналу вирівнюється внизу, а відповідні прогнози виводяться вгорі. На початку він утворює символ H. У певний момент вона отримує дуже високу ймовірність, а це означає, що модель впевнена, що чує звук, відповідний H.

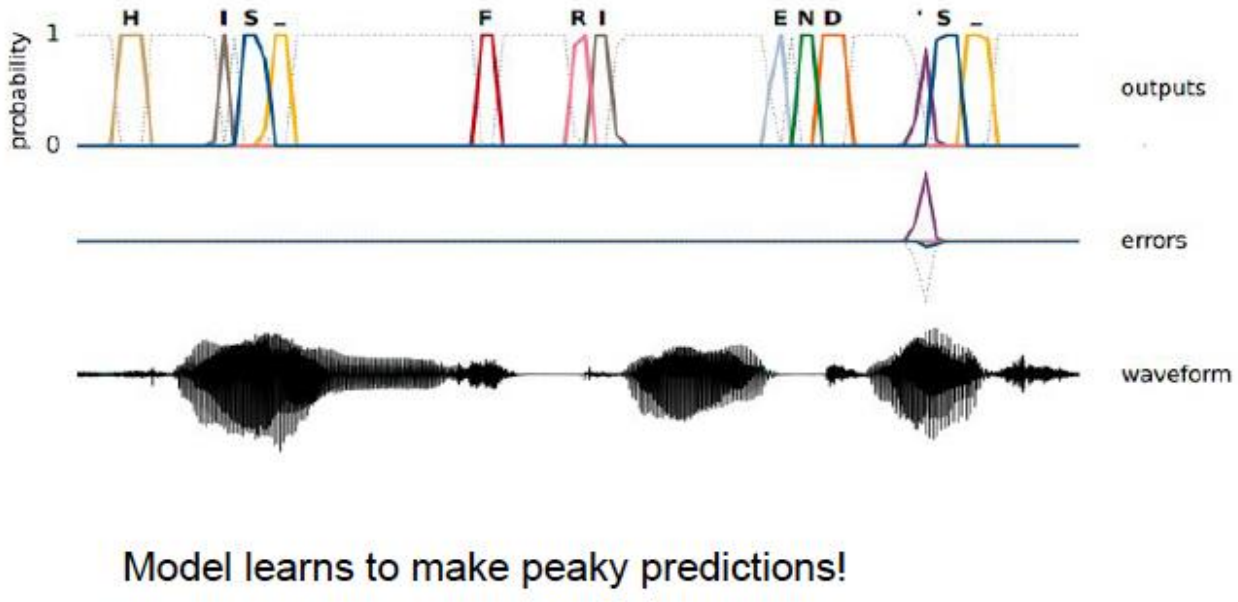


Рисунок 2.19 – Розпізнавання звукового сигналу «H» [53]

Однак у мовних моделей CTC є деякі недоліки. Вони часто неправильно пишуть слова і з’являються граматичні помилки. Отже, існує спосіб зрозуміти, як ранжувати різні шляхи, створені з моделі, і перекладати їх лише за мовною моделлю, результати повинні бути набагато кращими.

Google насправді вирішив ці проблеми, інтегрувавши мовну модель як частину самої моделі CTC під час навчання. Це така виробнича модель, яку зараз застосовує компанія.

Альтернативним підходом до обробки мовлення є модель sequence-to-sequence, яка робить прогнози наступного кроку. Дано деякі дані X, що потрібно ввести деякі символи від y1 до y {i}. Модель передбачає ймовірність наступного символу y {i + 1}. Мета тут полягає в тому, щоб в основному вивчити дуже хорошу модель для визначення імовірності.

З архітектурою моделі, як на рисунку 2.20, є нейронна мережа (яка є декодером у моделі sequence-to-sequence), яка переглядає весь вхід. Він подає символи шляху, які створюються як повторювана нейронна мережа, а потім передбачається наступний маркер як вихідний результат.

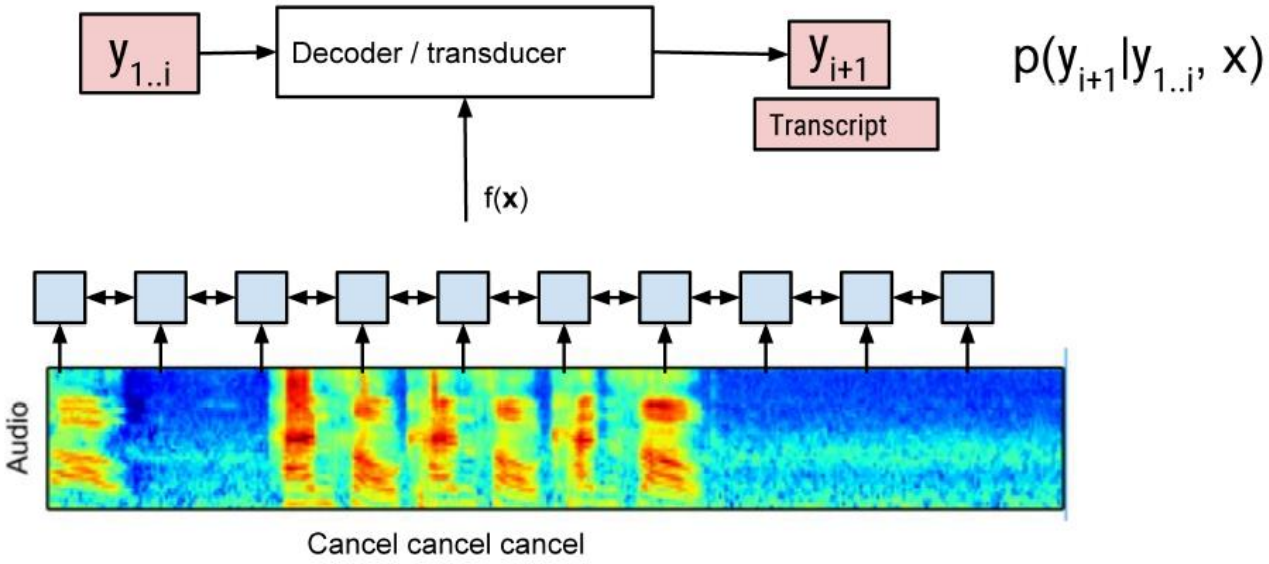


Рисунок 2.20 – Нейронна мережа, декодер з моделлю sequence-to-sequence [53]

Отже, ця модель розпізнає мовлення в рамках sequence-to-sequence. У перекладі X буде мовою-джерелом. У мовленнєвій області X був би величезною послідовністю звуку, кодованого тепер повторюваною нейронною мережею.

Для його функціонування потрібна здатність дивитись на різні частини часового простору, оскільки введення справді довге. Інтуїтивно результати перекладу погіршуються, оскільки вихідне речення стає довшим. Це тому, що

моделі справді важко шукати в потрібному місці. Виявляється, ця проблема посилюється набагато сильніше з довшими аудіо потоками. Тому потрібно застосувати механізм уваги, щоб ця модель взагалі працювала.

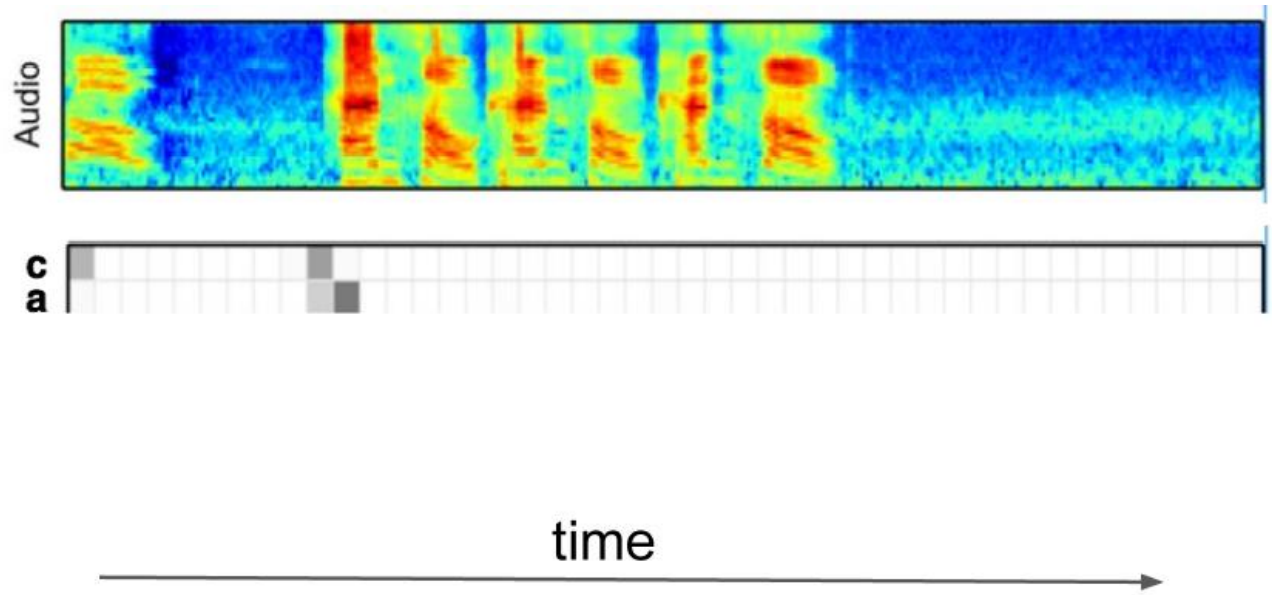


Рисунок 2.21 – Застосування механізму уваги в seq-2-seq [53]

Як видно з прикладу на рисунку 2.21, модель намагається створити 1-й символ C. Створюється вектор уваги, який, по суті, розглядає різні частини кроків часу введення і створює наступний розділ (а це A) після зміни уваги.

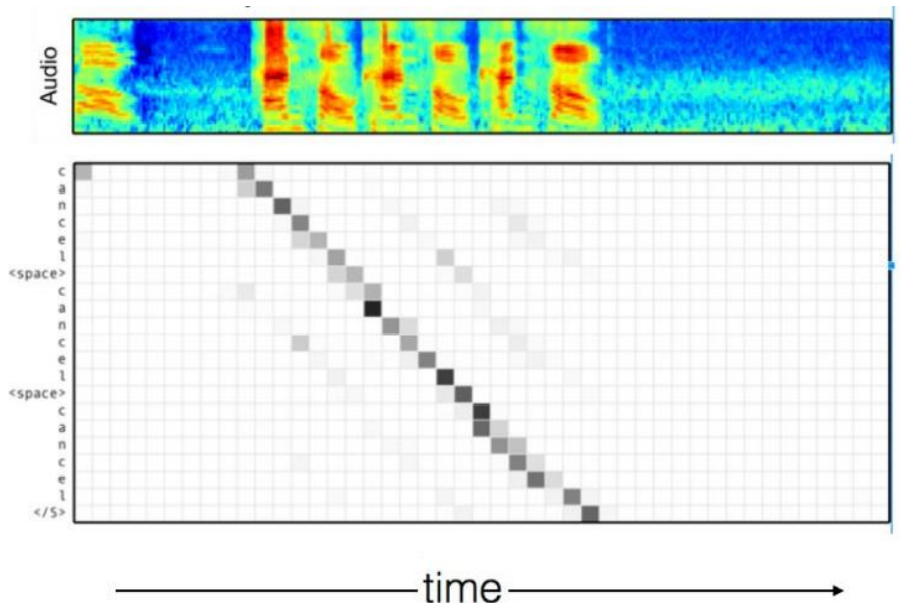


Рисунок 2.22 – Завершення розпізнавання з механізмом уваги в seq-2-seq [53]

Якщо продовжити робити це по всьому вхідному потоку, увага, яку лише тільки но засвоїла сама модель, просунеться. Тут видно, що він виробляє вихідну послідовність «cancel, cancel, cancel» (Рис.2.22).

Модель Listen, Attend і Spell [54] - це канонічна модель для категорії seq-2-seq (Рис. 2.23).

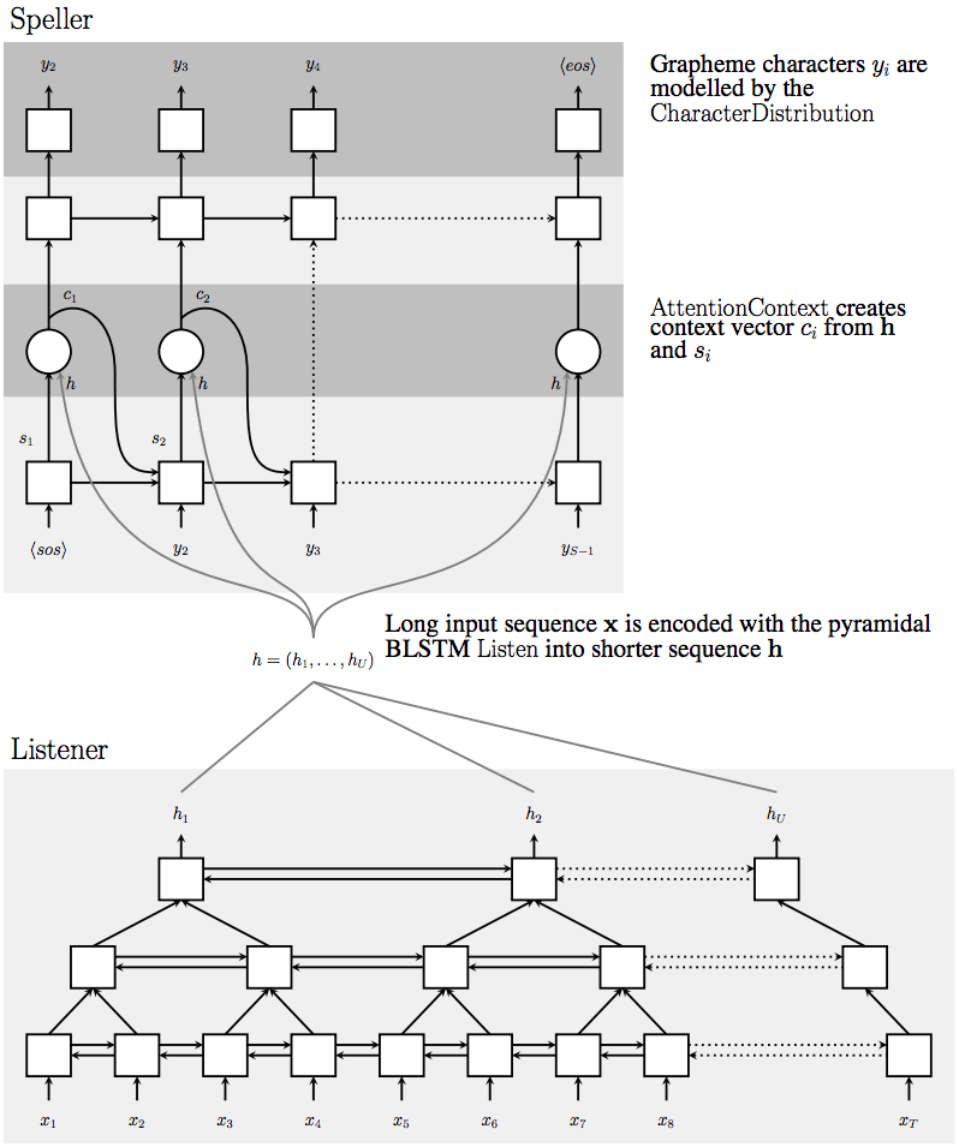


Рисунок 2.23 - Модель Listen, Attend і Spell [54]

- В архітектурі Listener є структура кодера. Для кожного часового кроку введення він створює векторне подання, яке кодує вхідні дані і представляється як h_t на часовому кроці t.

- В архітектурі Speller є архітектура декодера. Наступний символ c_t генерується на кожному часовому кроці t.
- Модель LAS використовує ієрархічний кодер для заміни традиційної рекурентної нейронної мережі. Замість того, щоб обробляти один кадр для кожного часового кроку, він згортає сусідні кадри під час подачі в наступний шар. Це зменшує кількість тимчасових кроків для обробки, тим самим пришвидшуючи обробку.

Обмеження моделі sequence-to-sequence:

- Одне з великих обмежень, що перешкоджає його використанню в Інтернет-системі, полягає в тому, що вироблений результат отримується з усього сигналу на вході. Це означає, що якщо помістити модель у реальну систему розпізнавання мови, доведеться спочатку дочекатися отримання всього звуку, перш ніж виводити символ.
- Іншим обмеженням є те, що сама модель уваги є обчислювальним вузьким місцем, оскільки кожен вихідний маркер приділяє увагу кожному кроку часу введення. Це ускладнює і сповільнює навчання моделі.
- Далі, коли вхідні дані приймаються і стають довшими, рівень помилок слова зменшується.

Онлайн-моделі sequence-to-sequence розроблені для подолання обмежень моделей sequence-to-sequence – не потрібно чекати, поки надійде вся вхідна послідовність, а також зникає необхідність використання самої моделі уваги протягом усієї послідовності. По суті, намір полягає у створенні результатів по мірі надходження вхідного сигналу.

Найбільш помітною онлайн моделлю seq-2-seq називається Neural Transducer [55]. Якщо взяти сигнал, який надходить, і раз у раз із регулярним інтервалом, можна запустити модель seq-2-seq на те, що було отримано в останньому блоці. Як видно з архітектури на рисунку 2.24, увага кодера (замість

того, щоб дивитись на весь вхід) буде зосереджена лише на невеликому блоці. Перетворювач видасть вихідні символи.

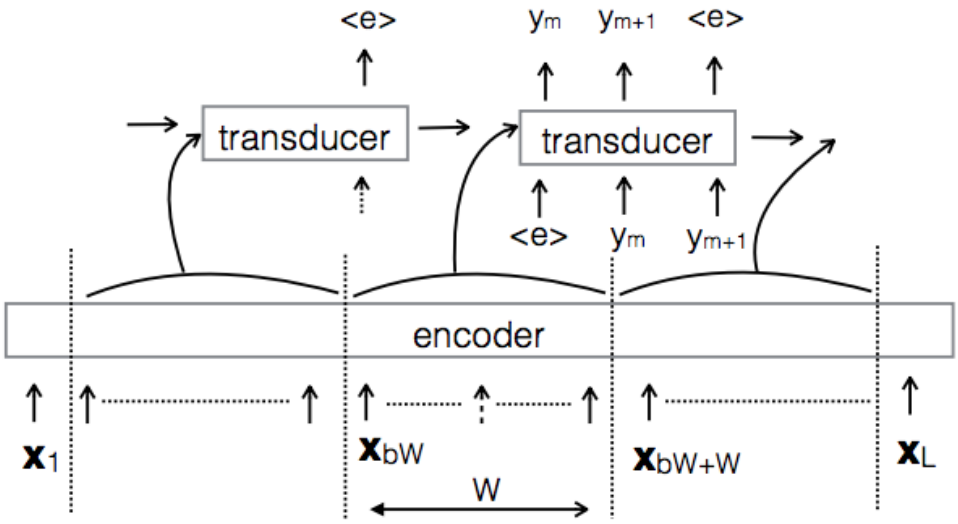


Рисунок 2.24 – Модель Neural Transducer [54]

Щоб покращити цю модель, необхідно використати згорткові нейронні мережі, запозичені з комп'ютерного зору. У роботі [56] використовуються згорткові мережі CNN, щоб виконувати сторону кодування в архітектурі мовлення.

Існує модель для піраміди і замість того, щоб будувати піраміду, просто складаючи дві речі разом, можна покласти нейронну архітектуру зверху, при виконанні укладання (Рис.2.25).

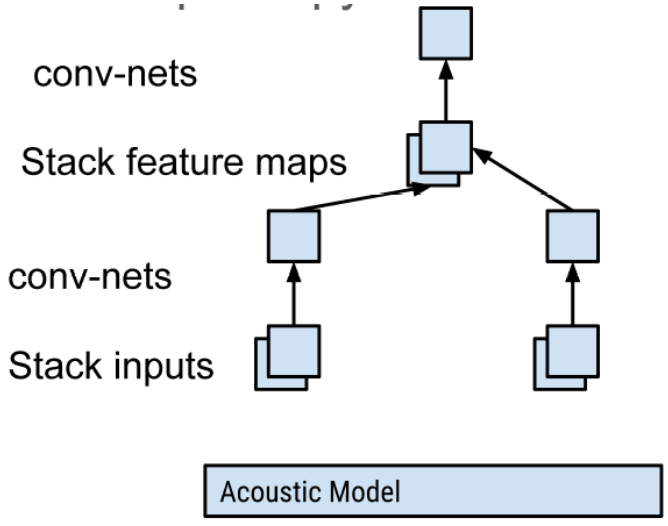


Рисунок 2.25 - Архітектура з використанням згорткової нейронної мережі [53]

Більш конкретно можна скласти їх як карти функцій і поставити CNN зверху. Що стосується проблеми розпізнавання мови, смуги частот і позначки часових функцій будуть відповідати природній підструктурі вхідних даних. Згорткова архітектура по суті розглядає цю підструктуру.

2.3 Взаємодія засобу IoT з командами додатку

Після отримання транскрипції людського мовлення необхідно перетворити текст в команду, яку б зрозумів IoT пристрій.

Розпізнавання намірів (Intent detection) - іноді його називають класифікацією намірів - це завдання взяти письмове чи усне введення та класифікувати його на основі того, чого користувач хоче досягти. Розпізнавання намірів є важливою складовою чат-ботів і знаходить застосування в конверсіях продажів, підтримці клієнтів та багатьох інших сферах.

Розпізнавання намірів - це форма обробки природної мови (NLP), підвид завдання з використанням штучного інтелекту. NLP займається обробкою та аналізом природної мови, тобто будь-якої мови, яка розвинулася природним, а не штучним шляхом.

Розуміння природної мови (NLU) - це підполе в обробці природних мов (NLP), яке зосереджується на організації неструктурованого вводу користувача таким чином, щоб комп’ютер міг його зрозуміти та проаналізувати [57]. Цей процес включає:

1. Синтаксичний аналіз: виявлення основних граматичних правил, організації слів, поєднання та відношення одне до одного. Це складається з [58]:
    - розбиття тексту на менші сегменти (слова, короткі речення), які називаються "токени".
    - Позначення лексем як іменника, дієслова, прикметника тощо. Цей крок називається «Part of Speech tagging (PoS)».
    - Скорочення слів до їх коренів для кращого аналізу.
    - Фільтрування заповнюючих слів, щоб заощадити простір та час при обробці великих даних.

2. Семантичний аналіз: висновок про значення вхідного речення за допомогою [58]:
    - Розрізнення контексту кожного слова.
    - Розуміння зв’язку між словами в тексті.

Моделі NLU використовують:

- Контрольоване машинне навчання для кроків аналізу синтаксису (токенізація, тегування PoS), таких як векторні машини підтримки (SVM), байєсівські мережі та алгоритми максимальної ентропії.
- Машинне навчання без нагляду для семантичного аналізу, такого як алгоритми кластеризації.

Класифікатори намірів проходять навчання на відповідних маркованих наборах даних, отже, це навчальна програма з вчителем. Класифікатори використовують:

- Відповідність шаблону на основі правил
- Алгоритми класифікації машинного навчання, такі як дерева рішень, наївний Байєс та логістична регресія.
- Глибоке навчання та штучні нейронні мережі.

Класифікатор намірів використовується для узгодження результатів процесу NLU з відповідними заздалегідь визначеними мітками в наборі даних навчальної програми. Наприклад, коли користувач каже голосовому помічнику: «Я хочу забронювати рейс з Києва до Харкова», класифікатор намірів класифікує контекст і послідовність слів під міткою «book_flight».

Таким чином встановлюються і інші відповідності текстових транскрипцій до машинних команд.

Висновки до другого розділу

Було досліджено архітектуру мережі інтернет речей та її складові. Для збору даних та управління пристроями IoT використовуються датчики та виконавчі механізми. Класична архітектура мережі включає в собі такі рівні:

фізичний, транспортний, моніторингу, попередньої обробки, зберігання та безпеки.

Також було розглянуто архітектуру системи розпізнавання мовлення. Така система містить модуль введення голосового сигналу (фронтенд) та забезпечує попередню обробку. Після цього інформація потрапляє до декодера, який складається із словника, мовної моделі та акустичної моделі.

Було проаналізовано та описано алгоритми, які використовуються в сучасних системах розпізнавання мовлення. Два основних: статистичний алгоритм прихованих моделей Маркова та алгоритми машинного навчання. Для побудови мовних моделей використовують рекурентні нейронні мережі, а для акустичних моделей використовують згорткові нейронні мережі. Для перетворення отриманого текстового повідомлення в машинні команди використовуються методи NLP (Natural Language Processing).

Використання алгоритмів машинного навчання збільшило точність розпізнавання біглого мовлення, але в свою чергу їх використання поступається статистичним алгоритмам у використанні обчислювальних ресурсів.

РОЗДІЛ 3
АНАЛІЗ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СТВОРЕННЯ ДОДАТКУ ГОЛОСОВОГО УПРАВЛІННЯ ЗАСОБОМ IOT

3.1 Аналіз потоків даних у IoT системах з голосовим управлінням

Перш ніж досліджувати потоки даних у IoT системах з голосовим управлінням, необхідно розглянути базову архітектуру та складові класичної IoT системи.

Хоча кожна система IoT відрізняється, основа для кожної архітектури Інтернету речей, а також загального потоку процесів обробки даних приблизно однакова. Перш за все, він складається з речей, які є об'єктами, підключеними до Інтернету, які за допомогою вбудованих датчиків та виконавчих механізмів здатні відчувати навколишнє середовище та збирати інформацію, яка потім передається шлюзам IoT. Наступний етап складається із систем збору даних IoT та шлюзів, які збирають величезну масу необроблених даних, перетворюють їх у цифрові потоки, фільтрують та попередньо обробляють, щоб вони були готові до аналізу. Третій рівень представлений крайовими пристроями, відповідальними за подальшу обробку та посилений аналіз даних. На цьому рівні також можуть втрутитися технології візуалізації та машинного навчання. Після цього дані передаються до центрів обробки даних, які можуть бути засновані на хмарі або встановлені локально. Тут дані зберігаються, керуються та аналізуються для поглибленого аналізу [59].

Основні чотири шари архітектури IoT зображено на рисунку 3.1:

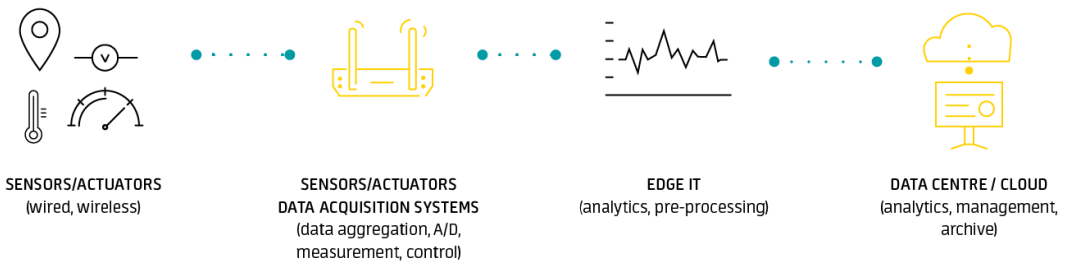


Рисунок 3.1 – Основні шари архітектури IoT [59]

Як основа для кожної системи IoT, підключені пристрої відповідають за надання Інтернетом речей наборами даних. Для збору інформації з навколишнього або самих пристроїв IoT для них необхідні датчики. Вони можуть бути вбудовані в самі прилади або реалізовані як зовнішні пристрої для вимірювання та збору даних. Наприклад, охоронні системи будинку, які за допомогою вбудованих сенсорів збирають інформацію про рухи об’єктів, інфрачервоні датчики, що реагують на тепло.

Ще одним незамінним елементом цього шару є виконавчі механізми. У тісній співпраці з датчиками вони можуть перетворювати дані, генеровані розумними об’єктами, у фізичну дію. Прикладом можуть бути все ті ж охоронні системи будинків. Як тільки в полі зору інфрачервоних сенсорів з’являється жива істота, інформація передається до камер відеоспостереження та виконавчі механізми вмикають відеозапис. У разі спрацьовування датчиків руху в помешканні виконавчі механізми передають про це інформацію до охоронних підприємств.

Важливим є також те, що підключені об'єкти повинні мати можливість не тільки двосторонньо взаємодіяти з відповідними шлюзами або системами збору даних, але також мати можливість розпізнавати та комунікувати між собою для збору та обміну інформацією та співпраці в реальному часі, щоб використовувати значення всієї розгорнутої системи. Особливо у випадку обмежених ресурсів та пристроїв, що працюють від батареї, досягнення цього не є простим завданням, оскільки такий зв’язок вимагає великої обчислювальної потужності та споживає дорогоцінну енергію та пропускну здатність. Таким чином, надійна архітектура може забезпечити ефективне управління пристроями лише тоді, коли вона використовує спеціальні, безпечні та полегшені протоколи зв'язку [59].

Шлюзи та системи збору даних IoT.

Незважаючи на те, що цей рівень все ще функціонує в безпосередній близькості від датчиків та виконавчих механізмів на певних пристроях, важливо описати його як окремий етап архітектури IoT, оскільки він має вирішальне

значення для процесів збору, фільтрації та передачі даних до інфраструктури крайових туманних та хмарних платформ. З огляду на величезний обсяг введення та виведення даних, який може створити розгортання мільйонних пристроїв, можливості збору, відбору та транспортування даних повинні бути в центрі уваги [59]. Як посередники між IoT пристроями та хмарою та аналітикою, шлюзи та системи збору даних забезпечують необхідну точку зв'язку, яка пов'язує решту шарів (Рис. 3.2).

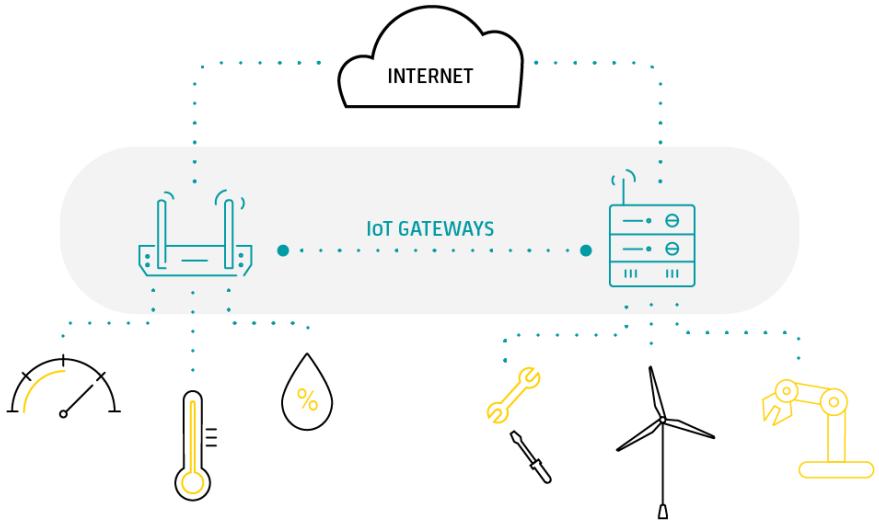


Рисунок 3.2 – Шлюзи та системи збору даних IoT [59]

Шлюзи полегшують зв'язок між датчиками та рештою системи, перетворюючи дані датчиків у формати, які легко передаються та використовуються для інших компонентів системи. Більше того, вони можуть контролювати, фільтрувати та відбирати дані, щоб мінімізувати обсяг інформації, яку потрібно пересилати в хмару, що позитивно впливає на витрати на передачу по мережі та час відгуку. Таким чином, шлюзи забезпечують місце для локальної попередньої обробки даних датчиків, які стискаються в корисні пакети, готові до подальшої обробки.

Ще одним аспектом, який підтримують шлюзи, є безпека. Оскільки шлюзи відповідають за управління потоком інформації в обох напрямках, за допомогою належних засобів шифрування та захисту вони можуть запобігти витоку даних

хмарних даних IoT, а також зменшити ризик зловмисних зовнішніх атак на пристрої IoT.

Крайові обчислення.

Не дивлячись на те, що такі системи є не в кожній IoT екосистемі, крайові пристрої можуть принести значні переваги, особливо великим проектам IoT. В умовах обмеженої доступності та швидкості передачі даних до хмарних платформ IoT, крайові системи можуть забезпечити швидший час відгуку та більшу гнучкість у обробці та аналізі даних IoT. Оскільки швидкість аналізу даних є ключовою у деяких програмах Industrial Internet of Things, нещодавно крайові обчислювальні технології різко зросли в популярності серед екосистем Industrial Internet of Things (Рис. 3.3).

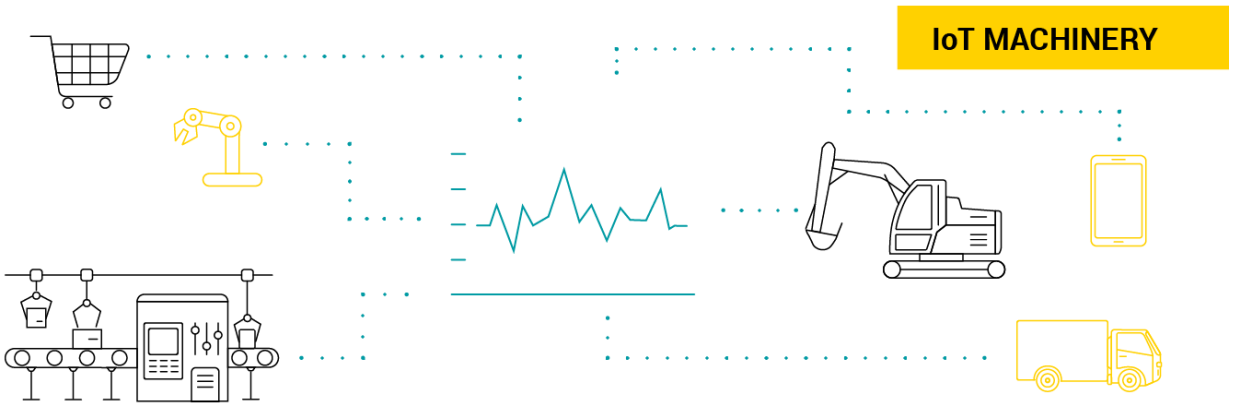


Рисунок 3.3 – Крайові обчислення в Industrial IoT [59]

Оскільки крайова інфраструктура може бути розташована ближче до джерела даних у фізичному відношенні, їй простіше і швидше діяти на вихідні дані IoT в режимі реального часу та забезпечувати висновок у формі миттєвого діючого інтелекту. У цьому сценарії туди пересилаються лише більші фрагменти даних, яким дійсно потрібна потужність Хмари. Мінімізуючи вплив мережі, безпеку можна значно підвищити, тоді як зменшення споживання енергії та пропускної здатності сприяє більш ефективному використанню бізнес-ресурсів.

Дата центри та хмарні обчислення.

Якщо датчики є нейронами, а шлюз є основою IoT, то хмара - це основний шар обробки даних Інтернету речей. На відміну від крайових рішень, дата центр або хмарні технології призначені для зберігання, обробки та аналізу великих обсягів даних для глибшого розуміння за допомогою потужних механізмів аналізу даних та механізмів машинного навчання, які крайові системи ніколи не зможуть підтримати.

Хмарні обчислення сприяють підвищенню рівня виробництва, скороченню кількості обчислювальних простоїв та зменшення споживання електроенергії енергії.

На хмарних середовищах зазвичай і розташовані системи автоматичного розпізнавання мовлення для приладів IoT. Так, у «хмарі» опрацьовують дані найпопулярніші голосові помічники Alexa, Google Assistant, Siri, Yandex Аліса та інші. Завдяки цьому розробники зуміли досягти високої точності розпізнавання, яка на даний момент не доступна будь-яким рішенням для автоматичного розпізнавання мовлення з відкритим кодом, таких як CMUSphinx, Kaldi, DeepSearch тощо.

Але у такої архітектури є певні недоліки. Перш за все постає питання безпеки у приладах, що підключені до всесвітньої мережі Інтернет. Навіть добре налаштований шлюз не може гарантувати 100% безпеку. Так до системи можуть втрутитися зловмисники для управління засобами IoT всередині системи та перехоплення даних, саботажу тощо.

По-друге, при використанні сторонніх хмарних сервісів для розпізнавання мовлення від таких компаній, як Google, Amazon, Yandex, піднімає питання конфіденційності користувацьких даних. Так, для навчання власних алгоритмів вищеназвані корпорації збирають та зберігають користувацькі дані на хмарних серверах. Таким чином неможливо захиститися від імовірного витоку конфіденційної інформації.

Для деяких бізнесів та компаній описані проблеми можуть бути фатальними. Тож для вирішення завдання інтеграції IoT систем для таких сфер пропонується розглянути закриту «офлайн» архітектуру засобів IoT.

У такої системи залишаються основні складові класичної IoT архітектури:

1. IoT прилади за допомогою сенсорів збирають інформацію з навколишнього світу.
2. Ця інформація за допомогою шлюзів може відправлятися на виділені сервери, всередині закритої мережі для обробки та зберігання.
3. За допомогою внутрішньої мережі та шлюзів оброблені дані повертаються на IoT прилад та активують виконавчий механізм для зміни фізичного стану.

Деякі вузькоспеціалізовані IoT пристрої можуть виконувати обчислення на самому приладі. А шлюзи, в свою чергу, використовувати тільки для комунікації та передачі даних.

Базовий принцип роботи таких пристроїв з використанням голосового управління зображено на рисунку 3.4.

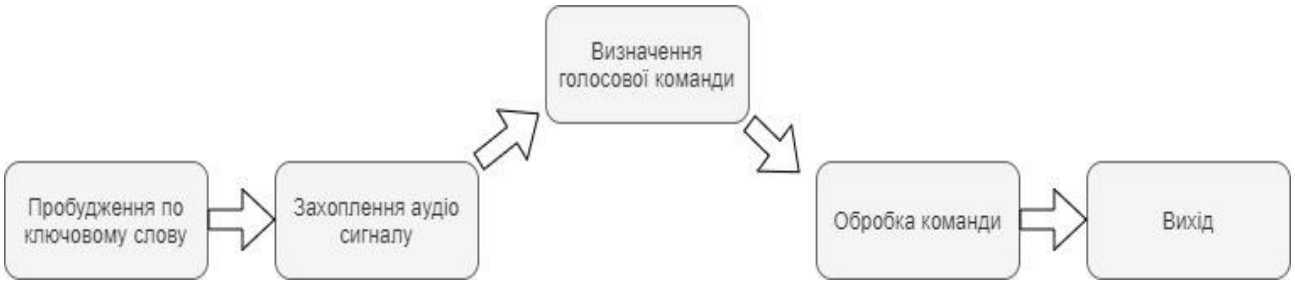


Рисунок 3.4 – Принцип роботи IoT пристрою з голосовим управлінням

Перше у ланці такої системи - це певна «система розпізнавання слова пробудження». Вона буде безперервно слухати звук за допомогою мікрофону, чекаючи фрази чи слова активації.

Почувши це слово, механізм розбудить решту системи та почне записувати звук, щоб зафіксувати будь-які вказівки користувача.

Після того, як аудіо було захоплено, воно відправить його на сервер для розпізнавання.

Сервер обробляє аудіо та опрацьовує те, про що просить користувач.

У деяких системах сервер може обробляти запит користувача, викликаючи інші служби, щоб виконати запит користувача.

Для реалізації функціоналу розпізнавання мовлення можна використати сторонні фреймворки.

CMU Sphinx наразі є найбільшим проектом по розпізнанню людського мовлення. У інструментарії, що йде наступних програмах та бібліотеках [60]:

- Pocketsphinx - невелика програма, яка приймає на вхід довільні акустичні моделі, граматики і словники, а також звуковий потік (або звуковий файл, або сам бере потік з мікрофона). На виході виходить розпізнаний текст. Написана на C, працює швидко. Може бути встановлена на мікрокомп'ютери по типу Raspberry Pi.
- Sphinxbase - бібліотека необхідна для роботи Pocketsphinx.
- Sphinx4 - гнучка бібліотека для розпізнавання, написана на Java. В четвертій версії використано нові алгоритми навчання акустичних моделей.
- Sphinxtrain - програма для навчання акустичних моделей.

CMUSphinx реалізовує розпізнавання мовлення з використанням статистичних алгоритмів HMM та GMM. Система на даний момент містить велику базу підтримки. Система також містить найширшу базу вже натренованих акустичних моделей для багатьох мов.

CMUSphinx містить функціонал для налаштування слів для пробудження. Але основний недолік полягає в тому, що навіть найлегший Pocketsphinx для розпізнавання мовлення використовує значну кількість ресурсів центрального процесору та оперативної пам'яті. Тому використання на найменших мікроконтролерах неможливе.

Іншим популярним та найперспективнішим фреймворком для розпізнавання мовлення є Kaldi. Kaldi має найвищу точність розпізнавання, за швидкістю роботи незначно програючи конкурентам. Також система Kaldi здатна надати користувачеві найбільш багатий вибір алгоритмів для різних завдань і дуже зручна у використанні.

В основі роботи Kaldi використовуються такі алгоритми [61]:

- Для вилучення акустичних ознак з вхідного сигналу використовуються або широко відомі MFCC (Mel-Frequency Cepstral Coefficients), або трохи менш популярні PLP (Perceptual Linear prediction.
- Модуль акустичного моделювання включає в себе приховані Марківські моделі (HMM), модель суміші гаусівських розподілів (GMM), глибокі нейронні мережі, а саме Time-Delay Neural Networks (TDNN).
- Мовне моделювання здійснюється за допомогою кінцевого перетворювача, або FST (finite-state transducer). FST кодує відображення з вхідної послідовності символів в вихідну послідовність символів, при цьому для переходу існують ваги, які визначають ймовірність обчислення вхідного символу в вихідний.
- Декодування відбувається за допомогою алгоритму прямого-зворотного ходу.

Так, завдяки використанню глибокого навчання вдалося підвищити точність розпізнавання мовлення, але продуктивність за рахунок цього зменшилася порівняно із конкурентом CMUSphinx. Це робить бібліотеку також неможливою для використання на мікроконтролерах.

Таким чином, вузьким місцем наведеної архітектури є функціонал пробудження пристрою IoT за активаційним словом. Виникає потреба оптимізувати цей функціонал так, щоб девайс пробуджувався від активаційного слова. При цьому кількість помилкових реакцій необхідно мінімізувати та продуктивність алгоритму має бути такою, щоб можна було виконувати його на невеликих комп'ютерних схемах та мікроконтролерах.

3.2 Оптимізація підходів до проекту створення додатку голосового управління засобом IoT

Визначення пробуджуючих слів - це завдання розпізнавання мовлення для активації мовного помічника або IoT приладу, наприклад, «Hey, Alexa» для Amazon Echo. Враховуючи, що такі системи призначені для підтримки повністю

автоматичного розпізнавання мови та мають реалізовані алгоритми для цього, завдання здається простим. Однак це створює інший набір завдань, які необхідно вирішити при проектуванні такої системи, оскільки вони повинні бути завжди увімкненими та перманентно слухати команду пробудження, ефективними при використанні внутрішніх ресурсів, тим самим будучи енергоефективними та, головним чином, враховувати аспект конфіденційності.

Системи для розпізнавання мовлення з відкритим кодом, такі як CMUSphinx, Kaldi, DeepSearch та інші, мають вбудований функціонал виявлення слів пробудження. Вони мають таку ж ефективність, як і при автоматичному розпізнаванні біглого мовлення. Проте продуктивність використання цього функціоналу залишається низьким, тому що необхідно мати активною усю акустичну модель в оперативній пам’яті, і таке рішення не є ресурсно ефективним. Крім цього це накладає обмеження до використання простих мікроконтролерів при розробці IoT приладів. Так, для використання вищеназваних бібліотек необхідно, щоб мікрокомп’ютер достатньо потужний процесор та достатню кількість оперативної пам’яті. Та це впливає в свою чергу на розміри приладу, роблячи їх значно більшими та громіздкими.

Хоча основні голосові помічники, такі як Siri та Alexa, керуються фірмовими системами розпізнавання слів, існують такі відкриті набори інструментів, як Porcupine та Snowboy [62]. Такі екосистеми надають набір інструментів моделювання, та широкі можливості розгортання. На жаль, ці екосистеми залишаються закритими; вони використовують власні дані, моделі розпізнавання. Що стосується екосистем із відкритим кодом, Precise позбавлений деяких з цих мінусів, але його набори даних обмежені, а його основним пристроєм розгортання є лише Raspberry Pi.

Отже, необхідно оптимізувати процес пробудження IoT приладу за визначеною голосовою командою. Основними критеріями оцінки є висока ефективність реагування з низькою кількістю помилок, а також можливість дистрибуції на маленьких та непотужних мікроконтролерах.

Мікроконтролери - це, як правило, невеликі, малопотужні обчислювальні пристрої, вбудовані в апаратне забезпечення, яке вимагає базових обчислень. Додаючи машинне навчання до крихітних мікроконтролерів, можна додати інтелектуальні обчислення для мільярдів пристроїв, які люди використовують у своєму житті, включаючи побутову техніку та пристрої Інтернету речей, не покладаючись на дороге обладнання та надійні з'єднання з Інтернетом, що часто залежить від пропускної здатності та обмеження потужності та призводить до великої затримки. Це також може допомогти зберегти конфіденційність, оскільки дані не залишаються на пристрої. Тобто це такі розумні прилади, які можуть адаптуватися до повсякденного режиму користувача, інтелектуальні промислові датчики.

Для вирішення поставленої задачі запропоновано використовувати бібліотеку машинного навчання TensorFlow для створення акустичної моделі. Створений командою Google Brain, TensorFlow - це бібліотека з відкритим кодом для чисельних обчислень та масштабованого машинного навчання [63]. TensorFlow поєднує в собі безліч моделей машинного навчання та глибокого навчання (нейронних мереж) та алгоритмів. Він використовує Python для забезпечення зручного інтерфейсного API для побудови програм із фреймворком, виконуючи ці програми в високопродуктивному C ++. TensorFlow може навчати та запускати глибокі нейронні мережі для класифікації рукописних цифр, розпізнавання зображень, вбудовування слів, періодичних нейронних мереж, моделей sequence-to-sequence для машинного перекладу, обробки природної мови та моделювання на основі PDE (диференціальне рівняння).

TensorFlow - найпопулярніша бібліотека з усіх, оскільки вона створена для того, щоб бути доступною для всіх. Бібліотека Tensorflow включає різні API для побудови масштабних архітектур глибокого навчання, таких як CNN або RNN[64]. TensorFlow базується на обчисленні графіків; це дозволяє розробнику візуалізувати побудову нейронної мережі за допомогою Tensorboad. Цей інструмент корисний для налагодження програми. Нарешті, Tensorflow

створений для масштабного розгортання. Він може працювати на процесорі та графічному процесорі.

Назва TensorFlow безпосередньо походить від його основної структури: Tensor. У TensorFlow усі обчислення беруть участь у тензорах. Тензор - це вектор або матриця n-вимірів, що представляє всі типи даних. Усі значення в тензорі містять однаковий тип даних із відомою (або частково відомою) формою. Форма даних - це розмірність матриці або масиву.

Тензор може бути створений із вхідних даних або результату обчислення. У TensorFlow всі операції проводяться всередині графіка. Графік - це набір обчислень, що відбувається послідовно. Кожна операція називається операційним вузлом і пов'язана між собою.

Графік тензору окреслює операції та зв’язки між вузлами. Однак він не відображає значень. Крайні значення вузлів є тензором, тобто способом заповнення операції даними.

Нижче наведено список підтримуваних алгоритмів TensorFlow [64]:

- Лінійна регресія: tf.estimator.LinearRegressor
- Класифікація: tf.estimator.LinearClassifier
- Класифікація глибокого навчання: tf.estimator.DNNClassifier
- Глибоке навчання: tf.estimator.DNNLinearCombinedClassifier
- Підсилення деревної регресії: tf.estimator.BoostedTreesRegressor
- Класифікація за допомогою підсиленого дерева: tf.estimator.BoostedTreesClassifier

Оптимізація моделей TensorFlow для мікроконтролерів та мобільних пристроїв відбувається за допомогою використання бібліотеки TensorFlow Lite.

TensorFlow Lite для мікроконтролерів призначений для запуску моделей машинного навчання на мікроконтролерах та інших пристроях із лише кількома кілобайтами пам'яті. Основне середовище виконання просто вміщується в 16 Кб на Arm Cortex M3 і може працювати з багатьма базовими моделями. Він не вимагає підтримки операційної системи, будь-яких стандартних бібліотек C або C ++ або динамічного розподілу пам'яті.

Але при розробці необхідно враховувати наступні обмеження:

- Підтримка обмеженого підмножини операцій TensorFlow.
- Підтримка обмеженого набору пристроїв.
- API низького рівня C ++, що вимагає ручного управління пам’яттю.
- Навчання на пристрої не підтримується.

Але не дивлячись на дані обмеження можна сказати, що ця бібліотека цілком підходить для вирішення задачі реалізації акустичної моделі для пробудження пристрою IoT від голосової команди.

Перш за все необхідно знайти певний датасет аудіофайлів для навчання моделі. Для цього підходить тестовий датасет speech_commands [64], який пропонує сама бібліотека TensorFlow.

Speech_commands - звуковий набір вимовлених слів, призначений для навчання та оцінки систем виявлення ключових слів. Її основною метою є створення способу побудови та тестування невеликих моделей, які виявляють, коли промовляється одне слово, із набору з десяти цільових слів, з якомога меншою кількістю помилкових спрацьовувань від фонового шуму або незв’язного мовлення. Хоча в тестовому наборі сегменти мовчання - це звичайні файли на 1 секунду, в навчанні вони надаються як довгі сегменти в папці "background_noise". Необхідно розділити ці фонові шуми на кліпи в 1 секунду, а також зберегти один із файлів для набору валідації. Датасет містить достатню кількість слів для навчання моделі. Обсяг датасету складає 8.17ГБ [64] (Рис.3.5).

Split	Examples
'test'	4,890
'train'	85,511
'validation'	10,102

Рисунок 3.5 – Обсяг вибірки голосових сигналів TensorFlow

Слова, для яких створено аудіо сегменти та їх кількість зображено на рисунку 3.6.

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052

Рисунок 3.6 – Склад вибірки голосових сигналів TensorFlow.

Для реалізації поставленого завдання обрано цільове слово «Sheila».

Обравши наші навчальні дані, необхідно подумати про те, якими особливостями звукових сигналів потрібно тренувати нейронну мережу. Навряд чи подача необробленого звукового сигналу в нейронну мережу дасть хороші результати.

Популярний підхід до розпізнавання слів за допомогою нейронних мереж - перевести задачу на розпізнавання образів.

Необхідно перетворити звукові зразки на щось, що схоже на зображення - для цього можна використати спектрограму.

Щоб отримати спектрограму звукового сигналу, необхідно розбити сигнал на невеликі ділянки, а потім виконати дискретне перетворення Фур'є на кожному розділі. Це виділить частоти, які присутні в цьому фрагменті звуку.

Складання цих частотних зрізів дає спектрограму звукового сигналу. Функцію для отримання спектрограми для звукового сигналу зображено на рисунку 3.7.

```
def get_spectrogram(audio):
    # нормалізація звукового сигналу
    audio = audio - np.mean(audio)
    audio = audio / np.max(np.abs(audio))
    # створення спектрограми
    spectrogram = audio_ops.audio_spectrogram(audio,
                                              window_size=320,
                                              stride=160,
                                              magnitude_squared=True).numpy()
    # зменшимо кількість частотних вікон для збільшення чутливості спектрограми
    spectrogram = tf.nn.pool(
        input=tf.expand_dims(spectrogram, -1),
        window_shape=[1, 6],
        strides=[1, 6],
        pooling_type='AVG',
        padding='SAME')
    spectrogram = tf.squeeze(spectrogram, axis=0)
    spectrogram = np.log10(spectrogram + 1e-6)
    return spectrogram
```

Рисунок 3.7 – Функція перетворення звукового сигналу в спектрограму.

Ця функція спочатку нормалізує звуковий сигнал, щоб усунути будь-які розбіжності в обсязі наших шаблонів. Потім вона обчислює спектрограму - у спектрограмі є досить багато даних, тому для зменшення їх розбіжності застосовується середнє об’єднання.

Врешті-решт, необхідно логарифмувати спектрограми, щоб не подавати екстремальні значення в нейронну мережу, що може ускладнити тренування.

Перш ніж створювати спектрограму, потрібно додати випадковий шум і дисперсію до вибірки. Випадковим чином необхідно змістити аудіо вибірку на 1-секундний сегмент - це гарантує, що нейронна мережа генералізується навколо положення аудіо (Рис.3.8).

```
# змінюємо положення звуку у сигналі випадковим чином
voice_start, voice_end = get_voice_position(audio, NOISE_FLOOR)
end_gap=len(audio) - voice_end
random_offset = np.random.uniform(0, voice_start+end_gap)
audio = np.roll(audio,-random_offset+end_gap)
```

Рисунок 3.8 – Додавання випадкового шуму та зміщення вибірки

Після цього до випадкової вибірки додається фоновий шум. Це допомагає нейронній мережі опрацювати унікальні особливості цільового слова та ігнорувати фоновий шум (Рис.3.9).

```
# додаємо випадковий фоновий шум
background_volume = np.random.uniform(0, 0.1)
# отримуємо файли з випадковим шумом із вибірки
background_files = get_files('_background_noise_')
background_file = np.random.choice(background_files)
background_tensor = tfio.audio.AudioIOTensor(background_file)
background_start = np.random.randint(0, len(background_tensor) - 16000)
# нормалізація фонового шуму
background = tf.cast(background_tensor[background_start:background_start+16000], tf.float32)
background = background - np.mean(background)
background = background / np.max(np.abs(background))
# поєднуємо фоновий шум із аудіо файлом
audio = audio + background_volume * background
# отримуємо спектрограму
return get_spectrogram(audio)
```

Рисунок 3.9 – Додавання фонового шуму до аудіозаписів

Щоб переконатися, що збалансований набір даних отримано, необхідно додати до вибірки більше зразків слова «Sheila». Це також допомагає нейронній мережі краще навчитися, оскільки в 1-секундній вибірці буде кілька зразків слова з різними фоновими шумами та в різних положеннях (Рис. 3.10).

```
# проводимо обробку всіх слів та файлів
for word in tqdm(words, desc="Processing words"):
    if '_' not in word:
        # створюємо більшу кількість екземплярів цільового слова
        repeat = 70 if word == 'sheila' else 1
        process_word(word, repeat=repeat)
```

Рисунок 3.10 – Балансування вибірки

Потім необхідно додати зразки з фонового шуму, проглянути кожен файл фонового шуму та поділити його на 1-секундні зразки, обчислити спектрограму

та додати їх до негативних прикладів. Розподіл вибірки зображено на рисунку 3.11.

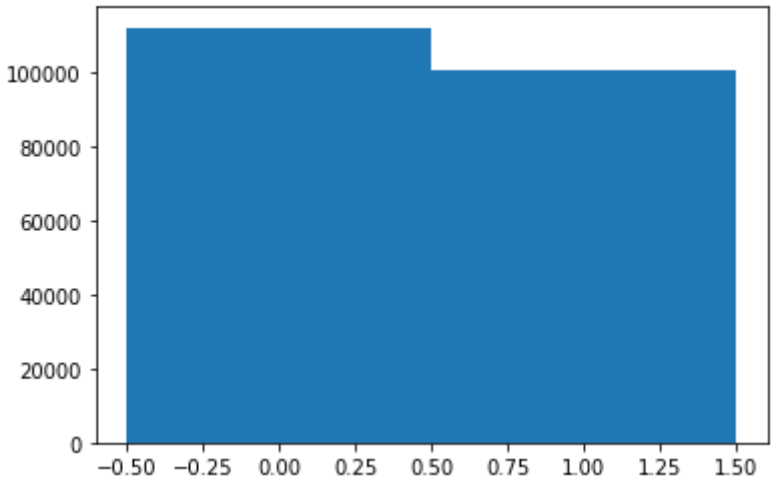


Рисунок 3.11 – Розподіл вибірки

Судячи з графіку, отримано збалансовану вибірку з приблизно однаковим розподілом позитивних та негативних результатів.

З усіма цими даними створено набір даних для навчання, валідації та тестування, що має достатньо великі розміри.

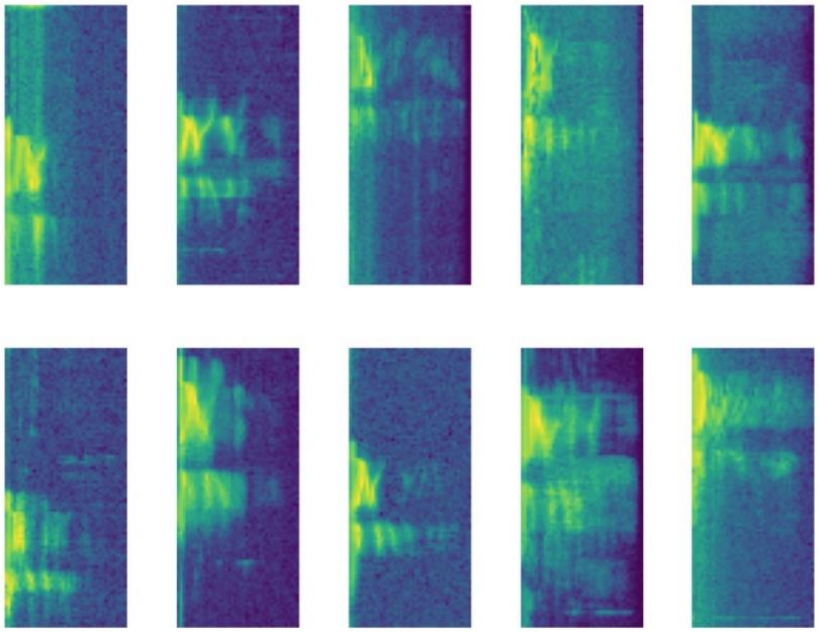


Рисунок 3.12 – Приклад створених спектрограм для цільового слова «Sheila»

Екземпляри декількох створених спектрограм для цільового слова «Sheila» зображено на рисунку 3.12. Навчальні дані підготовлені повністю.

Для розроблюваної моделі важливе лише виявлення слова «Sheila», тому мітки Y змінено так, щоб воно було 1 для «Sheila» і 0 для всього іншого (Рис. 3.13).

```
Y_train = [1 if y == words.index('sheila') else 0 for y in Y_train_cats]
Y_validate = [1 if y == words.index('sheila') else 0 for y in Y_validate_cats]
Y_test = [1 if y == words.index('sheila') else 0 for y in Y_test_cats]
```

Рисунок 3.13 – Встановлення міток для вибірок

Ці необроблені дані подано до наборів даних TensorFlow – відбувається налаштування навчальних даних так, щоб вони повторюватися, крім цього випадковим чином перемішувалися і виходили партіями (Рис. 3.14).

```
# створюємо тренувальний датасет
batch_size = 30

train_dataset = Dataset.from_tensor_slices(
    (X_train, Y_train)
).repeat(
    count=-1
).shuffle(
    len(X_train)
).batch(
    batch_size
)

validation_dataset = Dataset.from_tensor_slices((X_validate, Y_validate)).batch(X_validate.shape[0])

test_dataset = Dataset.from_tensor_slices((X_test, Y_test)).batch(len(X_test))
```

Рисунок 3.14 – Створення тренувального датасету

Для побудови акустичної моделі запропоновано використати згорткові нейронні мережі.

У структурі нейронної мережі є перший шар згортки, за яким слідує шар максимального об’єднання, а потім інший шар згортки та шар максимального об’єднання. Результат цього подається в щільно зв’язаний шар і, нарешті, до вихідного нейрона (Рис. 3.15).

```
model = Sequential([
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer1',
           input_shape=(IMG_WIDTH, IMG_HEIGHT, 1)),
    MaxPooling2D(name='max_pooling1', pool_size=(2,2)),
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer2'),
    MaxPooling2D(name='max_pooling2', pool_size=(2,2)),
    Flatten(),
    Dropout(0.2),
    Dense(
        40,
        activation='relu',
        kernel_regularizer=regularizers.l2(0.001),
        name='hidden_layer1'
    ),
    Dense(
        1,
        activation='sigmoid',
        kernel_regularizer=regularizers.l2(0.001),
        name='output'
    )
])
model.summary()
```

Рисунок 3.15 – Створення структури згорткової нейронної мережі

Після тренування моделі отримано таку точність (Таблиця 3.1):

Таблиця 3.1. Точність моделі

Dataset	**Accuracy**
Training Dataset	0.9683
Validation Dataset	0.9567
Test Dataset	0.9562

Точність розпізнавання на тестовому та валідаційному датасеті склала близько 96%. Це досить непоганий результат для простої моделі.

Результат розпізнавання алгоритму показано у таблиці 3.2. Використовуючи високий поріг (0,9) для істинного класу, отримано дуже мало прикладів фонового шуму, який класифікується як "Sheila", і досить багато слів "Sheila" класифікується як фоновий шум.

Таблиця 3.2 – Матриця суміжності моделі

	Predicted Noise	**Predicted Sheila**
Noise	13980	63
Sheila	1616	11054

Для поставленого завдання такий результат цілком підходить, та кількість випадкових пробуджень буде мінімальною.

Ефективність навчання зображено на рисунку 3.16.

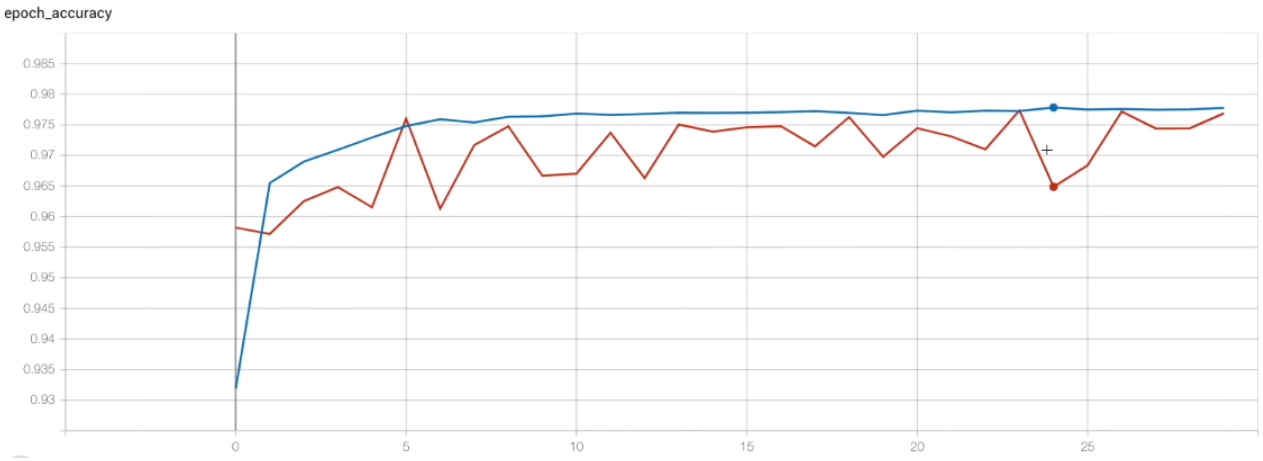


Рисунок 3.16 – Ефективність навчання акустичної моделі

Точність роботи на тестовій вибірці збільшується пропорційно зі збільшенням навчальної вибірки. Ознак перенавчання моделі не спостерігається. Тому для більшої оптимізації можна об’єднати тренувальну, тестову та валідаційну вибірки.

Модель побудована успішно. Наступним кроком буде оптимізація та конвертація моделі за допомогою бібліотеки TensorFlow Lite (Рис.3.17).

```
converter2 = tf.lite.TFLiteConverter.from_saved_model("fully_trained.model")
converter2.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_dataset_gen():
    for i in range(0, len(complete_train_X), 100):
        yield [complete_train_X[i:i+100]]
converter2.representative_dataset = representative_dataset_gen
# converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
converter2.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_quant_model = converter2.convert()
open("lite_acoustic_model.tflite", "wb").write(tflite_quant_model)
```

Рисунок 3.17 – Конвертація та оптимізація акустичної моделі

Цей процес перетворення бере створену повну модель і перетворює її на набагато компактнішу версію, яку можна ефективно запускати на мікроконтролерах та мікрокомп’ютерах.

Після цього необхідно скомпілювати tflite модель до коду C за допомогою команди:

```
xxd -i lite_acoustic_model.tflite > acoustic_model.cc
```

3.3 Ефективність оптимізації підходів до створення проекту додатку голосового управління засобом IoT

Завдяки використанню бібліотеки Tensorflow Lite для машинного навчання вдалося створити акустичну модель, скомпільовану на мові C, що дозволяє їй працювати ефективно навіть на найменших комп’ютерах з обмеженою потужністю. Отримані ключові переваги:

- Невибагливість до системних ресурсів.
- Ефективність на тестових та валідаційних даних склала близько 95%, що є гарним результатом.
- Завдяки підвищення порогу, кількість помилкових спрацьовувань мінімізовано.
- Швидкість реакції на ключове слово складає приблизно 100мс. Крім цього варто наголосити, що існують можливості для оптимізації створеної моделі.

Крім цього розроблений алгоритм має певні недоліки:

- Для навчання мережі необхідно велику кількість навчальних даних, що створено в різних умовах, різними мовцями.
- Не вдалося повністю позбутися помилкових спрацьовувань. Про це свідчить матриця суміжності. Це можна вирішити продовженням та оптимізацією навчання. Крім цього для запобігання названої проблеми можна використати апаратні вдосконалення, такі як мікрофонні мости. Це дозволить відсіювати фоновий шум та збільшити точність виділення голосового повідомлення, що пропорційно призведе до покращення точності алгоритму.
- Система спрацьовує на слова, що схожі на активаційне. Так, для слова «Sheila» це можуть бути слова «Shell», «Шило» і т.п. Щоб цього запобігти необхідно більшу кількість негативних тренувальних даних, в які буде включено схожі слова.

Отже, створений алгоритм може стати базою для подальших досліджень, створення комерційних проектів та розробки власних систем IoT. При чому продуктивність алгоритму дозволяє налаштувати дистрибуцію на слабких системах, що допоможе мінімізувати витрати та зменшити розміри самого пристрою.

Висновки до третього розділу

Було проаналізовано потоки даних у мережі інтернету речей та виділено критичні точки. Так, використання сучасних методів та бібліотек не є ресурсно ефективним при створенні механізмів пробудження пристроїв IoT по ключовому слову. Це встановлює певні обмеження щодо мінімальних апаратних вимог до пристроїв.

Для удосконалення механізму пробудження по ключовому слову було створено модель за допомогою інструментів, представлених бібліотекою машинного навчання для мови програмування Python TensorFlow. Було використано алгоритм згорткових нейронних мереж для створення акустичної моделі. Точність розпізнавання заданого слова склала близько 95% та швидкість відгуку близько 100мс. Мінімізована за допомогою інструментарію бібліотеки

TensorFlow Lite модель може бути використана на малопотужних комп’ютерах та мікроконтролерах.

ВИСНОВКИ

У ході виконання магістерської роботи було розглянуто основи та історичні особливості створення додатків голосового управління. Робота в напрямку розпізнавання мовлення розпочалася ще в 1950х роках. Переломним моментом стало дослідження статистичних методів розпізнавання мовлення на основі прихованих моделей Маркова. Такі методи використовуються і зараз, але з деякими відмінностями. За останнє десятиліття здобувають популярність методи з використанням нейронних мереж.

Крім цього було складено вимоги до інформаційного забезпечення створення додатку голосового управління. Розглянуто критерії оцінки та прийому алгоритмів і систем розпізнавання мовлення.

Також було розглянуто особливості сфери інтернету речей і особливості голосового управління пристроями IoT. Описано типові процеси розпізнавання мовлення та підготовки аудіо даних. Сформульовано завдання дослідження.

Було досліджено архітектуру мережі інтернет речей та її складові. Для збору даних та управління пристроями IoT використовуються датчики та виконавчі механізми. Класична архітектура мережі включає в собі такі рівні: фізичний, транспортний, моніторингу, попередньої обробки, зберігання та безпеки.

Також було розглянуто архітектуру системи розпізнавання мовлення. Така система містить модуль введення голосового сигналу (фронтенд) та забезпечує попередню обробку. Після цього інформація потрапляє до декодера, який складається із словника, мовної моделі та акустичної моделі.

Було проаналізовано та описано алгоритми, які використовуються в сучасних системах розпізнавання мовлення. Два основних: статистичний алгоритм прихованих моделей Маркова та алгоритми машинного навчання. Для побудови мовних моделей використовують рекурентні нейронні мережі, а для акустичних моделей використовують згорткові нейронні мережі. Для перетворення отриманого текстового повідомлення в машинні команди використовуються методи NLP (Natural Language Processing).

Використання алгоритмів машинного навчання збільшило точність розпізнавання біглого мовлення, але в свою чергу їх використання поступається статистичним алгоритмам у використанні обчислювальних ресурсів.

Було проаналізовано потоки даних у мережі інтернету речей та виділено критичні точки. Так, використання сучасних методів та бібліотек не є ресурсно ефективним при створенні механізмів пробудження пристроїв IoT по ключовому слову. Це встановлює певні обмеження щодо мінімальних апаратних вимог до пристроїв.

Для удосконалення механізму пробудження по ключовому слову було створено модель за допомогою інструментів, представлених бібліотекою машинного навчання для мови програмування Python TensorFlow. Було використано алгоритм згорткових нейронних мереж для створення акустичної моделі. Точність розпізнавання заданого слова склала близько 95% та швидкість відгуку близько 100мс. Мінімізована за допомогою інструментарію бібліотеки TensorFlow Lite модель може бути використана на малопотужних комп'ютерах та мікроконтролерах.

Результати дослідження можуть бути використаними для подальших досліджень у сфері розпізнавання мовлення та голосовим управлінням пристроями IoT. Крім цього ефективність розробленої моделі дозволяє створювати власні мережі інтернету речей, а продуктивність дозволяє використовувати модель на слабких пристроях для мінімізації витрат та розмірів самого пристрою.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yeshchenkov V., Mezentseva O. Identification of the Main Problems of Collection and Analysis of Speech Data Using Machine Learning Information Technology and Interactions (Satellite): Conference Proceedings, December 04, 2020, Kyiv, Ukraine / Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor).). - Kyiv: Stylos, 2020. – P.184-186
2. K. H. Davis, R. Biddulph, and S. Balashek. Automatic Recognition of Spoken Digits // Journal of the Acoustical Society of America, Том 24, № 6, 1952. с. 627-642.
3. J. W. Forgie, C. D. Forgie. Results Obtained from a Vowel Recognition Computer // Journal of the Acoustical Society of America, Том. 31, №11, 1959. с. 1480-1489.
4. J. Suzuki, K. Nakata. Recognition of Japanese Vowels—Preliminary to the Recognition of Speech // Journal of the Radio Research Laboratory, Том 37, №8, 1961. с. 193-212 .
5. Goodine, D., Hirschman, L., Polifroni, J., Seneff, S., Zue, V. Evaluating interactive spoken language systems // Proceedings of the 2nd International Conference on Spoken Language Processing (ICSLP’92), Банф, 1992. с. 201–204.
6. Walker, M., Litman, D., Kamm, C., Abella, A. PARADISE: A framework for evaluating spoken dialogue agents // Proc. of the ACL/EACL 35th Ann. Meeting of the Assoc. for Computational Linguistics, Мадрид, 1997. с. 271–280.
7. Young, S. та інші. Multilingual large vocabulary speech recognition: The European SQALE project // Comput. Speech Lang, 11(1), 1997. с. 73–89.
8. Fraser, N. Assessment of Interactive Systems. Handbook on Standards and Resources for Spoken Language Systems: Mouton de Gruyter, Берлін, 1997. с. 564–615.
9. Pallett, D., Fiscus, J., Fisher, W., Garofolo, J. Benchmark tests for the DARPA spoken language program // Proc. DARPA Human Language Technology Workshop, Принстон, 1993. с. 7–18.
10. Strik, H., Cucchiarini, C., Kessens, J. Comparing the performance of two CSRs: How to determine the significance level of the differences // Proc. 7th Eur. Conf. on Speech Communication and Technology (EUROSPEECH 2001 – Scandinavia), Aalborg, 2001. с. 2091–2094.

11. Jacquemin, C., Mariani, J., Paroubek, P. Parameters describing the interaction with spoken dialogue systems using evaluation within HLT programs // Results and trends: Proc. CLASS Pre-Conf. Workshop to LREC 2000, Женева, Афіни, 2005.
12. Hirschman L., Pao C. The cost of errors in a spoken language system. // Proc. 3rd Eur. Conf. on Speech Communication and Technology (EUROSPEECH'93), Берлін, 1993. с. 1419–1422.
13. Gibbon, D., Mertins, I., Moore, R. Handbook of Multimodal and Spoken Dialogue Systems // Resources, Terminology and Product Evaluation. Бостон, 2000.
14. Young, S. Speech recognition evaluation: A review of the ARPA CSR programme // Proc. Speech and Language Technology (SALT) Club Workshop on Evaluation in Speech and Language Technology, Шефілд, 1997. с. 197–205.
15. Hirschman L., Thompson, H. Overview of evaluation in speech and natural language processing: Survey of the State of the Art in Human Language Technology // Cambridge University Press and Giardini Editori, Піза, 1997. с. 409–414.
16. Jeksoch, U. Voice and Speech Quality Perception. Assessment and Evaluation. Берлін, 2005.
17. ISO 9241-11. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). Part 11: Guidance on Usability. International Organization for Standardization, Женева, 1998.
18. Möller, S. . Quality of Telephone-based Spoken Dialogue Systems. Нью-Йорк, 2005.
19. Möller, S. A new taxonomy for the quality of telephone services based on spoken dialogue systems // Proc. 3rd SIGdial Workshop on Discourse and Dialogue. Філадельфія, 2002. с. 142–153.
20. Picone, J., Goudie-Marshall, K., Doddington, G., Fisher, W. Automatic text alignment for speech system evaluation // IEEE Trans. Acoust., Speech, Signal Process. 34(4), 1996. с. 780–784.
21. J. Melià-Seguí. RFID EPC-Gen2 for Postal Applications: A Security and Privacy Survey // Proc. 2010 RFID-Technology and Applications Int. Conf., Гуанчжоу, 2010.
22. G. Lawton. Machine-to-Machine Technology Gears Up for Growth // IEEE Computer Society, том 37, теза 9, Нью-Йорк: IEEE, 2004, с. 12-15.

23. Sometimes, Less Power is More [Електронний ресурс]: веб-сайт. URL: https://www.qualcomm.com/products/wifi-platforms.
24. Brain Power [Електронний ресурс]: веб-сайт. URL: http://research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml.
25. Bluetooth/ Bluetooth Low Energy [Електронний ресурс]: веб-сайт. URL: http://www.st.com/web/en/catalog/sense_power/FM1968/CL1976/SC1898?sc=bluetoothlowenergy.
26. The Internet of Things [Електронний ресурс]. URL: http://raymondjames.com/pointofview/article.aspx?a=2023.
27. J. McQuivey. Your Voice Will Control the Future [Електронний ресурс]. URL: http://blogs.forrester.com/james_mcquivey/14-04-16-your_voice_will_control_the_future.
28. G. N. Meenakshi, P. K. Ghosh. Automatic Gender Classification Using the Mel Frequency Cepstrum of Neutral and Whispered Speech: a Comparative Study // Twenty First National Conference on Communications, Мумбай, 2015, с. 1-6.
29. S. Salvador, P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time Space // Intelligent Data Analysis, том 11, № 5: IOS Press, 2007, с. 561-580.
30. A. Sannino. Analyzing Discontinuous Speech in EU Conversations: A Methodological Proposal, Journal of Pragmatics // Journal of Pragmatics, том. 38: Elsevier B.V., 2006, с. 543-566.
31. J. Kirriemuir. Speech recognition technologies. Retrieved December 5, 2005.
32. I. Mashal, O. та інші. Choices for interaction with things on Internet and underlying issues // Ad Hoc Networks, том 28, 2015. с 68– 90.
33. O. Said , M. Masud. Towards internet of things: survey and future vision// International Journal of Computer Networks, том 5, № 1, 2013. с. 1–17.
34. R. Khan, S. U. Khan, R. Zaheer, and S. Khan. Future internet: the internet of things architecture, possible applications and key challenges // Proceedings of the 10th International Conference on Frontiers of Information Technology (FIT ’12), December 2012. с. 257–260.

35. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things // Proceedings of the 1st ACM MCC Workshop on Mobile Cloud Computing, 2012. с. 13–16.
36. M. Aazam, E.-N. Huh. Fog computing and smart gateway based communication for cloud of things// Proceedings of the 2nd IEEE International Conference on Future Internet of Things and Cloud (FiCloud ’14), Барселона, 2014. с. 464–470.
37. Pallavi Sethi, Smruti R. Sarangi. Internet of Things: Architectures, Protocols, and Applications. 2017. с. 5.
38. A. Schmidt and K. Van Laerhoven. How to build smart appliances? // IEEE Personal Communications, том 8, № 4, 2001. с. 66–71.
39. W. Z. Khan, Y. Xiang, M. Y. Aalsalem, Q. Arshad. Mobile phone sensing systems: a survey // IEEE Communications Surveys & Tutorials, том 15, № 1, 2013. с. 402–427.
40. N. Bui , M. Zorzi. Health care applications: a solution based on the internet of things. // Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL ’11), ACM, Барселона, 2011.
41. M. J. McGrath, C. N. Scanaill. Body-worn, ambient, and consumer sensing for health applications. // Sensor Technologies, 2013. с. 181–216.
42. A. Pantelopoulos , N. G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis // IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, том 40, № 1, 2010. с. 1–12.
43. J. H. Gruzelier. EEG-neurofeedback for optimising performance. I: a review of cognitive and affective outcome in healthy participants. // Neuroscience and Biobehavioral Reviews, том 44, 2014. с. 124–141.
44. P. K. Sekhar, E. L. Brosha, R. Mukundan, F. H. Garzon. Chemical sensors for environmental monitoring and homeland security // The Electrochemical Society Interface, том 19, № 4, 2010. с. 35–40.
45. N. Bhattacharyya, R. Bandhopadhyay. Electronic nose and electronic tongue. // Nondestructive Evaluation of Food Quality, Берлін 2010. с. 73–100.
46. X. Huang, A. Acero , H.-W. Hon. Spoken Language Processing: a guide to theory, algorithm, and system development. Прентіс Хол, 2001.

47. D. Jurafsky, J. H. Martin. Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Прентіс Хол, 2009.
48. M. A. Anusuya , S. Katti. Front end analysis of speech recognition: a review // Int. J. Speech Technology, том 14, № 2, 2011. с. 99–145.
49. Karpagavalli S, Chandra E. A Review on Automatic Speech Recognition Architecture and Approaches, 2016.
50. R. Lawrence, B.-H. Juang. Fundamentals of Speech Recognition‖, Прентіс Хол 1993.
51. Speech Recognition — GMM, HMM [Електронний ресурс]: веб-сайт. URL: https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196.
52. Mark Gales, Steve Young. The Application of Hidden Markov Models in Speech Recognition: Foundations and Trends in Signal Processing. Том 1, № 3, 2007. с. 205-216.
53. The 3 Deep Learning Frameworks For End-to-End Speech Recognition That Power Your Devices [Електронний ресурс]: веб-сайт. URL: https://heartbeat.fritz.ai/the-3-deep-learning-frameworks-for-end-to-end-speech-recognition-that-power-your-devices-37b891ddc380.
54. W. Chan, N. Jaitly, Q. Le, O. Vinyals. Listen, Attend, and Spell // in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2015.
55. N. Jaitly, D. Sussillo, Q. Le, O. Vinyals, I. Sutskever, S. Bengio. A Neural Transducer, 2016.
56. N. Jaitly, W. Chan, Y. Zhang. Very Deep Convolutional Networks for End-to-End Speech Recognition, 2016.
57. What Is Natural Language Understanding (NLU) & How Does It Work? [Електронний ресурс]: веб-сайт. URL: https://monkeylearn.com/blog/natural-language-understanding/.
58. Intent Recognition in Chatbots in 2021 [Електронний ресурс]: веб-сайт. URL: https://research.aimultiple.com/chatbot-intent/.
59. What is IoT architecture? [Електронний ресурс]: веб-сайт. URL: https://www.avsystem.com/blog/what-is-iot-architecture/.

60. Overview of the CMUSphinx toolkit [Електронний ресурс]: веб-сайт. URL: https://cmusphinx.github.io/wiki/tutorialoverview/.
61. About the Kaldi project [Електронний ресурс]: веб-сайт. URL: https://kaldi-asr.org/doc/about.html.
62. Porcupine SDK Introduction [Електронний ресурс]: веб-сайт. URL: https://picovoice.ai/docs/porcupine/.
63. Introduction to TensorFlow [Електронний ресурс]: веб-сайт. URL: https://www.tensorflow.org/learn.
64. speech_commands [Електронний ресурс]: веб-сайт. URL: https://www.tensorflow.org/datasets/catalog/speech_commands.

ДОДАТКИ

Додаток А

Програмний код реалізації алгоритму розпізнавання мовлення

```
# перетворення аудіо файлу в спетккрограму
def process_file(file_path):
    # завантаження аудіо файлу
    audio_tensor = tfio.audio.AudioIOTensor(file_path)
    # конвертація аудіо файлу в набір чисел від -1 до 1
    audio = tf.cast(audio_tensor[:], tf.float32)
    audio = audio - np.mean(audio)
    audio = audio / np.max(np.abs(audio))
    # змінюємо положення звуку у сигналі випадковим чином
    voice_start, voice_end = get_voice_position(audio, NOISE_FLOOR)
    end_gap=len(audio) - voice_end
    random_offset = np.random.uniform(0, voice_start+end_gap)
    audio = np.roll(audio,-random_offset+end_gap)
    # додаємо випадковий фоновий шум
    background_volume = np.random.uniform(0, 0.1)
    # отримуємо файли з випадковим шумом із вибірки
    background_files = get_files('_background_noise_')
    background_file = np.random.choice(background_files)
    background_tensor = tfio.audio.AudioIOTensor(background_file)
    background_start = np.random.randint(0, len(background_tensor) - 16000)
    # нормалізація фонового шуму
    background = tf.cast(background_tensor[background_start:background_start+16000], tf.float32)
    background = background - np.mean(background)
    background = background / np.max(np.abs(background))
    # поєднуємо фоновий шум із аудіо файлом
    audio = audio + background_volume * background
    # отримуємо спектрограму
    return get_spectrogram(audio)
```

```
def get_spectrogram(audio):
    # нормалізація звукового сигналу
    audio = audio - np.mean(audio)
    audio = audio / np.max(np.abs(audio))
    # створення спектрограми
    spectrogram = audio_ops.audio_spectrogram(audio,
                                              window_size=320,
                                              stride=160,
                                              magnitude_squared=True).numpy()
    # зменшимо кількість частотних вікон для збільшення чутливості спектрограми
    spectrogram = tf.nn.pool(
        input=tf.expand_dims(spectrogram, -1),
        window_shape=[1, 6],
        strides=[1, 6],
        pooling_type='AVG',
        padding='SAME')
    spectrogram = tf.squeeze(spectrogram, axis=0)
    spectrogram = np.log10(spectrogram + 1e-6)
    return spectrogram
```

```
# перехід і отримання всіх файлів у директорії
def get_files(word):
    return gfile.glob(SPEECH_DATA + '/'+word+'/*.wav')

# отримання позиції голосу
def get_voice_position(audio, noise_floor):
    audio = audio - np.mean(audio)
    audio = audio / np.max(np.abs(audio))
    return tfio.experimental.audio.trim(audio, axis=0, epsilon=noise_floor)
```

```
Y_train = [1 if y == words.index('sheila') else 0 for y in Y_train_cats]
Y_validate = [1 if y == words.index('sheila') else 0 for y in Y_validate_cats]
Y_test = [1 if y == words.index('sheila') else 0 for y in Y_test_cats]
```

```
# створюємо тренувальний датасет
batch_size = 30

train_dataset = Dataset.from_tensor_slices(
    (X_train, Y_train)
).repeat(
    count=-1
).shuffle(
    len(X_train)
).batch(
    batch_size
)

validation_dataset = Dataset.from_tensor_slices((X_validate, Y_validate)).batch(X_validate.shape[0])

test_dataset = Dataset.from_tensor_slices((X_test, Y_test)).batch(len(X_test))
```

```
model = Sequential([
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer1',
           input_shape=(IMG_WIDTH, IMG_HEIGHT, 1)),
    MaxPooling2D(name='max_pooling1', pool_size=(2,2)),
    Conv2D(4, 3,
           padding='same',
           activation='relu',
           kernel_regularizer=regularizers.l2(0.001),
           name='conv_layer2'),
    MaxPooling2D(name='max_pooling2', pool_size=(2,2)),
    Flatten(),
    Dropout(0.2),
    Dense(
        40,
        activation='relu',
        kernel_regularizer=regularizers.l2(0.001),
        name='hidden_layer1'
    ),
    Dense(
        1,
        activation='sigmoid',
        kernel_regularizer=regularizers.l2(0.001),
        name='output'
    )
])
```

```
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath="checkpoint.model",
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)

history = model.fit(
    train_dataset,
    steps_per_epoch=len(X_train) // batch_size,
    epochs=epochs,
    validation_data=validation_dataset,
    validation_steps=1,
    callbacks=[tensorboard_callback, model_checkpoint_callback]
)
```

```
training_spectrogram = np.load('training_spectrogram.npz')
validation_spectrogram = np.load('validation_spectrogram.npz')
test_spectrogram = np.load('test_spectrogram.npz')

X_train = training_spectrogram['X']
X_validate = validation_spectrogram['X']
X_test = test_spectrogram['X']

complete_train_X = np.concatenate((X_train, X_validate, X_test))
```

```
# конвертація акустичної моделі за допомогою Tf lite
converter2 = tf.lite.TFLiteConverter.from_saved_model("fully_trained.model")
converter2.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_dataset_gen():
    for i in range(0, len(complete_train_X), 100):
        yield [complete_train_X[i:i+100]]
converter2.representative_dataset = representative_dataset_gen
# converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
converter2.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_quant_model = converter2.convert()
open("lite_acoustic_model.tflite", "wb").write(tflite_quant_model)
```