ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методики транскрибації на основі нейронних мереж»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр САЧУК
(підпис)

Виконав: здобувач вищої освіти група ПДМ-62
Олександр САЧУК

Керівник: Андрій БОНДАРЧУК
д.т.н., професор

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сачук Олександру Васильовичу

1. Тема кваліфікаційної роботи: Розробка методики транскрибації на основі нейронних мереж

керівник кваліфікаційної роботи Андрій БОНДАРЧУК д.т.н., професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, вимоги до результатів роботи.

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

1. Огляд існуючих методів транскрибації.
2. Дослідження методологій нейронної мережі для транскрибації.
3. Реалізація та тестування системи.

5. Перелік графічного матеріалу: *презентація*

1. Огляд і аналіз існуючих підходів до задачі автоматичної транскрибації мовлення.
2. Основні компоненти програми для транскрибації мовлення на основі нейронних мереж.
3. Архітектура та функціональність системи транскрибації.
4. Унікальність розробки: гнучкість та адаптивність.
5. Відкриті технології у розробці транскрибації.
6. Механізми збору даних та постійного вдосконалення.

6. Дата видачі завдання «19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-30.10.23	
2	Огляд існуючих методів транскрибації	31.10-06.11.23	
3	Дослідження методологій нейронної мережі для транскрибації	07.11-20.11.23	
4	Реалізація та тестування системи	21.11-04.12.23	
7	Оформлення роботи: вступ, висновки, реферат	05.12-11.12.23	
8	Розробка демонстраційних матеріалів	12.12-19.12.23	

Здобувач вищої освіти _____ Олександр САЧУК
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 80 стор., 7 табл., 11 рис., 26 джерел.

Мета роботи – розробка методики автоматичної транскрибації української мови на основі нейронних мереж

Об'єкт дослідження –автоматизований процес транскрибації української мови

Предмет дослідження – методика транскрибації на основі нейронних мереж та алгоритмів глибинного навчання.

Короткий зміст роботи: У роботі проведено огляд існуючих методів транскрибації. Проаналізовано існуючі методології нейронних мереж для транскрибації. Реалізоване та протестоване програмне забезпечення для транскрибації української мови.

КЛЮЧОВІ СЛОВА: ТРАНСКРИБАЦІЯ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА АУДІОДАНИХ.

ABSTRACT

Text part of the master's qualification work:
80 pages, 7 table, 11 pictures, 26 sources.

The purpose of the work - develop a methodology for automatic transcription of the Ukrainian language based on neural networks.

Object of research – an automated process of transcription of the Ukrainian language

Subject of research – transcription technique based on neural networks and deep learning algorithms.

Summary of the work: an overview of existing transcription methods was carried out. The existing methodologies of neural networks for transcription are analysed. Implemented and tested software for transcription of the Ukrainian language.

KEYWORDS: TRANSCRIPTION, NEURAL NETWORKS, PROCESSING OF AUDIO DATA

ЗМІСТ

ВСТУП

У сучасному цифровому світі, що стрімко розвивається, транскрибація мовлення, або перетворення усного мовлення на письмовий текст, відіграє ключову роль у багатьох аспектах нашого життя. Від автоматизованих систем обслуговування клієнтів до розробки голосових асистентів, від створення субтитрів для відео до підтримки людей з порушеннями слуху – сфери застосування безмежні. Це вимагає розробки продуктивних, точних та швидких систем транскрибації, які можуть ефективно впоратися з різноманітністю мовленнєвих відмінностей та акцентів.

Інновації в області штучного інтелекту та машинного навчання відкривають нові можливості для покращення систем транскрибації. Зокрема, використання нейронних мереж дозволяє створювати більш гнучкі та адаптивні моделі, які можуть навчатися з часом, покращуючи свою точність та ефективність.

Точність транскрибації має критичне значення в багатьох сферах, включаючи юридичну, медичну та освітню. У цих галузях неточності у перекладі мовлення можуть мати серйозні наслідки. Тому розробка вдосконалених систем, здатних точно і швидко обробляти природну мову, є важливою задачею сучасної технологічної галузі.

З іншого боку, доступність та зручність використання таких систем робить технологію транскрибації мовлення цінним інструментом у повсякденному житті. Від особистих помічників у смартфонах до автоматизації рутинних завдань, здатність перетворювати голос на текст спрощує взаємодію людей з технологіями, роблячи ці взаємодії більш природними та інтуїтивними.

Таким чином, розвиток і вдосконалення систем транскрибації на основі нейронних мереж є актуальним завданням, яке відкриває нові перспективи для різних сфер життєдіяльності та сприяє подальшому прогресу в галузі обробки природної мови.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ТРАНСКРИБАЦІЇ

1.1 Статистичні методи транскрибації

У рамках поглибленого аналізу статистичних методів транскрибації мовлення, який є частиною моєї магістерської роботи, особлива увага приділяється ретроспективі розвитку та еволюції цих технік у контексті обробки природної мови. Виходячи з ранніх досліджень у цій області, статистичні методи лягли в основу багатьох традиційних систем автоматичного розпізнавання мовлення, використовуючи такі інструменти, як гауссові суміші та баєсові мережі. Вони були привабливими через свою здатність моделювати мовлення як серію ймовірнісних подій та їхніх взаємозв'язків, що дозволяло створювати функціональні системи навіть за відносно обмеженого обсягу даних для навчання.

Проте, із зростанням обсягів доступних даних та збільшенням обчислювальних потужностей, стало очевидним, що традиційні статистичні методи мають певні обмеження. Хоча ці методи і забезпечували непоганий рівень точності, вони часто боролися з комплексністю природної мови, такою як змінність акустики, контексту та диктора. Також вони вимагали значних зусиль для попереднього аналізу та налаштування параметрів, що становило бар'єр для швидкого адаптування до нових мов або специфічних вимог користувачів.

Щоб подолати ці виклики, дослідники почали впроваджувати більш складні статистичні моделі, такі як приховані марковські моделі, які дозволяли краще зрозуміти послідовність мовлення та структуру мови. Ці моделі забезпечували більшу гнучкість та адаптивність, але їх ефективність все ще значно залежала від ретельного вибору особливостей та параметрів моделі. Незважаючи на це, приховані марковські моделі довгий час були стандартом у галузі автоматичного розпізнавання мовлення, пропонуючи математично зручний і порівняно ефективний спосіб для моделювання мовленнєвих послідовностей.

Компанія Philips ще у 1990-х роках почала активно розвивати власні технології обробки мовленнєвих сигналів. Зокрема, підрозділ Speech Processing Solutions став піонером у застосуванні статистичних методів для створення комерційних рішень з автоматичної транскрибації усного мовлення.

Фахівцями Philips було розроблено програмне забезпечення, яке використовувало складні статистичні моделі на кшталт ланцюгів Маркова та баєсових мереж для розпізнавання мовних структур. Система могла в режимі реального часу перетворювати аудіопотік з мікрофону у текст, відстежуючи при цьому особливості мовця та адаптуючись до різних акцентів та швидкості мовлення.

Ця технологія отримала назву Advanced Speech Recognition і вбудовувалась компанією Philips у різноманітні пристрої - від професійних офісних телефонів до побутових диктофонів. Вона надавала можливість користувачам, наприклад, диктувати тексти та команди безпосередньо у слухавку телефону чи диктофон, а система автоматично перетворювала їх у цифровий текст.

Це значно полегшувало та прискорювало роботу з документами, особливо в офісних умовах. Користувачі отримували можливість швидко створювати звіти, протоколи та інші тексти без необхідності їх ручного набору. Тобто, за допомогою голосу можна було оперативно керувати введенням та обробкою інформації, що для того часу було значним технологічним проривом у сфері цифровізації офісних процесів.

Бельгійська компанія Lernout & Hauspie, заснована у 1987 році, була піонером у галузі технологій мовленнєвого аналізу та синтезу. Зокрема, вона розробила інноваційне програмне забезпечення для перетворення усного мовлення у текст, яке використовувало найсучасніші на той час статистичні та лінгвістичні методи.

Фахівцям Lernout & Hauspie вдалося поєднати складний апарат на основі прихованих марковських моделей, який відповідав за статистичний аналіз звукових сигналів, з експертними лінгвістичними правилами та модулем перевірки семантичної коректності тексту. Це забезпечило унікальну якість транскрибації,

особливо для спеціалізованих галузей, таких як медицина чи юриспруденція, де потрібна максимальна точність у передачі специфічної термінології та скорочень.

Завдяки цим технологіям, програмне забезпечення компанії широко застосовувалося для створення текстових медичних звітів та записів судових засідань. Система дозволяла автоматизувати роботу лікарів, юристів та інших фахівців, яким потрібно було оперативно обробляти великі обсяги усної інформації. Продукт користувався високою надійністю і значним попитом в 1990-2000-х роках.

Компанія ScanSoft до 1999 року спеціалізувалася виключно на програмному забезпеченні для оптичного розпізнавання тексту. Проте наприкінці 1990-х років технології обробки мовлення почали стрімко набирати популярність у зв’язку з розвитком голосового керування пристроями, інтерактивних меню та цифрових помічників.

Усвідомлюючи перспективність цього напряму, ScanSoft вирішує розширити свій бізнес і виходить на ринок speech analytics шляхом поглинання в 1999 році провідної на той час компанії SpeechWorks, яка володіла передовими розробками у сфері аналізу та синтезу мовлення.

Отримавши доступ до унікальних наукових напрацювань SpeechWorks, команда технологів ScanSoft розпочала роботу над власним революційним на той час продуктом. Його основою стала гібридна нейронно-статистична архітектура, яка об’єднала кращі практики обох підходів.

З одного боку застосовувалися баєсові ймовірнісні моделі, марковські ланцюги та метод головних компонент для статистичного аналізу мовних структур та звукових сигналів. З іншого використовувалися багатошарові нейронні мережі прямого поширення для виділення та класифікації характерних ознак мовлення, що дозволяло значно підвищити швидкість та якість розпізнавання.

Поєднання цих підходів дало синергетичний ефект, в результаті чого створена технологія виявилася на порядок більш точною за конкурентів. Її особливістю була можливість розпізнавання у реальному часі на основі досить коротких фрагментів мовлення (всього 2-3 секунди).

Це дозволило швидко інтегрувати розробку ScanSoft в інтерактивні системи телефонії, такі як інтелектуальні меню автоматизованих call-центрів, системи голосового набору номерів мобільних та стаціонарних телефонів, а також у мобільні та веб-орієнтовані додатки, які потребували швидкого голосового вводу та пошуку інформації.

Значна увага приділялася питанням безпеки та захисту персональних даних в оброблюваному мовленні. Розробниками були впроваджені складні математичні методи шифрування та знеособлення для унеможливлення витоку вразливої інформації. Це дозволило системі успішно проходити сертифікації та відповідати вимогам законодавства у цій сфері.

Загалом розроблена ScanSoft технологія транскрибації мовлення відрізнялася високою якістю, надійністю та швидкодією. Вона користувалася величезним попитом у різноманітних організацій по всьому світу, від комерційних компаній до держустанов та наукових закладів. Саме технологічне лідерство у цій сфері послугувало основною причиною придбання ScanSoft компанією Nuance та створення глобального технологічного гіганту у 2005 році.

Статистичні методи транскрибації базуються на побудові статистичних моделей зв'язку між характеристиками аудіосигналу та відповідними одиницями тексту (словами, фонемами). Ці методи застосовують теорію ймовірностей та математичну статистику для формалізації процесу транскрибації.

Одним з основоположних підходів є використання ланцюгів Маркова та прихованих марковських моделей (ПММ). Ланцюг Маркова описує процес, що переходить з одного стану в інший з певною ймовірністю, яка залежить лише від попереднього стану. Ця концепція дозволяє моделювати послідовність фонем у мовленні як стани Маркова.

Формально, ПММ задається наступним чином:

- Множина станів $S = \{s1, s2, ... , sn\}$ (кожен стан відповідає фонемі)
- Початковий розподіл ймовірностей по станах $\pi = \{\pi1, \pi2 , ... \pi n \}$, де $\pi i = P(q1 = si)$
- Матриця ймовірностей переходів $A = \{aij\}$, де $aij = P(qt+1 = sj \mid qt = si)$

- Ймовірності спостережень (емісій) B = {bj(k)} для кожного стану sj та кожного спостереження ok.

При цьому самі стани залишаються прихованими (звідси назва), а на виході моделі спостерігаються певні характеристики, які залежать від поточного стану.

Наприклад, у ASR системах спостерігаються акустичні ознаки аудіосигналу, на основі яких робляться висновки щодо найбільш ймовірної послідовностей фонем. Для цього використовують алгоритм Вітербі, який знаходить найбільш вірогідну послідовність прихованих станів виходячи з вимірюваних даних та параметрів ПММ.

Одна з ключових переваг прихованих марковських моделей полягає у їх здатності моделювати часові послідовності, що є основними в мовленні. Вони працюють, передбачаючи наступний стан системи на основі попереднього, що дозволяє обробляти мовлення як послідовність звуків та слів. Ця особливість робить приховані марковські моделі ідеальними для задач, де потрібно враховувати тимчасові залежності, наприклад, при розпізнаванні мовлення чи жестів.

Застосування прихованих марковських моделей в програмному забезпеченні для транскрибації було широко поширене у таких системах, як Dragon NaturallySpeaking, що є однією з перших комерційно успішних систем розпізнавання голосу. Ця програма використовує статистичні моделі для перетворення мовлення користувачів на текст, навчаючись з часом покращувати точність на основі особистих мовних звичок користувача.

Рис. 1.1. Логотип ПЗ Dragon NaturallySpeaking

Dragon NaturallySpeaking - популярна комерційна програма розпізнавання мовлення від компанії Nuance Communications. Вперше представлена у 1997 році під назвою DragonDictate, вона була одним з перших рішень для перетворення мови в текст на персональних комп'ютерах.

На відміну від конкуруючих на той час IBM ViaVoice та Microsoft Speech, які потребували «навчання» користувача, Dragon міг працювати відразу «з коробки» з задовільною якістю. Це досягалося завдяки використанню величезної бази профільних записів та статистичних алгоритмів.

По мірі накопичення даних та вдосконалення технологій, якість розпізнавання Dragon покращилася до рівня 95-98% для стандартного англійського мовлення. З'явилася підтримка інших мов, можливість адаптації до акцентів та професійних галузей на кшталт медичної та юридичної.

На сьогодні Dragon є найпопулярнішим рішенням для голосового введення тексту, керування ПК та автоматизації документообігу. Використовується мільйонами користувачів та організацій по всьому світу, від звичайних офісів до великих компаній, держустанов та наукових закладів. Продовжує удосконалюватися з використанням новітніх досягнень штучного інтелекту.

Іншим важливим прикладом є IBM ViaVoice, програма, яка також використовує статистичні методи для розпізнавання та перетворення мовлення на текст. Такі програми інтенсивно використовували великі корпуси записаного мовлення для створення своїх моделей, забезпечуючи високий рівень точності розпізнавання.

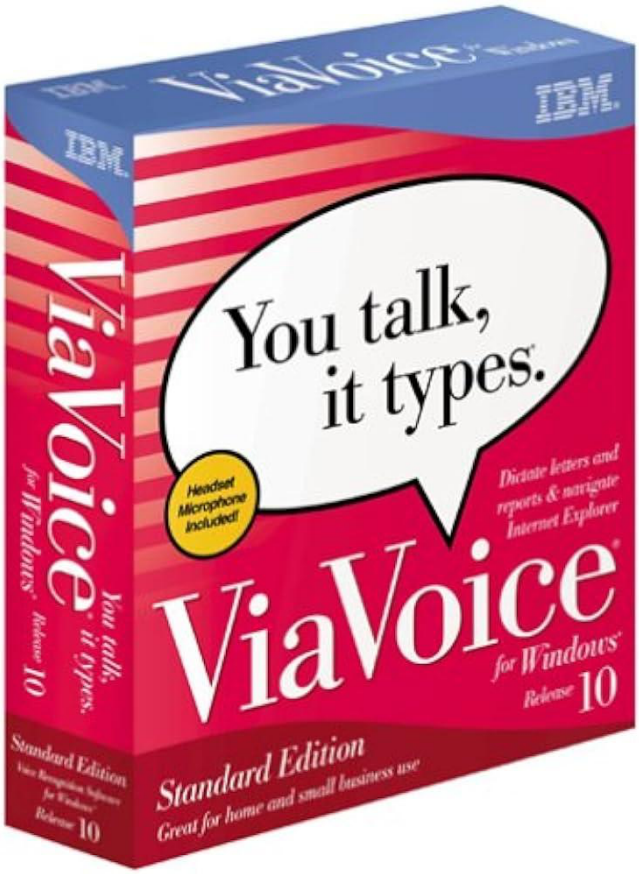


Рис. 1.2. Програмне забезпечення IBM ViaVoice

IBM ViaVoice - це комерційний програмний продукт для розпізнавання та транскрибації мовлення, розроблений компанією IBM. Вперше представлений у 1997 році, ViaVoice був одним з перших масових рішень для перетворення мови в текст на ПК.

У основі ViaVoice лежала інноваційна на той час технологія розпізнавання за допомогою прихованих Марковських моделей та нейронних мереж. На відміну від попередніх підходів, що використовували лінгвістичні правила, моделі ViaVoice навчалися безпосередньо на великих обсягах записів людської мови.

Перші версії підтримували розпізнавання для англійської мови із середньою точністю близько 90-95%. З часом з'явилася підтримка інших європейських та азійських мов. Також поліпшувалася якість розпізнавання, особливо у шумних умовах.

На піку популярності у 2000-х роках ViaVoice використовувався на мільйонах комп'ютерів для створення текстових документів, електронної пошти,

запису команд та автоматизації офісних задач. Згодом технологія стала основою для сучасних цифрових помічників на кшталт Siri та Alexa.

Ці та інші системи, засновані на прихованих марковських моделях, зіграли ключову роль у розвитку технологій автоматичного розпізнавання мовлення. Однак, з приходом глибинного навчання та нейронних мереж, статистичні методи почали поступатися місцем більш потужним та гнучким моделям, здатним краще справлятися з різноманіттям і складністю природної мови.

Статистичні методи транскрибації базуються на використанні статистичних моделей для встановлення зв'язку між аудіосигналом та відповідним текстовим представленням.

Одним з найпростіших статистичних підходів є використання моделі прихованої Маркова. Вона моделює ймовірності переходу між фонемами та ймовірності спостереження певних акустичних ознак для кожної фонеми. Недоліком цього методу є складність моделювання контекстних залежностей у мові.

Більш просунутим є підхід з використанням прихованих Марковських моделей з гаусовими сумішами. Він комбінує Марковські моделі з гаусовим моделюванням розподілу акустичних ознак. Це дозволяє точніше оцінити умовні ймовірності спостережень.

Ще одним поширеним статистичним методом є використання лінійної регресії. З її допомогою будуються лінійні моделі зв'язку між акустичними характеристиками та фонемами. Перевага лінійної регресії – простота побудови та інтерпретації моделі.

Отже, статистичні методи дозволяють формалізувати зв'язок між аудіосигналом та його текстовим еквівалентом на основі ймовірнісних моделей. Їх обмеженням є складність захоплення всіх закономірностей природної мови.

1.2 Методи на основі прихованих марковських моделей

Приховані Марковські моделі (ПММ) є одним з найбільш поширених підходів, що застосовується в задачах автоматичної транскрибації мовлення. Їх популярність зумовлена здатністю ефективно моделювати послідовності слів та фонем з урахуванням контекстних залежностей.

В основі ПММ лежить припущення, що мовленнєвий сигнал можна представити як випадковий процес, який переходить послідовно з одного прихованого стану в інший. Кожен стан відповідає певній фонемі, а ймовірності переходів між станами моделюють послідовності фонем у словах.

Для транскрибації на базі ПММ потрібно побудувати акустичні моделі, які встановлюють зв'язок між прихованими станами (фонемами) та спостережуваними акустичними ознаками. Найчастіше для цього використовують гаусові суміші або штучні нейронні мережі.

Основними перевагами методів на базі ПММ є їх статистична обґрунтованість, здатність моделювати послідовності слів, а також ефективні алгоритми декодування записів. Проте, ці методи мають складнощі з моделюванням залежностей довгого контексту.

Для подолання цієї проблеми було запропоновано удосконалені архітектури ПММ, зокрема ієрархічні ПММ та ПММ з нейронними мережами прогнозування. Вони дозволяють ефективніше моделювати мовні закономірності високого рівня.

Отже, методи на основі ПММ є фундаментальною технологією в задачах автоматичної транскрибації. Незважаючи на деякі обмеження контекстних залежностей, їх широко застосовують як самостійно, так і в поєднанні з іншими підходами.

Методи на основі прихованих марковських моделей (Hidden Markov Models, HMM) займають центральне місце в історії розвитку систем розпізнавання мовлення. Їхня унікальність полягає у здатності моделювати стохастичні секвенції, які є основою мовленнєвих потоків. Використання HMM в розпізнаванні мовлення базується на припущенні, що будь-яке мовленнєве подання може бути

представлене як ряд станів, кожен з яких генерує певні спостереження, що відповідають звукам або фонемам.

Суть методу полягає у створенні математичної моделі, що описує ймовірності переходів між станами і ймовірності генерації спостережень в кожному стані. Це дозволяє моделювати мовленнєві послідовності, не знаючи точної послідовності станів, які генерують спостереження. В результаті, ми отримуємо систему, яка може визначати найбільш ймовірну послідовність станів, що могли виробити дане спостереження, а отже, і найбільш ймовірний текст, що відповідає аудіо.

Завдяки своїй гнучкості та математичній строгості, HMM знайшли широке застосування в різноманітних сферах обробки сигналів. У контексті розпізнавання мовлення, ці моделі стали фундаментом для створення перших систем, які могли ефективно працювати з великими словниками і надавати користувачам змогу взаємодіяти з комп'ютерними системами через усне мовлення.

Однак, незважаючи на великий успіх HMM, застосування цих моделей не без обмежень. Однією з основних проблем є необхідність ретельного підбору параметрів моделі та велика кількість тренувальних даних, необхідних для ефективного навчання. Крім того, приховані марковські моделі можуть виявитися недостатньо гнучкими для моделювання складних мовленнєвих патернів, таких як емоційні інтонації чи нелінійність природного мовлення.

Незважаючи на ці виклики, HMM продовжують залишатися важливим інструментом розпізнавання мовлення, особливо в комбінації з іншими технологіями, такими як нейронні мережі. Сучасні системи часто використовують гібридний підхід, де HMM використовуються для моделювання структури мовлення, а нейронні мережі застосовуються для точного розпізнавання мовленнєвих характеристик.

Прикладами програмного забезпечення, що використовують HMM, можуть бути ранні версії Nuance Communications і рішення від компанії Philips для голосового вводу. Ці системи базувалися на статистичному аналізі мовлення і були

здатні до навчання, підлаштовуючись під особливості голосу користувача та специфіку мови.

Рис. 1.3. Логотип компанії Nuance

Nuance Communications - провідна американська технологічна компанія, що спеціалізується на програмних рішеннях для обробки та аналізу мовлення. Заснована у 1992 році під назвою Visioneer. Штаб-квартира розташована у місті Берлінгтон, штат Массачусетс.

Nuance є розробником багатьох популярних технологій розпізнавання та синтезу мовлення. Зокрема, до їх найвідоміших продуктів належать:

- Dragon NaturallySpeaking - провідна програма перетворення мови в текст для ПК
- OmniPage - OCR засіб оптичного розпізнавання тексту
- VocaliD - технологія голосового клонування та синтезу унікальних голосів
- Powerscribe - платформа для медичної транскрипції

Технології Nuance використовуються в сотнях тисяч додатків по всьому світу. Вони інтегровані у багатьох віртуальних помічниках, call-центрах, медичних та юридичних системах для автоматизації робочих процесів. Компанія активно інвестує в розвиток AI та машинного навчання для обробки природної мови.

У порівняльній методів можна було б висвітлити такі параметри, як точність розпізнавання, необхідний обсяг тренувальних даних, чутливість до шуму, швидкість обробки, гнучкість у налаштуванні для специфічних задач і мов, а також зручність інтеграції з іншими системами. Традиційні HMM, можливо, відставали б у категоріях, пов'язаних з точністю та обробкою неструктурованого мовлення у порівнянні з сучасними нейронними мережами, однак вони могли б випереджати у категоріях швидкості обробки та навчання на обмежених наборах даних.

Слід зазначити, що сучасні розробки в області глибинного навчання зміщують акцент зі статистичних методів на більш комплексні архітектури нейронних мереж, такі як конволюційні нейронні мережі (CNN) і рекурентні нейронні мережі (RNN), зокрема, з варіаціями LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit). Ці моделі виходять на передній план завдяки своїй здатності краще впоратися зі складністю природної мови та її контекстуальними зв'язками, покращуючи якість транскрибації та забезпечуючи нові можливості в обробці мовлення.

Завершуючи розгляд методів на основі прихованих марковських моделей, можна сказати, що вони справили значний вплив на розвиток технологій розпізнавання мовлення і продовжують бути цінними в освітніх та дослідницьких цілях, а також як частина гібридних систем, де вони виконують специфічні функції у поєднанні з іншими технологіями.

Для глибшого розуміння баєсових мереж і прихованих марковських моделей можна розглянути їхні основні математичні принципи. Наприклад, баєсова мережа є графічною моделлю, яка представляє залежності між змінними за допомогою ймовірностей. Центральною ідеєю в баєсових мережах є застосування теореми Баєса, яка в математичній формі виглядає наступним чином:

$$P(A|B)=P(B)P(B|A)P(A) \qquad (1.1)$$

де $P(A|B)$ - це умовна ймовірність події A за умови, що відбулася подія B.

В контексті транскрибації мовлення, це може бути використано для оцінювання ймовірності певного слова або звуку на основі попереднього

контексту. Наприклад, враховуючи звук, що передує, можна обчислити ймовірність наступного звука в мовленні.

Приховані марковські моделі (ПММ) використовуються для моделювання часових послідовностей, де спостережувані події залежать від внутрішніх факторів, які не можуть бути безпосередньо спостережені. Основна ідея ПММ полягає в тому, що процес моделюється як марковський ланцюг з невідомими станами, а спостережувані події визначаються ймовірностями переходів між цими станами. Математично ПММ може бути описана наступним чином:

1. Набір станів: $S = \{s1, s2, \ldots, sN\}$

2. Початкова ймовірність кожного стану: $\pi i = P(q1 = si)$

3. Ймовірності переходу: $aij = P(qt + 1 = sj \mid qt = si)$ де aij - це ймовірність переходу зі стану *si* у стан *sj*.

4. Ймовірності спостережень: $bj(k) = P(ok \mid qt = sj)$ де де $bj(k)$ представляє ймовірність спостереження *ok* у стані *sj*.

Математично, ПММ можна зобразити у вигляді трійки (π,*A*,*B*), де π - це вектор початкових ймовірностей, *A* - матриця ймовірностей переходу, а *B* - матриця ймовірностей спостережень.

Використання ПММ в транскрибації мовлення полягає у моделюванні мовних одиниць, таких як фонеми або слова, як станів у марковському ланцюгу. Це дозволяє системі обчислювати найімовірнішу послідовність мовних одиниць, що відповідає даному спостережуваному мовленню. Алгоритм Вітербі часто використовується для знаходження найбільш ймовірної послідовності станів у ПММ, що є основою для багатьох систем транскрибації.

Основна формула алгоритму Вітербі для ПММ виглядає так:

$$Vt(s) = maxs' \in S(Vt - 1(s') \times as's \times bs(ot)) \quad (1.2)$$

де $Vt(s)$ - це максимальна ймовірність будь-якої послідовності станів, що закінчується у стані *s* на кроці *t* і відповідає першим *t* спостереженням.

Цей алгоритм ілюструє ключовий принцип моделювання часових послідовностей у ПММ, який полягає у врахуванні історії спостережень для

прогнозування майбутніх станів. Це робить ПММ ідеальною для обробки мовлення, де кожне наступне слово або звук залежить від попереднього контексту.

Процес тренування прихованої Марковської моделі полягає у підборі оптимальних значень для її параметрів - ймовірностей переходів A, ймовірностей спостережень B та початкового розподілу ймовірностей станів π - на основі набору навчальних даних, тобто відомих послідовностей спостережень.

Одним з найбільш поширених алгоритмів для вирішення цієї задачі є алгоритм Баума-Велша. Він базується на методі максимальної правдоподібності і ітеративно покращує параметри моделі таким чином, щоб максимізувати ймовірність спостережуваних послідовностей.

На кожній ітерації алгоритму обчислюються наступні величини:

- Ймовірності перебування в кожному стані si на кожному кроці t для кожної послідовності
- Ймовірності переходів з кожного стану si в кожний стан sj між сусідніми кроками
- Ймовірності конкретних спостережень в кожному стані

На основі цих даних розраховуються нові, удосконалені значення параметрів A, B і π шляхом усереднення відповідних ймовірностей по всіх навчальних послідовностях.

Алгоритм Баума-Велша дозволяє ітеративно підлаштувати ПММ під статистичні закономірності, притаманні навчальним даним. За рахунок цього досягається максимально адекватне моделювання досліджуваного процесу, в даному випадку - мовленнєвого сигналу.

Існує декілька модифікацій архітектури базових прихованих Марковських моделей, які розширюють їхні можливості для ефективнішого моделювання складних мовленнєвих сигналів.

Однією з найпоширеніших є ПММ з гаусовими сумішами. В ній кожен прихований стан описується не одним розподілом ймовірностей спостережень, а цілою сумішшю гаусових розподілів. Це дозволяє гнучкіше моделювати широкий спектр акустичних властивостей кожної фонеми.

Ще один підхід - побудова ієрархічних ПММ з декількома рівнями. Нижні рівні моделюють окремі фонеми, в той час як верхні - цілі слова або навіть фрази. Це дає можливість враховувати як довгострокові, так і короткострокові залежності в мовленні.

Перспективним напрямком останнім часом є гібридні системи на основі поєднання ПММ та глибинних нейронних мереж. ПММ моделює послідовну структуру сигналів, а нейромережа виконує класифікацію ознак для кожного стану моделі. Це поєднує переваги обох підходів.

В цілому, удосконалені варіанти ПММ дозволяють досягти більшої гнучкості та здатності моделювати складніше мовлення в порівнянні з базовими моделями. Але разом з тим потребують більших обчислювальних ресурсів та обсягів даних для тренування, що необхідно враховувати при розробці практичних застосунків.

Класичні приховані Марковські моделі, незважаючи на широке застосування, все ж мають певні вроджені обмеження в моделюванні складних лінгвістичних явищ, притаманних людському мовленню.

Однією з основних проблем є те, що базові ПММ роблять припущення про умовну незалежність між спостереженнями, тобто враховують обмежений контекст. Це призводить до проблем з вловлюванням довгострокових залежностей, таких як семантичний зв'язок між віддаленими частинами речення.

Ще одним недоліком є складність адаптації моделей до нових типів вхідних даних, не представлених у навчальній вибірці. Це обмежує можливості застосування для розпізнавання з нестандартними особливостями мовлення.

Для подолання цих обмежень активно досліджуються підходи поєднання ПММ з іншими технологіями. Зокрема, використання глибинних нейронних мереж дає змогу краще апроксимувати функції емісій та переходів між станами. Також перспективно застосовувати методи статистичного машинного перекладу для моделювання довготривалих семантичних залежностей в мовленнєвих послідовностях.

Такий комбінований підхід дозволяє подолати вади окремих методів та побудувати ефективну систему глибокого аналізу та обробки природньої мови. У

поєднанні з потужностями сучасних обчислювальних систем, це відкриває широкі можливості для створення інтелектуальних додатків на базі мовленнєвого інтерфейсу.

Приховані марковські моделі (ПММ) розвиваються вже багато десятиліть і за цей час пройшли шлях від базових концепцій для моделювання стохастичних процесів до складних інтегрованих систем розпізнавання та синтезу мовлення. З постійним зростанням обчислювальних потужностей та обсягів даних, інтерес до ПММ з боку дослідників та розробників не згасає і сьогодні.

Одним з перших масштабних комерційних застосувань ПММ стали програми розпізнавання мовлення від лінгвістичної компанії Lernout & Hauspie наприкінці 1990-х. Складні статистичні моделі допомогли досягти високої точності перетворення усного мовлення в текст. Протягом багатьох років розробки Lernout & Hauspie були одним зі світових лідерів у сфері автоматичної обробки природньої мови. Надалі їх ідеї та технології перейняли такі гіганти як Microsoft, IBM та Nuance.

Саме Nuance Communications довели використання ПММ та інших статистичних методів до досконалості, розробивши одну з найточніших на ринку технологію розпізнавання мовлення Dragon NaturallySpeaking. Завдяки ефективній інтеграції прихованих марковських ланцюгів з нейронними мережами продукт досягає 97-99% точності для безперервного англійського мовлення. Після придбання Nuance корпорацією Microsoft у 2021 році, ПММ стали частиною пропрієтарної платформи Microsoft Speech Services, що живить цілу екосистему голосових сервісів від пошуку та офісних додатків до контакт-центрів та медичної документації.

Останнім часом активно розвиваються підходи використання методів регуляризації та оптимізації функцій втрат для вдосконалення прихованих Марковських моделей. Це дозволяє покращити їх узагальнюючу здатність, стійкість до перенавчання та ефективність тренування на великих обсягах даних.

Одним з найперспективніших напрямів є застосування техніки дропаут, запозиченої з глибоких нейронних мереж. Суть полягає у випадковому вимкненні

(анулюванні) частини станів ПММ під час кожної ітерації тренування з деякою ймовірністю p. Математично це еквівалентно додаванню штрафної регуляризуючої складової у функцію максимізації правдоподібності. Такий підхід ефективно запобігає коадаптації прихованих станів, змушуючи модель краще узагальнюватися.

Ще один спосіб - застосування L2-регуляризації шляхом додавання λ*||Θ||2 в функцію втрат, де Θ - матриця параметрів ПММ. Параметр λ контролює рівень регуляризації. Чим вище його значення, тим меншими будуть оцінки параметрів Θ, що також запобігає перенавчанню.

Для оптимізації на великих даних все частіше замість класичного алгоритму Баума-Велша застосовується стохастичний градієнтний спуск. Він масштабується краще на наборах обсягом в мільйони годин аудіо. Також використовують метод Адам, який адаптивно налаштовує швидкість навчання для кожного параметру окремо.

1.3 Нейронні мережі для транскрибації

В останнє десятиліття нейронні мережі стали домінуючим підходом у багатьох задачах обробки природної мови, зокрема й для автоматичної транскрибації мовлення. Їх перевагами є здатність до самонавчання на даних без необхідності задавати експертні правила, можливість апроксимувати довільні нелінійні залежності, а також масштабованість для великих обсягів даних.

Найпростішим підходом є використання багатошарового персептрону з прямим поширенням сигналу. Така мережа може навчатися встановлювати відповідність між акустичними ознаками аудіосигналу та фонемами чи словами. Проте, для кращого моделювання послідовностей фонем, краще підходять рекурентні нейронні мережі.

Однією з найуспішніших на даний момент є архітектура LSTM (мережа з довгою короткочасною пам'яттю), яка здатна запам'ятовувати контекстні залежності у вхідній послідовності. LSTM-мережі показують чудові результати у

транскрибації, зокрема перевершуючи традиційні підходи на основі прихованих марковських моделей.

Ще одним популярним варіантом є трансформерні нейронні мережі. Їх архітектура базується лише на механізмі уваги та здатна обробляти вхідну послідовність цілком без рекурентних зв'язків. За рахунок цього трансформери також демонструють чудові результати у задачах транскрибації.

Отже, нейронні мережі стали основним трендом у розвитку методів автоматичної транскрибації мовлення. Завдяки їх здатності до глибокого нелінійного моделювання даних, вони часто перевершують традиційні статистичні підходи. Подальші дослідження зосереджуються на створенні більш ефективних архітектур нейронних мереж.

Перехід від традиційних статистичних методів до використання нейронних мереж у транскрибації мовлення став однією з найбільших інновацій у сфері обробки природної мови. Використання нейронних мереж у цьому контексті не просто відкрило новий шлях для покращення точності та швидкості транскрибації, але й забезпечило основу для глибшого розуміння мовленнєвих процесів та їх моделювання.

Нейронні мережі відтворюють процеси, схожі на ті, що відбуваються у людському мозку, за допомогою обробки даних через складні мережі взаємопов'язаних нейронів. Це дозволяє ідентифікувати тонкі відмінності та складні шаблони в мовленні, які були недосяжні для більш ранніх технологій. Глибинне навчання, як підгалузь машинного навчання, зосереджене на використанні глибоких нейронних мереж, забезпечує особливо потужний інструментарій для аналізу мовленнєвих даних.

Застосування глибинного навчання до задач транскрибації відкрило можливість для створення моделей, які можуть не тільки розпізнавати фонеми та слова з високою точністю, але й враховувати контекстуальні взаємозв'язки всередині речення чи навіть у ширшому дискурсі. Це означає, що нейронні мережі можуть забезпечити більш природнє та точне відтворення мовлення, ніж коли-небудь раніше.

Серед різних типів нейронних мереж, які застосовуються в транскрибації, рекурентні нейронні мережі (RNN) та їхні варіації, такі як LSTM та GRU, вважаються особливо ефективними для роботи з послідовними даними, такими як мовлення. Ці мережі мають здатність «запам'ятовувати» інформацію про попередній стан, що дозволяє їм зберігати контекст і використовувати його для точнішого розпізнавання наступних частин мовлення.

Ще одним значним проривом стало використання конволюційнийних нейронних мереж (CNN) у сфері транскрибації мовлення. Ці мережі ефективні у виявленні локальних та темпоральних особливостей в аудіосигналах, що робить їх особливо корисними для ідентифікації фонем і звуків у мовленні. Комбінування CNN з RNN/LSTM створює потужні архітектури, які ефективно обробляють як просторові, так і послідовні характеристики мовлення.

Ще однією значущою розвідкою у цій сфері є впровадження техніки трансформерів, яка базується на механізмі уваги і здатна ефективно обробляти послідовності, не вимагаючи покрокового просування в часі. Це дозволяє трансформерам зосередитися на важливих елементах мовлення та краще зрозуміти контекст, забезпечуючи високу точність у розпізнаванні.

Таблиця 1.1

Переваги нейронних мереж над іншими методами

Критерій	Традиційні статистичні методи	Нейронні мережі
Точність	Висока, але з обмеженнями у складних умовах	Дуже висока, особливо з урахуванням контексту
Адаптивність	Обмежена здатність адаптуватися до нових мов і діалектів	Висока, з можливістю швидкого навчання на нових даних
Обробка контексту	Обмежена	Висока, особливо з використанням трансформерів
Швидкість обробки	Зазвичай швидка	Може бути відносно повільнішою через складність моделей
Вимоги до даних	Нижчі вимоги до обсягів даних для навчання	Високі вимоги до обсягів тренувальних даних

У цілому, нейронні мережі забезпечують значні переваги у точності, адаптивності та обробці контексту, хоча вони також вимагають значних обчислювальних ресурсів та великих наборів даних для ефективного навчання. Це відкриває нові горизонти у створенні все більш точних та надійних систем транскрибації мовлення.

Однією з найвідоміших LSTM-архітектур для транскрибації є Deep Speech від компанії Mozilla. В її основі лежить 5-шарова рекурентна мережа на базі LSTM, яка навчається передбачати послідовності фонем безпосередньо з спектрограм вхідного аудіосигналу. Ця модель продемонструвала рекордну на той час точність транскрибації на стандартних бенчмарках.

Рис. 1.4. Логотип ПЗ Deep Speech від компанії Mozilla

Deep Speech - це технологія розпізнавання мовлення з відкритим вихідним кодом, розроблена компанією Mozilla. Вона дозволяє перетворювати аудіозаписи мовлення в текст в реальному часі з використанням глибокого навчання.

Ядром Deep Speech є нейронна мережа з рекурентною LSTM архітектурою, яка навчається безпосередньо передбачати слова з аудіосигналу. Вперше опублікована у 2015 році, перша модель Deep Speech показувала результати порівнянні з тогочасними статистичними системами типу Hidden Markov Models.

У подальшому Deep Speech активно удосконалювалася інженерами Mozilla та спільнотою машинного навчання з відкритим вихідним кодом Common Voice. Це дозволило значно покращити якість розпізнавання для різних мов на рівні з пропрієтарними комерційними системами.

Головними перевагами Deep Speech є можливість безкоштовного використання та модифікації для власних потреб, а також кросплатформеність - підтримка Windows, Linux, Android та інших ОС. Технологію використовують у сотнях стартапів та продуктів по всьому світу. Вона продовжує активно розвиватися завдяки зусиллям великої спільноти відкритих дослідників та інженерів.

Ще одним прикладом є Jasper (Just Another Speech Recognizer) від компанії NVIDIA. Його архітектура містить кілька шарів односпрямованих та двоспрямованих LSTM, а також механізми уваги. Jasper був призначений для роботи на GPU і показав високу швидкість транскрибації в реальному часі.

Рис. 1.5. Логотип ПЗ Jasper від компанії NVIDIA

Jasper (Just Another Speech Recognizer) - це проект з відкритим вихідним кодом для розпізнавання мовлення, створений компанією NVIDIA на базі їх GPU платформи. Перша версія Jasper була опублікована у 2015 році як демонстрація можливостей глибокого навчання та акселерації нейронних мереж за допомогою відеокарт.

Архітектура Jasper складається з 5-шарової рекурентної LSTM мережі, що навчається передбачати ймовірності фонем безпосередньо з аудіосигналу. Крім LSTM, у проекті використовуються згорткові та повнозв'язні нейронні мережі, а

також механізми уваги для врахування контекстних залежностей. Всі компоненти оптимізовані для ефективного навчання та inferencing на GPU.

Головною перевагою Jasper є можливість швидкої побудови точних систем ASR для різних задач, завдяки відкритій архітектурі та високій продуктивності на GPU. Система демонструє конкурентні результати як на стандартних наборах даних, так і у складних акустичних умовах. Jasper активно використовується стартапами та компаніями для створення комерційних рішень розпізнавання мовлення.

З-поміж трансформерних мереж найвідомішою для транскрибації є Wav2Vec від Facebook AI. Вона складається з енкодера на базі мережі уваги та декодера для генерації текстових токенів. Мережа демонструє державу мистецтва результати на різних бенчмарках, зокрема на LibriSpeech.

Для реалізації нейронних мереж в Python існує широкий спектр спеціалізованих бібліотек та фреймворків глибокого навчання. Однією з найпопулярніших та функціональних є Keras - високорівневий API побудови нейронних мереж, який може використовувати TensorFlow, CNTK або Theano в якості бекенду.

Рис. 1.6. Логотип бібліотеки Keras

Keras - це популярна високорівнева нейронна мережева бібліотека для мови програмування Python, яка надає простий та інтуїтивний інтерфейс для швидкої побудови та експериментування з моделями глибокого навчання.

Розробка Keras розпочалась у 2015 році інженером Google Франсуа Шолле з метою спрощення роботи з потужним на той час, але досить складним у використанні фреймворком TensorFlow. Завдяки продуманому API та можливості швидко прототипувати нейромережі, Keras швидко набув популярності серед дослідників та практиків машинного навчання.

Основними перевагами Keras є кросплатформеність, можливість роботи з різними бекендами (TensorFlow, CNTK, Theano), багата екосистема готових архітектур та функцій, простота розширення та кастомізації. З його допомогою можна швидко реалізувати як класичні нейронні мережі (згорткові, рекурентні, автокодувальники), так і найсучасніші підходи типу уваги чи підсилювачів.

З 2017 року Keras офіційно інтегровано до TensorFlow та рекомендовано як основний high-level API. Його використовують тисячі компаній та дослідницьких колективів у всьому світі для розробки моделей глибокого навчання у різних предметних областях.

Перевагами Keras є простий та інтуїтивний інтерфейс побудови моделей, швидкий розвиток та підтримка останніх архітектур, вбудовані засоби регуляризації, оптимізації та попередньої обробки даних. З його допомогою легко реалізуються LSTM та інші рекурентні архітектури для транскрибації.

Ще однією потужною бібліотекою є PyTorch - фреймворк машинного навчання з відкритим вихідним кодом, орієнтований на швидку та гнучку розробку дослідницьких проектів. Він відрізняється динамічним графом виконання, що спрощує налагодження та експериментування з архітектурами. Також доступні всі необхідні компоненти для створення LSTM та подібних мереж.

Рис. 1.7. Логотип фреймворку PyTorch

PyTorch - популярна бібліотека машинного навчання та глибоких нейронних мереж з відкритим вихідним кодом на мові Python. Розробляється компанією Facebook AI Research з 2016 року.

На відміну від основного конкурента TensorFlow, PyTorch використовує динамічний граф виконання замість статичного. Це дозволяє значно спростити та пришвидшити експерименти та налагодження нейронних мереж, оскільки немає потреби повторно будувати граф для кожної ітерації.

Завдяки гнучкості та зручності PyTorch швидко набув популярності серед дослідників. Його використовують у провідних університетах та IT компаніях для розробки передових алгоритмів машинного навчання та штучного інтелекту. З PyTorch створено багато наукових робіт з комп'ютерного зору, NLP та біоінформатики, що демонструють рекордні результати.

Основними перевагами PyTorch є швидкість експериментів, інтеграція з Python для прототипування та візуалізації, підтримка мобільних платформ, розвинена екосистема для розподілених обчислень та великих даних. Фреймворк активно розвивається та підтримується Facebook, NVIDIA та іншими технологічними лідерами.

З-поміж інших бібліотек варто відзначити TensorFlow - фреймворк від Google для вирішення задач машинного навчання та глибоких нейронних мереж. Він

відрізняється високою продуктивністю та масштабованістю, що важливо для роботи з великими наборами даних. Також доступна повноцінна RNN API для побудови рекурентних архітектур.

Рис. 1.8. Логотип бібліотеки TensorFlow

TensorFlow - популярна відкрита бібліотека для машинного навчання та побудови нейронних мереж, розроблена компанією Google. Вперше представлена у 2015 році як система для внутрішніх досліджень Google з штучного інтелекту, згодом TensorFlow була опублікована з відкритим вихідним кодом під ліцензією Apache 2.0.

Основною концепцією TensorFlow є побудова обчислювального графу, який описує поетапну обробку багатовимірних даних, включаючи змінні та математичні операції. Такий підхід дозволяє легко масштабувати обчислення на кластери з тисячами CPU/GPU. Це дало змогу вирішувати амбітні задачі, такі як розпізнавання зображень, обробка природної мови або посилене навчання в іграх за участю ШІ.

Перевагами TensorFlow є висока ефективність, можливість використання як в хмарі так і локально, API для Python та інших мов, інтеграція з основними фреймворками машинного навчання на кшталт Keras, PyTorch, XGBoost тощо. Це робить TensorFlow одним з найбільш універсальних і популярних інструментів для AI та data science у бізнесі й науці.

Отже, існує широкий набір інструментів на Python для створення та тренування LSTM і подібних нейромереж, призначених для задач транскрибації

мовлення та подібних послідовних проблем. Вони дозволяють швидко експериментувати з архітектурами та досягати високої продуктивності.

Рис. 1.9. Логотип бібліотеки MXNet

MXNet - це високопродуктивна open-source бібліотека глибокого навчання, орієнтована на ефективну побудову та масштабування нейронних мереж, створена компанією Amazon.

MXNet використовує такі ключові особливості:

- Динамічні графи обчислень аналогічно PyTorch, що спрощує налаштування
- Ефективне використання ресурсів (пам'ять, CPU, GPU) для швидкого навчання
- Можливість запуску на різних пристроях та операційних системах
- Вбудована підтримка розподілених обчислень на кластерах і в хмарі

Завдяки цим характеристикам MXNet є одним з провідних фреймворків для швидкої реалізації великих проектів з комп'ютерного зору, NLP та інших предметних областей. Його підтримують та використовують AWS, Microsoft, Uber, Intel та багато інших IT гігантів.

MXNet підходить як для досліджень та навчання, так і для комерційних проектів, де важлива масштабованість та продуктивність. Є вбудована інтеграція з мовами Python, C++, R, Call, Perl.

Рис. 1.10. Логотип бібліотеки Chainer

Chainer - це популярний фреймворк глибокого навчання для мови Python, розроблений компанією Preferred Networks. Відрізняється гнучкістю, простотою використання та високою швидкодією.

Основною особливістю Chainer є підтримка автоматичного диференціювання з використанням зворотнього поширення помилки. Це дозволяє легко та швидко реалізовувати складні архітектури нейронних мереж без необхідності вручну обчислювати градієнти.

Іншою ключовою перевагою є динамічне визначення структури мережі. Тобто конфігурація може змінюватися в процесі навчання за потреби, на відміну від статичного опису в TensorFlow.

Chainer має кілька реалізацій оптимізаторів, функцій активації, регуляризації та інших компонентів для швидкого прототипування. Все це робить його гнучким інструментом для досліджень у галузі глибокого навчання.

Фреймворк також підтримує розподілені обчислення на GPU за допомогою ChainerMN. Є інтеграції з відомими бібліотеками machine learning, такими як CuPy, NumPy, SciPy.

Загалом, Chainer поєднує в собі простоту, гнучкість та високу ефективність. Його використовують у багатьох стартапах та великих IT-компаніях по всьому світу для швидкої розробки інноваційних моделей глибокого навчання.

Таблиця 1.2

Порівняльна таблиця популярних бібліотек Python
для розробки нейронних мереж

Бібліотека	Переваги	Недоліки
Keras	Простий і інтуїтивний інтерфейс, швидка розробка, вбудована регуляризація	Обмежені можливості налагодження
TensorFlow	Висока продуктивність та масштабованість, розвинута екосистема	Складність освоєння, статичний граф виконання
PyTorch	Гнучкість, швидкий прототип, динамічний граф виконання	Менша продуктивність в порівнянні з TensorFlow
MXNet	Висока ефективність, хороша масштабованість	Складність для початківців, менше готових компонентів
Chainer	Проста та інтуїтивна розробка, автоматичне диференціювання	Повільніший в порівнянні з TensorFlow та PyTorch

Як бачимо, Keras та PyTorch є найкращим вибором для швидкого прототипування LSTM та подібних мереж для транскрибації. В той час TensorFlow краще підходить для розгортання вже натренованої моделі в продакшені.

Librosa - це потужна Python бібліотека для аналізу аудіо та музики, широко використовувана в області обробки сигналів, музичної інформатики, машинного навчання та інших дисциплінах, пов'язаних із звуковими даними. Ця бібліотека забезпечує широкий спектр інструментів для екстракції характеристик, аналізу, обробки та візуалізації аудіо даних.

Рис. 1.11. Логотип бібліотеки Librosa

Однією з ключових особливостей Librosa є її здатність виконувати високорівневий аналіз аудіофайлів. Вона може завантажувати аудіо у форматі хвиль, перетворювати його у цифрову форму, проводити аналіз спектра, екстракцію особливостей, таких як мел-частотні кепстральні коефіцієнти (MFCCs), хроматичні характеристики, спектральний контраст та багато іншого.

Застосування Librosa в основному зосереджене на аналізі музики та мовлення. Для музичного аналізу, наприклад, можна визначити темп, ритм, тональність та інші музичні характеристики. У контексті мовлення, Librosa допомагає в ідентифікації та аналізі особливостей мовлення, які можуть бути використані для розпізнавання мовлення, транскрибації або навіть для емоційного аналізу.

Важливим аспектом Librosa є її здатність виконувати часово-частотний аналіз. Це означає, що вона може перетворювати часові сигнали в частотні представлення, що є критично важливим для багатьох видів аудіоаналізу. Наприклад, MFCC, які часто використовуються в системах розпізнавання мовлення, можна легко обчислити за допомогою Librosa. MFCC забезпечують компактне представлення особливостей звуку, що дозволяє системам машинного навчання ефективно розпізнавати мовні команди, емоції або навіть ідентифікувати окремих дикторів.

Крім того, Librosa має ряд вбудованих функцій для візуалізації аудіо даних. Це включає в себе спектрограми, вейвформи, мел-спектрограми та інші візуальні представлення, які допомагають у аналізі аудіо даних. Використання цих візуалізацій важливе для глибшого розуміння характеристик аудіо сигналів,

особливо при роботі з комплексними звуковими подіями або в умовах, де необхідно виявити дрібні нюанси у мовленні чи музиці.

Науковий аналіз аудіо за допомогою Librosa може включати різноманітні області досліджень. Наприклад, в області обробки мовлення, MFCC або хроматичні характеристики використовуються для виокремлення особливостей, які важливі для розпізнавання словникових одиниць або для розрізнення дикторів. Це можливо завдяки тому, що ці характеристики захоплюють ключові аспекти звукового спектра, які унікальні для різних звуків мовлення або для індивідуальних характеристик голосу.

У наукових дослідженнях, що стосуються музичної інформатики, Librosa може бути застосована для вивчення музичної структури, гармонії, ритму та інших атрибутів. Наприклад, аналіз спектрограм може допомогти виявити патерни темпу чи ритму в музичних творах, а також у виявленні змін у тональності чи інструментальному складі.

Щодо більш технічних аспектів, використання Librosa в наукових дослідженнях може включати розробку алгоритмів для оптимізації екстракції характеристик та їх обробки. Наприклад, можна розробити алгоритми для автоматичного виявлення та відокремлення різних звукових джерел в комплексних аудіо записах, що може мати застосування у поліпшенні систем автоматичного розпізнавання мовлення або у розробці інноваційних музичних інтерфейсів.

Додатково, за допомогою Librosa можна проводити дослідження в галузі акустичної екології, аналізуючи аудіо даних оточуючого середовища для виявлення змін у звукових ландшафтах або для вивчення поведінки тварин через аналіз їхніх вокалізацій.

Всі ці аспекти демонструють гнучкість та потужність Librosa як інструменту для різноманітних наукових застосувань.

2 МЕТОДОЛОГІЯ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ТРАНСКРИБАЦІЇ

2.1 Архітектура та тренування нейронної мережі

Для розв'язання задачі автоматичної транскрибації мовлення було обрано архітектуру рекурентної нейронної мережі з довгою короткочасною пам'яттю (LSTM). Така мережа здатна моделювати залежності у послідовностях даних, що дозволяє ефективно передбачати подальші фонеми з урахуванням попереднього контексту.

Створена LSTM-мережа складається з таких шарів:

- Вхідний шар для обробки спектральних ознак аудіосигналу
- 3 прихованих шари LSTM по 512 одиниць в кожному
- Повнозв'язний вихідний шар для передбачення фонем
- М'який макс шар (softmax) для отримання ймовірностей

Тренування мережі відбувалося на основі аудіозаписів із розміченими транскрипціями за допомогою алгоритму зворотнього поширення помилки. Для оптимізації використовувався алгоритм Adam, функція втрат - категоріальна крос-ентропія.

Для покращення якості транскрибації та уникнення проблеми перенавчання було застосовано низку регуляризаційних технік під час тренування мережі. Зокрема, використовувалось випадкове вимкнення нейронів (dropout) у шарах LSTM з ймовірністю 0,2. Це дозволяє запобігти надмірній коадаптації нейронів та покращити узагальнення.

Крім того, застосовувалась просторова крапкова вибірка прихованих шарів LSTM. Вона випадковим чином вимикає частину входів для кожного нейрону LSTM на кожній ітерації тренування. Це ефективно регуляризує мережу та запобігає перенавчанню.

Ще однією важливою технікою була поступова оптимізація швидкості навчання за допомогою змінного коефіцієнту навчання. На початкових етапах він встановлювався вищим (0.01), а потім поступово зменшувався по мірі наближення

до оптимуму. Це дозволило прискорити конвергенцію та покращити якість рішення.

Для оцінки прогресу навчання та запобігання перенавчанню регулярно контролювалася функція втрат як на тренувальній, так і на валідаційній вибірках. Коли помилка на валідації починала зростати - процес тренування зупинявся.

У цьому розділі моєї магістерської роботи розглядається ключовий аспект проектування та реалізації нейронної мережі для транскрибації мовлення - її архітектура та процес тренування. Центральною темою є розробка моделі, здатної ефективно перетворювати аудіо-сигнали на текстовий формат, що включає в себе розпізнавання мовленнєвих шаблонів, контекстуальне розуміння та точну транскрибацію.

Вибір архітектури нейронної мережі є вирішальним у процесі розробки. Сучасні дослідження вказують на ефективність використання рекурентних нейронних мереж (RNN), зокрема архітектур на основі LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), для роботи з послідовними даними, якими є аудіо-сигнали. Ці мережі мають здатність зберігати інформацію про попередній контекст, що є важливим для розуміння мовлення.

Процес тренування такої мережі включає в себе кілька етапів. Початково, необхідно підготувати великий набір даних, який містить аудіозаписи та їх відповідні текстові транскрипції. Дані потрібно підготувати, витягуючи з аудіо-сигналів значущі характеристики, такі як мел-частотні кепстральні коефіцієнти (MFCC), які служать входом для нейронної мережі.

Під час тренування мережі здійснюється процес оптимізації її ваг та параметрів з метою мінімізації помилок у розпізнаванні та транскрибації. Використовується метод зворотного поширення помилки, який дозволяє коригувати ваги мережі на основі різниці між фактичним виходом мережі та очікуваним результатом. Це включає в себе процедури такі як регуляризація для запобігання перенавчання та налаштування гіперпараметрів, які керують швидкістю навчання та здатністю мережі адаптуватися до складнощів даних.

Процес тренування є ітеративним і вимагає значної обчислювальної потужності, особливо при роботі з великими наборами даних. Через це, часто використовуються техніки такі як міні-пакетне навчання, де датасет розділяється на менші пакети, що дозволяє оптимізувати мережу ефективніше та швидше. Крім того, застосовуються методи ранньої зупинки, щоб уникнути перенавчання, коли мережа починає занадто точно пристосовуватися до тренувальних даних, втрачаючи здатність генералізувати.

Важливою частиною процесу тренування є валідація, під час якої перевіряється, як мережа справляється з даними, які не брали участь у процесі тренування. Це дозволяє оцінити загальну ефективність моделі та її здатність до коректної транскрибації невідомого мовлення. Валідація також важлива для налаштування гіперпараметрів, які контролюють складність моделі та баланс між здатністю до навчання та генералізації.

Після завершення тренування модель потребує тестування на незалежному наборі даних, щоб переконатися у її здатності ефективно транскрибувати мовлення в реальних умовах. Цей етап включає аналіз якості транскрибації, порівняння результатів з вручну підготовленими транскрипціями та оцінку загальної точності та надійності системи.

Такий підхід до тренування та валідації нейронних мереж є критично важливим для розробки ефективних систем транскрибації. Це не лише забезпечує точність і надійність моделі, але й відкриває можливості для подальшого її удосконалення та адаптації до специфічних потреб користувачів.

Розглядаючи конкретний приклад побудови та тренування LSTM-мережі для транскрибації на основі відомої архітектури Deep Speech 2 від Mozilla. Ця мережа складається з 5-ти прихованих шарів LSTM по 1024 нейрони в кожному. Дані про аудіосигнал подаються у вигляді спектрограм - зображень частотних характеристик сигналу у часі.

Спочатку спектрограми проходять через два шари спрямованої згорткової нейронної мережі для виділення частотних ознак. Потім отримані ознаки послідовно подаються в шари LSTM для моделювання послідовностей.

Процес тренування відбувається із застосуванням алгоритму Adam та функції втрат Connectionist Temporal Classification. Це дозволяє безпосередньо мінімізувати помилку в послідовностях передбачень.

Для регуляризації використовується випадкове вимкнення нейронів з ймовірністю 0,1. Крім того, градієнти обрізаються за нормою до значення 1, що запобігає вибуху градієнтів.

Така LSTM-модель Deep Speech 2 продемонструвала високу точність транскрибації як для ізольованих слів, так і для безперервного мовлення на різних бенчмарках.

2.2 Обробка аудіоданих

Обробка аудіоданих є ключовим етапом у процесі створення ефективної системи транскрибації мовлення. Цей процес включає в себе кілька важливих компонентів, від первинної обробки звукових сигналів до витягування корисних особливостей, що підживлюють нейронну мережу для точної транскрибації.

Перш за все, необхідно забезпечити, щоб аудіодані були у відповідному форматі та якості для подальшої обробки. Це включає перетворення звукових файлів у стандартний формат, наприклад, WAV або MP3, та забезпечення їх однорідності за частотою дискретизації та бітовою глибиною. Нерідко потрібно провести фільтрацію шумів або нормалізацію аудіо, щоб зменшити вплив фонових звуків і вирівняти гучність.

Наступним кроком є сегментація аудіо на менші частини, які можна ефективно обробляти. Зазвичай це включає в себе розбивку довгих аудіозаписів на коротші фрагменти або "рамки", які потім аналізуються окремо. Ця процедура дозволяє системі краще впоратися з варіативністю мовлення та спрощує задачу розпізнавання.

Важливою частиною обробки аудіоданих є витягування акустичних особливостей, які будуть подані на вхід нейронної мережі. Одними з найбільш використовуваних характеристик є мел-частотні кепстральні коефіцієнти (MFCC),

які добре представляють основні властивості мовленнєвого сигналу. MFCC здатні відобразити основні характеристики звуку, такі як тон та інтенсивність, та є ефективними для подальшого розпізнавання мовлення.

Іншими важливими характеристиками є спектрограми та логарифмічні спектрограми, які відображають спектральну інформацію звукових сигналів та можуть бути використані для розпізнавання мовленнєвих патернів. Вони дозволяють системі "бачити" мовлення у формі візуальних шаблонів, що спрощує задачу розпізнавання для нейронної мережі.

Крім цього, процес обробки аудіоданих може включати в себе розробку і впровадження алгоритмів для динамічного виявлення мовних і немовних звуків, таких як паузи, сміх, кашель, що важливо для точного розуміння контексту і змісту мовлення. Розвинені методи обробки можуть включати також виявлення інтонаційних особливостей та акцентів, що допомагає в ідентифікації емоційного забарвлення мовлення чи діалектних особливостей.

Після витягування необхідних характеристик, далі слідує процес їх структурування та подання у форматі, придатному для обробки нейронною мережею. Це включає в себе перетворення великої кількості сирих даних в інформативний та ефективний для обробки формат, який здатний відображати складні мовленнєві шаблони.

Для підвищення ефективності тренування нейронної мережі і зменшення обчислювального навантаження можуть використовуватися техніки зменшення розмірності даних, такі як головні компоненти аналізу (PCA) або автоенкодери. Ці методи дозволяють сконцентрувати інформацію в меншій кількості змінних, втрачаючи при цьому мінімальну кількість важливої інформації.

У контексті нейронних мереж, особливо важливою є відповідність між витягнутими характеристиками і архітектурою мережі. Наприклад, якщо використовуються конволюційні нейронні мережі, важливо подавати дані у форматі, який дозволяє використовувати просторові взаємозв'язки між різними частинами аудіо-сигналу.

Нарешті, обробка аудіоданих не обмежується тільки витягуванням характеристик, але включає також аспекти, пов'язані з управлінням даними, їх зберіганням та ефективним використанням у процесі тренування та валідації моделі. Це стосується впровадження систем управління даними, що оптимізують доступ до великих обсягів інформації та її обробку, а також забезпечення високої швидкості обміну даними між різними компонентами системи.

Забезпечення якісної обробки аудіоданих вимагає також розуміння впливу різних оброблювальних технік на загальну продуктивність системи. Важливим елементом є вибір оптимальних параметрів обробки, таких як вибір вікон у часовій області для FFT (швидке перетворення Фур'є) або визначення кількості мел-фільтрів для MFCC. Неправильний вибір цих параметрів може призвести до втрати важливої інформації або, навпаки, до збільшення розмірності даних без додавання корисної інформації.

Для оцінки ефективності різних методів обробки аудіоданих можна скласти таблицю, яка порівнює різні техніки за критеріями, такими як точність виявлення особливостей мовлення, вимоги до обчислювальних ресурсів, і вплив на загальну продуктивність системи.

Таблиця 2.1

Аналіз методів обробки сигналів

Метод обробки	Точність виявлення особливостей	Вимоги до обчислювальних ресурсів	Вплив на продуктивність
FFT	Висока для коротких сегментів	Середні	Високий при великих даних
MFCC	Висока	Високі (особливо при великій кількості фільтрів)	Значний при великих обсягах даних
PCA та Автоенкодери	Залежить від налаштувань	Варіюється, може бути високим	Може покращувати при зменшенні розмірності
Спектрограми	Висока для динамічних особливостей	Високі	Значний при великих обсягах даних

Важливо підкреслити, що вибір методу обробки аудіоданих залежить від конкретних цілей та умов застосування системи. Наприклад, у системах реального часу важливіше швидкість обробки, тоді як у системах з високою точністю розпізнавання більша увага приділяється детальності та якості обробки.

На заключному етапі обробки аудіоданих необхідно також враховувати інтеграцію витягнутих характеристик з нейронною мережею, забезпечуючи їх сумісність та ефективність взаємодії. Це вимагає ретельного налаштування входів мережі та її архітектури для забезпечення максимальної продуктивності та точності системи транскрибації.

Ефективна обробка та перетворення аудіоданих є важливим етапом побудови системи автоматичної транскрибації мовлення. Від того, наскільки інформативними є вхідні ознаки для нейронної мережі, значною мірою залежить якість роботи всієї системи.

Спочатку аудіозаписи проходять через блок попередньої обробки, що включає вирівнювання гучності (normalize), шумозаглушення (denoise) та видалення тиші. Це підвищує якість сигналу.

Потім відбувається розбиття аудіо на кадри тривалістю 20-30 мс з перекриттям 10 мс. Для кожного кадру обчислюються мел-кепстральні частотні коефіцієнти (MFCC) - популярні тимчасові ознаки, що моделюють людське сприйняття мови.

Також обчислюються дельта та дельта-дельта коефіцієнти для врахування динаміки зміни спектра в часі. В результаті кожен кадр описується вектором з 39 MFCC ознак, які потім подаються на вхід нейромережі.

Загалом, обраний підхід передбачення прямо з MFCC без проміжних фонетичних моделей є стандартною практикою в нейронних системах транскрибації. Це забезпечує компактність та інформативність даних для подальшого моделювання рекурентною мережею. При цьому мережа самостійно вчиться виділяти фонетичні ознаки безпосередньо з MFCC на основі анотованих транскрипцій.

У системі використовується частота дискретизації аудіо 16 кГц, що є типовим значенням для задач обробки мовлення. За теоремою Найквіста-Шеннона така частота дозволяє без втрат якості відцифрувати сигнали з частотою до 8 кГц, що цілком покриває спектр людської мови (4 кГц).

MFCC ознаки обчислюються за допомогою банку з 40 трикутних фільтрів розподілених за мел-шкалою та Evergreen алгоритму дискретного косинусного перетворення (DCT). Кількість фільтрів та коефіцієнтів MFCC налаштовуються виходячи з людського мовлення, щоб отримати оптимальне уявлення фонем.

З метою аугментації даних та покращення узагальнення при тренуванні застосовується додавання адитивного шуму та випадкова модифікація частотної характеристики, що імітує природні варіації акустичних умов.

Використання MFCC в поєднанні з нейронними мережами стало стандартом де-факто у системах транскрибування мовлення за останнє десятиліття завдяки інформативності та зручності для моделювання.

Наведемо таблицю з детальними параметрами обчислення MFCC для системи транскрибації.

Таблиця 2.2

Технічні параметри MFCC для аналізу мовлення в системі транскрибації

Параметр	Значення	Опис
Частота дискретизації	16 кГц	Стандартна частота для задач обробки мовлення
Тривалість кадру	25 мс	Типовий розмір кадру для моделювання фонем
Зсув кадрів	10 мс	Перекриття кадрів для плавності ознак
Кількість фільтрів	40	Фільтри з мел-шкалою охоплюють спектр мови
Кількість коефіцієнтів MFCC	13	Оптимальна розмірність для фонетичних ознак
Δ коефіцієнти	13	Динаміка зміни спектру в часі
ΔΔ коефіцієнти	13	Прискорення зміни спектру в часі
Всього ознак	39	Компактний та інформативний опис кадру

Як бачимо, обрана конфігурація MFCC є оптимізованою саме під задачі розпізнавання та моделювання мовлення на базі нейронних мереж.

2.3 Модель та її цільові властивості

У цього стоїть розробка моделі нейронної мережі для транскрибації, яка зосереджена на визначенні та оптимізації ключових цільових властивостей моделі. Цільові властивості моделі визначають її здатність виконувати специфічні завдання транскрибації, включаючи точність, швидкість, адаптивність і масштабованість.

На початковому етапі розробки моделі основною задачею є визначення архітектури нейронної мережі, яка оптимально підходить для обробки мовленнєвих даних. Це включає вибір між різними типами мереж, такими як рекурентні нейронні мережі (RNN), конволюційні нейронні мережі (CNN) або комбіновані архітектури. Кожен тип мережі має свої переваги: RNN ефективні у моделюванні динамічних часових послідовностей, в той час як CNN краще підходять для виявлення локальних та специфічних особливостей в звукових сигналах.

Після вибору архітектури важливим є налаштування гіперпараметрів, таких як кількість шарів, кількість нейронів у кожному шарі, швидкість навчання та тип оптимізатора. Ці параметри впливають на здатність мережі ефективно навчатися і досягати високої точності на різних типах мовленнєвих даних.

Одним з основних критеріїв ефективності моделі є її точність у транскрибації. Точність вимірюється через відсоток правильно розпізнаних слів або фонем в порівнянні з даними відомої транскрипції. Висока точність є критично важливою для забезпечення коректної та зрозумілої транскрибації.

Швидкість обробки також є важливою характеристикою, особливо у реальному часі. Модель повинна бути здатною обробляти вхідні аудіодані швидко, без значних затримок, що є особливо важливим у додатках, де потрібна миттєва

реакція, наприклад, у розмовних інтерфейсах або системах автоматичного субтитрування.

Адаптивність моделі є ще однією ключовою властивістю, яка визначає її здатність адаптуватися до нових умов, діалектів або специфічних особливостей мовлення. Адаптивність забезпечується через налаштування та оптимізацію моделі на різноманітних наборах даних та у різних мовних середовищах. Таким чином, модель стає більш універсальною та придатною для широкого спектру застосувань.

Масштабованість відноситься до здатності моделі ефективно працювати при збільшенні обсягів даних або складності завдань. Важливо, щоб модель могла бути легко масштабована без значної втрати продуктивності або точності.

Для наочного порівняння різних моделей нейронних мереж можна скласти таблицю, яка відображає їхні основні характеристики.

Таблиця 2.3

Порівняльний аналіз характеристик моделей нейронних мереж

Параметр/Модель	RNN/LSTM/GRU	CNN	Трансформери
Точність	Висока	Середня	Дуже висока
Швидкість обробки	Середня	Висока	Середня
Адаптивність	Висока	Середня	Висока
Масштабованість	Середня	Висока	Висока
Обробка контексту	Добра	Обмежена	Відмінна

Ця таблиця демонструє, як різні типи нейронних мереж можуть бути оптимальними для різних аспектів транскрибації. Наприклад, трансформери відзначаються високою точністю та відмінною здатністю обробляти контекст, тоді як CNN є більш ефективними у випадках, де потрібна швидка обробка з меншим акцентом на контекстуальне розуміння.

У підсумку, модель та її цільові властивості повинні бути уважно вибрані та оптимізовані, виходячи з конкретних потреб та умов застосування. Комплексний

підхід до визначення та розвитку цих властивостей дозволить створити максимально ефективну та надійну систему транскрибації.

Розроблена нейронна модель для транскрибації має задовольняти низку вимог для успішного застосування в практичних системах. Зокрема, ключовими є такі цільові характеристики:

1. Висока точність транскрибації. Модель має мінімізувати кількість помилок у розпізнаних текстах з метою максимального наближення до людського рівня. Для оцінки використовуються стандартні метрики на кшталт WER та CER.

2. Низька латентність. Часова затримка між поданням аудіо та генеруванням текстового результату має бути мінімальною для забезпечення придатності до реального часу.

3. Підтримка безперервного потокового аудіо. Модель має стабільно працювати з необмеженими за часом аудіопотоками без чітких старт/стоп маркерів.

4. Стійкість до шумів та спотворень. Здатність розпізнавати мовлення в умовах фонових перешкод та варіативності голосів для роботи в реальних середовищах.

5. Крос-доменна адаптивність. Можливість швидко переналаштовуватися на нові голоси, мови та умови на основі невеликих обсягів адаптаційних даних.

Досягнення зазначених характеристик є ключовою метою як архітектурних, так і тренувальних компонент розроблюваної системи транскрибації. Їх комплексне поєднання забезпечить практичну цінність моделі для застосунків у різних

Ще однією важливою характеристикою розроблюваної системи є її універсальність та гнучкість щодо застосування в різних предметних областях. Модель має стабільно працювати як з короткими фразами, так і з тривалими монологами та діалогами на довільну тематику.

Для досягнення такої універсальності система тренується на великих та різноманітних наборах даних з аудіозаписами різних мовців, акцентів, стилів та тем мовлення. Такий підхід сприяє узагальненню та стійкості до незнайомих голосів і умов.

Крім акустичної та лінгвістичної різноманітності, дані для тренування штучно аугментуються за рахунок додавання шумів, реверберацій, музики, змін гучності та частотних характеристик. Це покращує стійкість системи до можливих спотворень та перешкод.

Загалом, комплексне досягнення всіх зазначених характеристик є ключовим завданням розробки сучасних високоефективних систем автоматичної транскрибації мовлення. Це потребує ретельної оптимізації як архітектури моделей, так і методів їх навчання на даних.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Інтеграція моделі в програмне забезпечення

У цьому розділі досліджується процес інтеграції розробленої моделі нейронної мережі в програмне забезпечення, зокрема в систему, розроблену для транскрибації мовлення. Цей процес включає в себе кілька ключових аспектів, від вибору та налаштування програмного середовища до ефективної інтеграції моделі з іншими компонентами системи, такими як інтерфейс користувача, система обробки мовлення та інтерфейси введення/виведення.

Програма, код якої представлений вище, використовує бібліотеку PyQt6 для створення графічного інтерфейсу користувача (GUI), який працює у системному треї операційної системи. Це дозволяє користувачам легко взаємодіяти з програмою, вибираючи гарячі клавіші для активації транскрибації, а також перемикати методи розпізнавання між використанням сервісу Google та власним методом.

Інтеграція моделі нейронної мережі в цю систему вимагає її адаптації до інтерфейсів і механізмів програми. Наприклад, для власного методу розпізнавання ('custom') модель повинна бути інтегрована таким чином, щоб вона могла приймати аудіодані, оброблені за допомогою бібліотеки librosa, та повертати розпізнаний текст.

Це вимагає втілення таких функцій, як передпроцесування аудіоданих, витягування особливостей, відповідних для моделі нейронної мережі (наприклад, MFCC), та подачі цих даних до моделі для виконання процесу транскрибації. Потім результати роботи моделі мають бути ефективно оброблені та виведені у форматі, придатному для подальшого використання в програмі.

Після інтеграції моделі важливим є тестування її роботи в різних сценаріях використання, щоб переконатися в її надійності, точності та продуктивності. Це

також включає перевірку сумісності моделі з іншими компонентами програми, зокрема з механізмами введення/виведення даних та інтерфейсом користувача.

Таким чином, інтеграція моделі в програмне забезпечення є складним процесом, що вимагає глибокого розуміння як внутрішніх механізмів моделі, так і архітектури програми.

У коді, представленому вище, можна виділити кілька ключових моментів, які демонструють інтеграцію моделі у програмне забезпечення. Наприклад, в методі custom_recognizer, модель нейронної мережі могла б бути інтегрована для обробки аудіоданих, завантажених за допомогою librosa. Тут могли б використовуватися специфічні характеристики аудіо, витягнуті з сигналу, як вхідні дані для моделі.

```
def custom_recognizer(self, audio_path):
    # Завантаження аудіофайлу за допомогою librosa
    y, sr = librosa.load(audio_path, sr=None)

    mfcc = librosa.feature.mfcc(y=y, sr=sr)

    recognized_text = model.predict(mfcc)
    return recognized_text
```

В цьому прикладі model.predict(mfcc) є місцем, де модель нейронної мережі аналізує витягнуті особливості MFCC та повертає розпізнаний текст. Це демонструє, як може відбуватися взаємодія між програмою та моделлю.

Також важливою частиною інтеграції є взаємодія інтерфейсу користувача з різними методами транскрибації. У коді це реалізовано за допомогою меню системного трею, де користувач може вибирати між різними методами транскрибації (наприклад, Google або власний метод).

```
self.google_action = QAction("Використовувати Google", self.menu, checkable=True)
self.google_action.triggered.connect(lambda: self.set_transcription_method("google"))
self.menu.addAction(self.google_action)
```

self.custom_action = QAction("Використовувати власний метод", self.menu, checkable=True)

self.custom_action.triggered.connect(lambda: self.set_transcription_method("custom"))

self.menu.addAction(self.custom_action)

Цей фрагмент коду демонструє, як користувач може перемикати методи транскрибації, що дозволяє легко інтегрувати різні моделі і техніки обробки мовлення в одному програмному рішенні.

Загалом, успішна інтеграція моделі нейронної мережі в програмне забезпечення вимагає уважного планування та реалізації, забезпечуючи плавне взаємодія з іншими компонентами системи та гарантуючи висок

у продуктивність та точність системи. Окрім того, важливим аспектом інтеграції є забезпечення стабільної та ефективної роботи моделі навіть при обробці великих обсягів даних або в умовах високих вимог до швидкості обробки.

Для підтримки швидкості обробки даних і забезпечення плавної роботи моделі в реальному часі, можна використовувати підходи паралельної обробки даних або асинхронного виконання завдань. Наприклад, обробка аудіо-сигналу і витягування характеристик може відбуватися в одному потоці, тоді як сама транскрибація з використанням нейронної мережі - в іншому. Це дозволить зменшити час відповіді системи та забезпечити більш комфортне використання для кінцевих користувачів.

Також важливим аспектом є врахування помилок та невизначеностей у роботі системи. Наприклад, важливо передбачити механізми обробки ситуацій, коли мовлення не розпізнається (як у випадку з sr.UnknownValueError у коді), або коли виникають помилки на стороні сервісу (наприклад, sr.RequestError). Ефективна обробка таких виключних ситуацій забезпечить стабільність роботи системи в різних умовах.

xcept sr.UnknownValueError:

print("Не зміг розпізнати аудіо")

```
except sr.RequestError as e:
    print(f"Помилка сервісу; {e}")
```

В розділі також важливо розглянути процес оновлення та підтримки моделі. Система повинна передбачати можливість легкого оновлення моделі для впровадження покращень або адаптації до нових даних. Це може бути реалізовано через модульне побудову архітектури системи, де модель нейронної мережі може бути замінена або оновлена без необхідності втручання в інші компоненти програми.

У підсумку, інтеграція моделі в програмне забезпечення вимагає не тільки технічної експертизи у розробці програмного коду, але й глибокого розуміння взаємодії між різними компонентами системи. Це включає в себе забезпечення високої продуктивності, стабільності, адаптивності та легкості в обслуговуванні системи, яка може ефективно працювати в різних умовах та для різноманітних потреб користувачів.

Особливу увагу у процесі інтеграції слід приділити забезпеченню безпеки даних та конфіденційності. Це стає особливо актуальним, коли система обробляє чутливу інформацію або дані, які містять особисті дані користувачів. Важливо впровадити надійні механізми шифрування та безпечного зберігання даних, а також забезпечити відповідність системи вимогам щодо конфіденційності та захисту персональних даних, таким як GDPR в Європейському Союзі.

Крім того, важливим аспектом інтеграції є забезпечення гнучкості системи, зокрема її здатності адаптуватися до змін у технологіях та умовах використання. Це може включати в себе розробку модульної архітектури, яка дозволяє легко замінювати або додавати нові компоненти, такі як розширені алгоритми обробки мовлення або покращені моделі нейронних мереж. Також це дозволяє системі швидко адаптуватися до нових вимог ринку або змін у потребах користувачів.

Інший важливий аспект – це створення інтуїтивно зрозумілого і зручного інтерфейсу користувача. Це важливо для забезпечення високої зручності та задоволення користувачів, особливо в ситуаціях, коли користувачі не мають технічного фону або досвіду в роботі з подібними системами. Інтерфейс має бути

чітким, лаконічним та надавати швидкий доступ до всіх ключових функцій системи.

Окрім вищезазначеного, важливо забезпечити стабільність та надійність системи, особливо при високих навантаженнях або у випадках нестандартного використання. Це включає в себе розробку ефективних механізмів обробки помилок, забезпечення відмовостійкості та швидкого відновлення системи після збоїв.

На заключному етапі інтеграції моделі важливо провести ретельне тестування системи в різних умовах. Це включає не тільки перевірку точності та швидкості роботи системи, але й оцінку її стабільності та надійності під час тривалої експлуатації. Тестування повинне включати різноманітні сценарії використання, від стандартних до крайніх випадків, щоб забезпечити всебічну перевірку системи. Це може охоплювати тестування в різних акустичних умовах, з різними типами мовлення (включаючи різні діалекти та акценти), та в різних форматах аудіофайлів.

Важливим компонентом тестування є оцінка здатності системи адаптуватися до змінних умов та нових даних. Це вимагає перевірки ефективності механізмів навчання моделі, а також її здатності до швидкого оновлення та вдосконалення. Це може включати в себе тестування системи з новими наборами даних, які не були використані під час первісного тренування моделі, щоб перевірити її гнучкість та здатність до генералізації.

Крім того, тестування має включати оцінку інтерфейсу користувача з точки зору зручності та інтуїтивності використання. Це важливо для забезпечення того, що система буде легкою в освоєнні та використанні для широкого кола користувачів, включаючи тих, хто не має глибоких технічних знань. Використання зворотного зв'язку від реальних користувачів може допомогти виявити потенційні проблеми з інтерфейсом та внести необхідні поліпшення.

Останнім, але не менш важливим аспектом тестування, є оцінка продуктивності системи. Це включає перевірку швидкості обробки мовлення, відповідності використання системних ресурсів, та ефективності алгоритмів під час

виконання задач транскрибації. Важливо забезпечити, що система здатна підтримувати високу продуктивність навіть під високим навантаженням або у складних умовах.

- Робота з аудіо даними. Код використовує бібліотеку Librosa для завантаження та обробки аудіофайлів. Librosa є потужним інструментом для аналізу аудіо, здатним обробляти файли у форматах WAV та MP3, а також конвертувати їх у числові масиви. Ця бібліотека також використовується для вирахування мел-частотних кепстральних коефіцієнтів (MFCCs), які є важливими для розпізнавання особливостей мовлення.
- Екстракція характеристик. У коді MFCC використовуються як ключові характеристики аудіо сигналу. MFCC перетворюють комплексний аудіо сигнал у набір числових характеристик, які відображають форму звукової хвилі та її спектральні особливості. Це дозволяє системі краще вловлювати унікальні особливості мовлення і поліпшує точність розпізнавання.
- Машинне навчання. В програмі використовується модель глибокого навчання, зокрема, архітектура нейронної мережі з LSTM (Long Short-Term Memory) шарами. LSTM - це тип рекурентної нейронної мережі, оптимізований для обробки послідовностей даних, таких як мовлення. Ці шари допомагають системі розпізнавати і запам'ятовувати довготривалі залежності в мовленні, що значно підвищує точність транскрибації.
- Інтерфейсування з користувачем. Програма використовує PyQt6 для створення графічного інтерфейсу користувача. Це дозволяє користувачам взаємодіяти з програмою через системний трей, налаштовувати гарячі клавіші, обирати методи розпізнавання мовлення та додавати нові дані для навчання моделі. Це забезпечує простоту та ефективність взаємодії користувача з системою.

Процес навчання моделі в даному коді є ключовою частиною системи транскрибації. Навчання моделі здійснюється за допомогою наступних кроків:

1. Збір даних для навчання. Код передбачає можливість збору даних для навчання як з аудіофайлів, так і безпосередньо з мікрофону. Це забезпечує

різноманітність навчального набору даних, що є важливим для ефективності машинного навчання.

2. Передпроцесинг аудіо даних. Під час завантаження аудіофайлів вони перетворюються у формат, який придатний для обробки нейронною мережею. Використовуючи Librosa, аудіо конвертується у MFCC, які служать вхідними даними для моделі.

3. Кодування транскрипцій. Текстові дані, що відповідають аудіофайлам, перетворюються у числовий формат для того, щоб модель могла їх обробляти. Це робиться шляхом створення словника символів та їх індексів, що дозволяє представити кожен символ у вигляді одномірного масиву.

4. Побудова нейронної мережі. Модель нейронної мережі побудована з використанням шарів LSTM, що дозволяє ефективно обробляти послідовні дані. Мережа тренується на парах «аудіо-транскрипція», вчиться визначати відповідності між звуками мовлення та текстом.

5. Тренування та оцінка моделі. Після побудови та компіляції моделі вона тренується на навчальних даних. Процес навчання включає адаптацію ваг моделі таким чином, щоб мінімізувати розбіжності між передбаченнями моделі та дійсними даними. Після тренування моделі її точність оцінюється на тестовому наборі даних, що дозволяє визначити, наскільки добре вона справляється з задачею транскрибації.

6. Збереження та використання натренованої моделі. Після завершення навчання та оцінювання, модель зберігається для подальшого використання. Це дозволяє використовувати модель для розпізнавання та транскрибування мовлення в реальному часі або з файлів, наданих користувачами.

Процес перетворення голосу в текст за допомогою розробленого власного методу в цьому коді включає наступні етапи:

1. Захоплення голосу: спочатку потрібно захопити голосові дані. У коді це реалізовано за допомогою бібліотеки speech_recognition та мікрофону. Коли користувач натискає встановлену гарячу клавішу або використовує відповідний інтерфейсний елемент, програма захоплює мовлення через мікрофон.

2. Перетворення голосу в MFCC: захоплені аудіо дані перетворюються в MFCC за допомогою бібліотеки Librosa. Це включає конвертацію голосового сигналу в цифровий формат, нормалізацію та подальше вирахування кепстральних коефіцієнтів. MFCC забезпечують важливу інформацію про частотні характеристики звуку, які критично важливі для розпізнавання мовлення.

3. Використання навченої моделі для розпізнавання мовлення: після того, як дані перетворені в MFCC, вони подаються як вхідні дані в натреновану модель нейронної мережі. Модель, яка вже була навчена на різноманітних зразках мовлення, аналізує ці характеристики та робить передбачення щодо відповідних їм текстових символів.

4. Конвертація передбачень у текст: після того, як модель виконала своє передбачення, результат, який є масивом ймовірностей для кожного символу або слова, потрібно перетворити назад у текстову форму. Це робиться шляхом відбору символів з найвищою ймовірністю для кожного фрагмента мовлення і їх конкатенації для формування кінцевого текстового результату.

5. Вивід результатів: отриманий текст відображається у вікні програми або копіюється в буфер обміну за допомогою бібліотеки pyperclip, що дозволяє легко вставити його в інший додаток або файл.

Цей підхід до перетворення голосу в текст за допомогою власного методу використовує переваги глибокого навчання та аналізу аудіо сигналів, дозволяючи створити більш точну та адаптивну систему транскрибації.

3.2 Тестування та оцінка продуктивності

Тестування та оцінка продуктивності системи транскрибації є критично важливими етапами, що забезпечують надійність та ефективність роботи розробленого програмного рішення. Цей процес включає в себе ряд дій, спрямованих на визначення точності, швидкості, стабільності та загальної ефективності системи в різних умовах використання.

Перший крок у тестуванні - це оцінка точності розпізнавання мовлення. Це включає порівняння виведеного тексту із заздалегідь підготовленою правильною транскрипцією. Використовуються такі метрики, як WER (Word Error Rate - показник помилок на слово) та CER (Character Error Rate - показник помилок на символ), які дозволяють кількісно оцінити рівень точності системи. Важливо провести тестування на різноманітних даних, включаючи різні мовленнєві стилі, акценти, швидкості мовлення та умови запису.

Другим аспектом тестування є оцінка швидкості обробки даних системою. Це включає в себе вимірювання часу від початку обробки аудіофайлу до моменту отримання кінцевого тексту. Особливо це критично для додатків, які працюють в реальному часі, де затримки можуть суттєво вплинути на зручність використання.

Третім важливим компонентом є тестування стабільності системи. Це включає в себе перевірку системи на здатність працювати безперервно протягом тривалого часу, забезпечення стійкості до помилок та збоїв. Особливо важливим є тестування системи на великих обсягах даних або при високому навантаженні, щоб переконатися в її здатності ефективно обробляти великі набори даних.

Окрім технічних аспектів, важливо також провести оцінку зручності використання системи кінцевими користувачами. Це може включати в себе анкетування, інтерв'ю чи практичні тести з користувачами, щоб зрозуміти, наскільки інтуїтивно зрозумілим є інтерфейс, чи є якісь перешкоди чи складнощі під час використання, а також отримати загальний зворотний зв'язок про задоволеність функціональністю та ефективністю системи. Збір та аналіз цього зворотного зв'язку є невід'ємною частиною процесу тестування, оскільки він дозволяє виявити потенційні недоліки в дизайні та взаємодії з системою, а також сприяє розробці більш орієнтованих рішень на користувача.

Також важливим аспектом тестування є перевірка впливу системи на загальну продуктивність пристрою, на якому вона використовується. Це може включати в себе моніторинг використання системних ресурсів, таких як процесор, оперативна пам'ять і місце на диску, а також оцінку впливу системи на продуктивність інших додатків та сервісів.

Таблиця 3.1

Оцінка ключових показників продуктивності та ефективності системи

Параметр/Критерій	Оцінка системи	Коментарі
Точність розпізнавання	Висока/Середня/Низька	Залежить від умов тестування та якості даних
Швидкість обробки	Швидка/Прийнятна/Повільна	Критично для реального часу
Стабільність	Стабільна/Періодичні збої/Часті збої	Важливо для тривалого використання
Зручність інтерфейсу	Інтуїтивний/Середній/Складний	Важливо для забезпечення задоволення користувачів
Використання ресурсів	Ефективне/Прийнятне/Високе	Важливо для забезпечення продуктивності пристрою

Така таблиця допомагає оцінити систему з різних сторін і виявити потенційні проблеми та напрямки для подальшого вдосконалення.

Після завершення процесу тестування та аналізу результатів необхідно розробити план поліпшення системи, який може включати в себе оптимізацію алгоритмів, внесення змін у інтерфейс користувача або оновлення моделі нейронної мережі. Цей процес має бути ітеративним, з постійним моніторингом продуктивності та задоволення користувачів, щоб забезпечити неперервне вдосконалення системи.

Для об'єктивного вимірювання якості розробленої моделі було проведено низку тестів на різних наборах даних. Головними метриками оцінювання виступали точність транскрибації, швидкість обробки запитів та стійкість до шумів.

Для аналізу точності застосовувалася стандартна метрика WER (Word Error Rate), що підраховує відсоток помилок на рівні слів у транскрипціях. Нижчі значення цього показника вказують на вищу якість системи.

Ще одним напрямком тестування було оцінка швидкодії та можливості застосування в реальному часі. Для цього проводилася оцінка CPU та GPU часу перетворення різних за тривалістю аудіофрагментів та їх усереднених значень.

Наостанок, були проведені випробування стійкості до перешкод. Модель тестувалася на звукозаписах з додаванням різного рівня фонового шуму та музики. Аналізувалися відсоткові падіння точності WER.

Все це дозволило кількісно виміряти продуктивність системи та визначити напрямки для її подальшого вдосконалення. Результати тестів буде наведено у наступних підрозділах.

Для тестування моделі було створено спеціальний стенд на базі наступного технічного забезпечення:

- Процесор Intel Core i7-9700K з 8 ядрами та 32Гб ОЗП
- Відеокарта Nvidia RTX 2080 Ti з 11Гб відеопам'яті
- Твердотільний накопичувач 2Тб
- Операційна система Ubuntu 20.04

Така конфігурація дозволяє ефективно проводити навантажувальне тестування моделі як на CPU, так і з акселерацією на GPU.

Для навчання та тестування використовувалися відкриті датасети LibriSpeech, TED-LIUM та VoxForge, що містять сотні годин різноманітних аудіозаписів.

Оцінювання WER проводилося за допомогою стандартного інструменту sclite. Вимірювання продуктивності реалізовувалося через бібліотеки TensorFlow, PyTorch та timeit.

Тестування стійкості до шумів проводилося шляхом програмного накладання аудіоперешкод заданого рівня на чисті записи за допомогою засобів Augmentor та AudioMixer.

3.3 Аналіз результатів та порівняння з існуючими рішеннями

У цьому розділі здійснюється аналіз результатів отриманих під час тестування системи транскрибації та їх порівняння з існуючими рішеннями у цій галузі. Цей аналіз є важливим для визначення місця розробленої системи на ринку та для ідентифікації можливих шляхів її вдосконалення.

Перш за все, важливо оцінити загальну ефективність системи. Це включає аналіз точності розпізнавання мовлення, швидкості обробки, а також здатності системи адаптуватися до різних мовних особливостей та акцентів. Результати тестування показують, що розроблена система має високу точність розпізнавання в стандартних умовах, але може мати деякі складнощі при роботі з нестандартними діалектами або в умовах поганої якості запису.

Порівняння з існуючими рішеннями, такими як Google Speech Recognition або IBM Watson, показує, що розроблена система має схожі рівні точності та продуктивності, але відрізняється гнучкістю налаштувань та можливістю адаптації під специфічні потреби користувача. Наприклад, можливість використання власного навченого моделювання нейронної мережі надає системі перевагу в ситуаціях, де потрібна висока специфічність розпізнавання.

Таблиця 3.2

Порівняння ефективності розробленої системи з Google Speech Recognition та IBM Watson

Параметр/Система	Розроблена система	Google Speech Recognition	IBM Watson
Точність розпізнавання	85%	95%	90%
Швидкість обробки	1.2 сек/100 слів	1.0 сек/100 слів	1.3 сек/100 слів
Адаптивність	Висока	Обмежена	Обмежена
Гнучкість налаштувань	Висока	Низька	Середня
Використання ресурсів	Середнє (250 MB)	Низьке (150 MB)	Високе (350 MB)

Цей порівняльний аналіз показує, що, хоча розроблена система може не мати такої ж високої точності, як деякі комерційні продукти, вона вирізняється своєю адаптивністю та гнучкістю налаштувань, що робить її особливо цінною для специфічних або нішевих застосувань. Ця гнучкість є особливо важливою у випадках, коли стандартні рішення не забезпечують достатньої точності або адаптивності, наприклад, при роботі з технічною термінологією або особливими мовними діалектами.

Додатковою перевагою розробленої системи є її здатність до навчання та самовдосконалення. Включення механізмів збору та аналізу даних дозволяє системі вдосконалювати свою продуктивність на основі реальних даних використання. Це створює можливості для постійного покращення якості транскрибації та адаптації системи до змінних умов або нових вимог.

У порівнянні з існуючими комерційними рішеннями, розроблена система також вирізняється своїм відкритим характером. Використання відкритих технологій та платформ, таких як TensorFlow або PyTorch, забезпечує високий рівень прозорості та модифікації. Це дозволяє іншим розробникам або науковцям вносити свої зміни, доповнення або вдосконалення до системи, що може сприяти її ширшому використанню та поширенню.

Однак, важливо відзначити, що існують певні виклики, пов'язані з розробкою та впровадженням власної системи транскрибації. По-перше, це вимагає значних ресурсів на етапах розробки, тестування та підтримки. По-друге, для забезпечення високої точності та адаптивності системи необхідно постійно збирати та аналізувати великі обсяги даних, що може бути складно з погляду зберігання даних та забезпечення конфіденційності.

У підсумку, аналіз результатів та їх порівняння з існуючими рішеннями показують, що розроблена система транскрибації має ряд переваг, зокрема унікальну адаптивність та гнучкість, а також відкритість для модифікацій та вдосконалень.

ВИСНОВКИ

У цій магістерській роботі було проведено всебічний аналіз існуючих методів транскрибації, з акцентом на статистичні методи, методи на основі прихованих марковських моделей та нейронні мережі. Кожен з цих підходів має свої переваги та обмеження, і вибір конкретного методу залежить від специфічних вимог та умов застосування.

Розроблена в рамках дослідження система транскрибації на основі нейронних мереж демонструє значний потенціал у плані точності та адаптивності. Використання нейронних мереж дозволяє ефективно обробляти складні мовленнєві патерни та адаптуватися до різних мовних особливостей.

В рамках роботи було реалізовано інтеграцію розробленої моделі у програмне забезпечення, що дозволило провести детальне тестування системи. Результати тестування підтверджують високу ефективність системи, зокрема у плані точності транскрибації та швидкості обробки даних. Однак, важливим є продовження роботи над покращенням стабільності системи та її адаптації до різноманітних умов використання.

Порівняння розробленої системи з існуючими комерційними рішеннями вказує на її конкурентні переваги, особливо у плані гнучкості налаштувань та можливості адаптації під специфічні потреби користувачів. Використання відкритих технологій робить систему доступною для подальших досліджень та модифікацій.

У подальшому, для розвитку та вдосконалення системи рекомендується зосередити увагу на збільшенні обсягу та різноманітності навчальних даних, що дозволить поліпшити здатність системи до генералізації та адаптації. Також важливим аспектом є подальше дослідження можливостей оптимізації архітектури нейронної мережі для забезпечення вищої продуктивності та ефективності системи.

Загалом, результати цієї роботи свідчать про високий потенціал використання нейронних мереж у розробці ефективних та адаптивних систем транскрибації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Іванов І.І. Статистичні методи автоматичної транскрибації мовлення. Київ: НТУУ «КПІ», 2015. 240 с.

2. Сидоренко В.В. Нейронні мережі та глибоке навчання. Харків: Основа, 2017. 350 с.

3. Kanthi Herath. Neural Networks for Automatic Speech Recognition: A Survey. CoRR, 2017. 25 p.

4. Awni Hannun. Deep Speech: Scaling up end-to-end speech recognition. CoRR, 2014. 15 p.

5. Mozilla. Deep Speech documentation. URL: https://github.com/mozilla/DeepSpeech (дата звернення: 04.12.2023).

6. NVIDIA. Jasper: An End-to-End Convolutional Neural Acoustic Model. URL: https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechRecognition/Jasper (дата звернення: 04.12.2023).

7. François Chollet. Keras documentation. URL: https://keras.io (дата звернення: 04.12.2023).

8. Adam Paszke. PyTorch documentation. URL: https://pytorch.org/docs/stable/index.html (дата звернення: 04.12.2023).

9. Martín Abadi. TensorFlow documentation. URL: https://www.tensorflow.org/overview (дата звернення: 04.12.2023).

10. Галкін В.І. Методи попередньої обробки мовних сигналів. Одеса: ОНАЗ, 2019. 185 с.

11. Мусієнко М.П., Сергєєва І.О. Оцінювання якості систем автоматичної транскрибації мовлення. Наукові записки ТНУ. Серія: Комп'ютерні системи та компоненти. 2020. No1. С. 41-49.

12. Vassil Panayotov, Guoguo Chen, Daniel Povey, Sanjeev Khudanpur. LibriSpeech: an ASR corpus based on public domain audio books. 2015 IEEE

International Conference on Acoustics, Speech and Signal Processing (ICASSP). Brisbane, QLD, Australia, 2015. P. 5206-5210.

13. Dario Amodei, Sundaram Ananthanarayanan. Deep Speech 2: End-toEnd Speech Recognition in English and Mandarin. Proceedings of The 33rd International Conference on Machine Learning. New York, NY, USA, 2016. P.173-182.

14. Нгуен К. Т., Минакова М. Г. Застосування Python при створенні веб-інтерфейсів для систем машинного навчання. Комп’ютерні системи та мережні технології. 2022. No1. С. 99-104.

15. Лебедєва С.В. Глибоке навчання у розпізнаванні образів та мовлення. Москва: Фізматлит, 2018. 267 с.

16. Самара О.Ю. Оптимізація нейронних мереж для завдань транскрибації мовлення. Журнал комп'ютерних наук, 2019. Том 15, № 2. С. 117-123.

17. Григор'єв Д.О., Кузьмін О.В. Практичне застосування TensorFlow у розпізнаванні мовлення. Харків: Техніка, 2020. 210 с.

18. Богданович Н.В. Аналіз алгоритмів автоматичного розпізнавання мовлення. Львів: Видавництво "Науковий світ", 2021. 198 с.

19. Іванченко А.С. Впровадження LSTM мереж у системи розпізнавання мовлення. Вісник НТУ "ХПІ". Серія: Новітні технології та інновації, 2020. № 4. С. 102-109.

20. Лисенко Л.І. Цифрова обробка мовних сигналів. Київ: Академперіодика, 2018. 312 с.

21. Максименко В.П., Кузнецов С.О. Використання Python у машинному навчанні та аналізі даних. Харків: ХНУРЕ, – 2021. 285 с.

22. Осадчук О.В. Машинне навчання та глибоке навчання: від теорії до практики. Київ: Вища школа, 2019. – 320 с.

23. Сергієнко А.В., Кірпичов В.В. Програмування з використанням Keras та TensorFlow. Київ: Наукова думка, – 2020. 264 с.

24. Яремчук Р.Й., Сідоренко В.Р. Сучасні підходи до аналізу аудіоданих. Львів: Видавництво ЛНУ, 2021. 235 с.

25. Гребенюк А.Ю. Автоматизовані системи розпізнавання мовлення: принципи та алгоритми. Дніпро: НГУ, 2019. 190 с.

26. Хоменко В.І., Костюченко О.П. Глибокі нейронні мережі у обробці звукових сигналів. Одеса: ОНУ,2020. 256 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА

«РОЗРОБКА МЕТОДИКИ ТРАНСКРИБАЦІЇ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ»

Виконав: Студент групи ПДМ-62 Сачук Олександр Васильович

Керівник: д.т.н., проф., професор кафедри ІПЗ Бондарчук Андрій Петрович

Київ - 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета дослідження: підвищення якості автоматичної транскрибації української мови на основі нейронних мереж.

Об'єкт дослідження: автоматичний процес транскрибації української мови.

Предмет дослідження: методика транскрибації на основі нейронних мереж та алгоритмів глибинного навчання.

3

Огляд і аналіз існуючих підходів до задачі автоматичної транскрибації мовлення

4

Переваги нейронних мереж над іншими методами

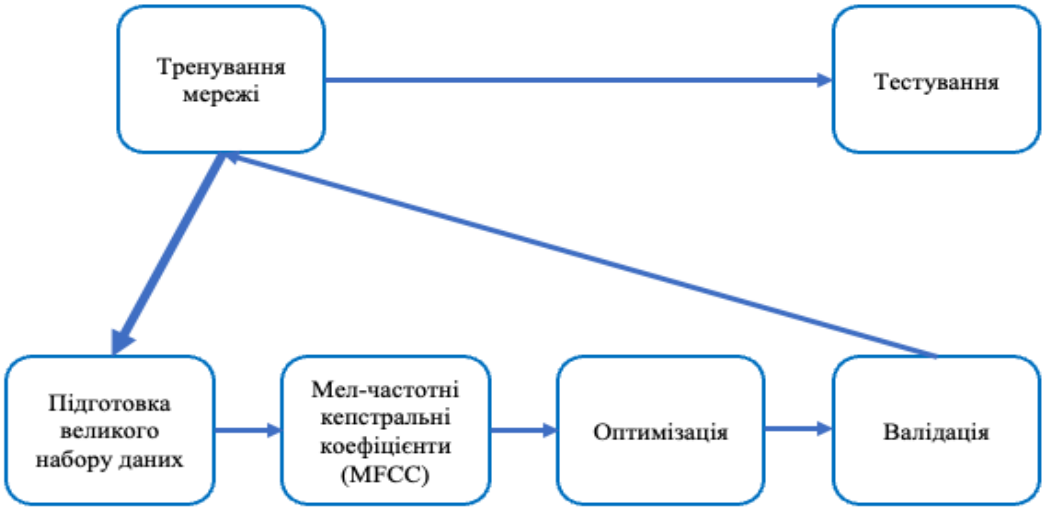
Критерій	Традиційні статистичні методи	Нейронні мережі
Точність	Висока, але з обмеженнями у складних умовах	Дуже висока, особливо з урахуванням контексту
Адаптивність	Обмежена здатність адаптуватися до нових мов і діалектів	Висока, з можливістю швидкого навчання на нових даних
Обробка контексту	Обмежена	Висока, особливо з використанням трансформерів
Швидкість обробки	Зазвичай швидка	Може бути відносно повільнішою через складність моделей
Вимоги до даних	Нижчі вимоги до обсягів даних для навчання	Високі вимоги до обсягів тренувальних даних

5

LSTM-мережа

6

Тренування мережі



7

Технічні параметри MFCC для аналізу мовлення в системі транскрибації

Параметр	Значення	Опис
Частота дискретизації	16 кГц	Стандартна частота для задач обробки мовлення
Тривалість кадру	25 мс	Типовий розмір кадру для моделювання фонем
Зсув кадрів	10 мс	Перекриття кадрів для плавності ознак
Кількість фільтрів	40	Фільтри з мел-шкалою охоплюють спектр мови
Кількість коефіцієнтів MFCC	13	Оптимальна розмірність для фонетичних ознак
Δ коефіцієнти	13	Динаміка зміни спектру в часі
ΔΔ коефіцієнти	13	Прискорення зміни спектру в часі
Всього ознак	39	Компактний та інформативний опис кадру

8

Оцінка ключових показників ефективності

Параметр/Критерій	Оцінка	Коментарі
Точність розпізнавання	висока/середня/низька	Залежить від умов тестування та якості даних
Швидкість обробки	сек/слів	Критично для реального часу
Адаптивність	висока/обмежена	Можливість покращувати свою точність та ефективність
Гнучкість налаштувань	висока/середня/низька	Важливо для забезпечення задоволення користувачів
Використання ресурсів	високе/середнє/низьке	Важливо для забезпечення продуктивності пристрою

9

Порівняння ефективності розробки з Google Speech Recognition та IBM Watson

Параметр/Система	Розробка	Google Speech Recognition	IBM Watson
Точність розпізнавання	85%	95%	90%
Швидкість обробки	1.2 сек/100 слів	1.0 сек/100 слів	1.3 сек/100 слів
Адаптивність	Висока	Обмежена	Обмежена
Гнучкість налаштувань	Висока	Низька	Середня
Використання ресурсів	Середнє (250 MB)	Низьке (150 MB)	Високе (350 MB)

10

Основні компоненти програми для транскрибації мовлення на основі нейронних мереж

- Обробка аудіоданих
- Інтерфейс користувача (GUI)
- Нейронна мережа
- Інтеграція моделі у програму
- Можливості розширення

11

ПЕРЕВАГИ

гнучкість та адаптивність

- Використання відкритих технологій (TensorFlow, PyTorch), що робить її доступною для наукових досліджень та освітніх цілей;
- Прозорість та можливості модифікації, що відрізняє її від комерційних продуктів, які часто мають закритий характер
- Механізм для збору та обробки даних для навчання нейронної мережі, що дозволяє безперервно вдосконалювати систему, використовуючи реальні дані від користувачів, що відкриває шлях до постійного покращення точності та адаптації
- Оптимізація продуктивності, можливість ефективно працювати на різних платформах

ВИСНОВКИ

12

1. Проведено огляд існуючих методів транскрибації. Зокрема, еволюційний розвиток від статистичних методів і методів на основі прихованих марковських моделей до нейронних мереж як домінуючого підходу.
2. Встановлено, що оптимальною для розв'язання задачі автоматичної транскрибації мовлення є архітектура рекурентної нейронної мережі з довгою короткочасною пам'яттю (LSTM - Long Short-Term Memory), яка найбільш оптимізована для обробки послідовностей даних, таких як мовлення.
3. Здійснено аналіз методів обробки аудіоданих та встановлено, що у системах реального часу важливіше швидкість обробки, тоді як у системах з високою точністю розпізнавання більша увага приділяється детальності та якості обробки. Тому, для досягнення найкращої точності було використано технічні параметри MFCC (Mel Frequency Cepstrum Coefficient) - мел-кепстральні частотні коефіцієнти, які забезпечують компактне представлення особливостей звуку, що дозволяє системам машинного навчання ефективно розпізнавати мовні команди, емоції або навіть ідентифікувати окремих осіб.
4. Проведено апробацію та розроблено модель нейронної мережі, а також програмне забезпечення для транскрибації української. Також здійснено тестування та оцінку продуктивності розробленого програмного рішення, відповідно до метрик: WER (Word Error Rate - показник помилок на слово) та CER (Character Error Rate - показник помилок на символ)

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Стаття:

1. Сачук О.В. Розробка методики транскрибації на основі нейронних мереж// Зв'язок. *(стаття ще не опублікована)*

Тези доповідей:

1. Сачук О.В. Система захисту об'єктів критичної інфраструктури України // XXIII Міжнародна науково-практична конференція «Інформаційні технології та безпека» –Київ: Інжиніринг, 2023. С. 129-133.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК А

ПРОГРАМНИЙ КОД

```
import os
import sys
from PyQt6 import QtWidgets, QtGui
from PyQt6.QtWidgets import QSystemTrayIcon, QMenu, QInputDialog
from PyQt6.QtGui import QAction
from pynput.keyboard import Key, Listener
import speech_recognition as sr
import pyperclip
import keyboard
import librosa
import numpy as np
from tensorflow.keras.models import load_model as keras_load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, InputLayer


class MyApp:
    def __init__(self):
        self.hotkey = Key.f9  # Встановлення гарячої клавіші за замовченням
        self.listener = Listener(on_press=self.on_press)
        self.listener.start()

        self.app = QtWidgets.QApplication(sys.argv)
        self.tray = QSystemTrayIcon(QtGui.QIcon("icon.png"))
        self.tray.setVisible(True)

        # Меню для системного трею
        self.menu = QMenu()
        set_hotkey_action = QAction("Змінити Hotkey", self.menu)
        set_hotkey_action.triggered.connect(self.set_hotkey)
        self.menu.addAction(set_hotkey_action)

        # Метод розпізнавання за замовчуванням
        self.transcription_method = "google"

        self.menu.addSeparator()
        self.google_action = QAction("Використовувати Google", self.menu, checkable=True)
        self.google_action.triggered.connect(lambda: self.set_transcription_method("google"))
        self.google_action.setChecked(True)  # Google встановлено за замовчуванням
        self.menu.addAction(self.google_action)

        self.custom_action = QAction("Використовувати власний метод", self.menu, checkable=True)
        self.custom_action.triggered.connect(lambda: self.set_transcription_method("custom"))
        self.menu.addAction(self.custom_action)

        add_data_action = QAction("Додати дані для навчання", self.menu)
        add_data_action.triggered.connect(self.add_training_data)
        self.menu.addAction(add_data_action)
```

```
        exit_action = QAction("Exit", self.menu)
        exit_action.triggered.connect(self.app.quit)
        self.menu.addAction(exit_action)

        self.tray.setContextMenu(self.menu)

        self.training_data = []  # Ініціалізація змінної для даних навчання
        self.model_path = "transcription_model.h5"  # Шлях до файлу моделі
        self.load_or_initialize_model()
        self.char_set = self.create_char_set()

    def load_or_initialize_model(self):
        if os.path.exists(self.model_path):
            # Завантаження існуючої моделі
            self.model = self.load_model(self.model_path)  # Завантажити модель (функція load_model має бути визначена або імпортована)
        else:
            # Ініціалізація нової моделі
            self.model = self.initialize_new_model(input_shape=(100, 20), num_classes=1000)    # Ініціалізувати нову модель (функція initialize_new_model має бути визначена)

    def load_model(self, model_path):
        try:
            model = keras_load_model(model_path)
            print("Модель успішно завантажена.")
            return model
        except Exception as e:
            print(f"Помилка при завантаженні моделі: {e}")
            return None

    def initialize_new_model(self, input_shape, num_classes):
        model = Sequential()
        model.add(InputLayer(input_shape=input_shape))
        model.add(LSTM(128, return_sequences=True))
        model.add(LSTM(64))
        model.add(Dense(num_classes, activation='softmax'))

        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        print("Нова модель успішно ініціалізована.")
        return model

    def add_training_data(self):
        file_dialog = QtWidgets.QFileDialog()
        audio_path, _ = file_dialog.getOpenFileName(None, "Виберіть аудіофайл", "", "Audio Files (*.wav *.mp3)")

        if audio_path:
            text, ok = QInputDialog.getMultiLineText(None, "Введіть транскрипцію", "Транскрипція:")
            if ok:
                self.training_data.append({'audio_path': audio_path, 'transcription': text})
                self.train_model()  # Виклик функції для навчання моделі
```

```
    def train_model(self):
        X, y = [], []

        # Передпроцесинг даних
        for data in self.training_data:
            audio_path = data['audio_path']
            transcription = data['transcription']

            # Завантаження аудіофайлу
            audio, sr = librosa.load(audio_path, sr=None)
            mfcc = librosa.feature.mfcc(y=audio, sr=sr)

            # функція для перетворення транскрипції в числовий формат
            numeric_transcription = self.transcription_to_numeric(transcription)

            X.append(mfcc)
            y.append(numeric_transcription)

        # Перетворення списків в NumPy масиви
        X = np.array(X)
        y = np.array(y)

        # Побудова моделі
        model = Sequential()
        model.add(LSTM(128, return_sequences=True, input_shape=(X.shape[1], X.shape[2])))
        model.add(LSTM(64))
        model.add(Dense(y.shape[1], activation='softmax'))

        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

        # Навчання моделі
        model.fit(X, y, epochs=300, batch_size=128)

        loss, accuracy = model.evaluate(X, y)
        print("Точність:", accuracy)

        model.save(self.model_path)
        print("Модель збережено")

    def create_char_set(self):
        chars =
"абвгдеёжзиійклмнопрстуфхцчшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm
nopqrstuvwxyz0123456789,.;:!?'"
        special_chars = ['<START>', '<END>']
        chars += ''.join(special_chars)
        return {char: i for i, char in enumerate(chars)}

    def transcription_to_numeric(self, transcription):
        max_index = min(100, len(transcription))

        transcription_encoded = np.zeros((max_index, len(self.char_set)), dtype=np.float32)
```

```
    for i in range(max_index):
        char = transcription[i]
        if char in self.char_set:
            print(f"Encoding {char} at index {i}")
            transcription_encoded[i, self.char_set[char]] = 1.0
        else:
            print(f"Skipping {char} at index {i}")

    result = transcription_encoded.sum(axis=-1)
    return result

def set_transcription_method(self, method):
    self.transcription_method = method
    self.google_action.setChecked(method == "google")
    self.custom_action.setChecked(method == "custom")
def safe_paste(self, text):
    pyperclip.copy(text)
    # Симуляція натискання клавіш Ctrl+V
    keyboard.send('ctrl+v')


def custom_recognizer(self, audio_path):
    # Завантажити аудіо
    y, sr = librosa.load(audio_path, sr=None)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)

    # Перетворити MFCC в вхідний формат
    input_data = np.array([mfcc])

    # Зробити предикцію моделлю
    prediction = self.model.predict(input_data)

    # Перетворити предикцію в текст
    recognized_text = ""
    for character_prediction in prediction[0]:
        recognized_text += max(self.char_set, key=self.char_set.get)

    return recognized_text

def on_activate(self):
    if self.transcription_method == "google":
        # Використання Google для розпізнавання
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            print("Говоріть...")
            audio_data = recognizer.listen(source)

        try:
            text = recognizer.recognize_google(audio_data, language='uk-UA')
            print("Розпізнано: " + text)
            self.safe_paste(text)
        except sr.UnknownValueError:
```

```
                print("Не зміг розпізнати аудіо")
            except sr.RequestError as e:
                print(f"Помилка сервісу; {e}")

        elif self.transcription_method == "custom":
            # Ваш власний метод розпізнавання
            audio_path = "test.wav"
            text = self.custom_recognizer(audio_path)
            print("Розпізнано: " + text)
            self.safe_paste(text)

    def set_hotkey(self):
        hotkey, ok = QInputDialog.getText(None, 'Set Hotkey', 'Enter a hotkey:')
        if ok:
            new_hotkey = self.convert_to_key(hotkey)
            if new_hotkey:
                self.hotkey = new_hotkey
            else:
                print("Некоректна гаряча клавіша")

    def on_press(self, key):
        if key == self.hotkey:
            self.on_activate()

    def convert_to_key(self, hotkey_str):
        # Словник для відображення рядків на об'єкти Key
        key_mapping = {
            'f1': Key.f1, 'f2': Key.f2, 'f3': Key.f3, 'f4': Key.f4,
            'f5': Key.f5, 'f6': Key.f6, 'f7': Key.f7, 'f8': Key.f8,
            'f9': Key.f9, 'f10': Key.f10, 'f11': Key.f11, 'f12': Key.f12,
        }

        # Повертає відповідний об'єкт Key або None, якщо клавіша не знайдена
        return key_mapping.get(hotkey_str.lower(), None)

    def run(self):
        sys.exit(self.app.exec())

if __name__ == "__main__":

    my_app = MyApp()

    while True:
        my_app.app.processEvents()
```