КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавр
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА ІНСТРУМЕНТУ ДЛЯ ПРОГРАМУВАННЯ МОВОЮ C++ ІЗ ВИКОРИСТАННЯМ ГОЛОСОВОГО ВВЕДЕННЯ КОРИСТУВАЧА

Виконав студент 4-го курсу
Артем ГЕВОРГЯН

(підпис)

Науковий керівник:
асистент, кандидат фізико-математичних наук
Костянтин ЖЕРЕБ

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«_25_» _____травня_____2022р.,
протокол № _10_
Завідувач кафедри
Олександр ПРОВОТАР

(підпис)

Київ — 2022

РЕФЕРАТ

Обсяг роботи 40 сторінок, 1 таблиця, 3 ілюстрації, 42 джерела посилань.

ГОЛОСОВЕ ВВЕДЕННЯ, C++, ІНСТРУМЕНТИ ПРОГРАМУВАННЯ, ТЕКСТОВИЙ РЕДАКТОР, CLANG, FACEBOOK FLASHLIGHT, EMACS, ПОШУК У ФАЙЛІ, АБСТРАКТНЕ СИНТАКСИЧНЕ ДЕРЕВО, ПІДСВІТКА СИНТАКСИСУ.

Робота досліджує стан готовності галузі розпізнавання голосових команд для побудови користувацьких інтерфейсів для розробників програмного забезпечення. Предметом роботи є розроблена система, яка інтегрує необхідні інструменти для створення цілісного користувацького інтерфейсу.

Метою роботи є розроблення інструменту для програмування, який надає користувачу можливість працювати із програмним кодом за допомогою голосу.

Інструменти розроблення: операційна система GNU/Linux, середовище для програмування мовою C++, Python.

Результатом роботи є програма, яка дозволяє користувачу текстового редактора Emacs виділяти кольором входження у вихідному програмному коді синтаксичних складових. Розроблений додаток сприймає голосові команди користувача.

Робота пропонує спосіб працювати із програмним кодом використовуючи його синтаксичне представлення. Було здійснено крок у напрямку написання інструментів для роботи з мовою програмування, які працюють із стандартизованими поняттями мови програмування C++.

Подальший розвиток може бути здійснений у напрямку покращення роботи голосового інтерфейсу, підтримки модифікуючих операцій над вихідним кодом в розробленому фреймворку та виявленні сценаріїв для нових команд для роботи із програмним кодом.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ASR – Automatic Speech Recognition, автоматичне розпізнавання мовлення;

DSL – Domain Specific Language, спеціалізована мова програмування;

GMM– Gaussian Mixture Model;

HMM– Hidden Markov Model; прихована марковська модель, ПММ;

IDE – Integrated Development Environment, інтегроване середовище програмування;

JIT – Just-in-time (compilation), динамічна трансляція;

LLVM IR – LLVM Intermediate Representation, проміжне представлення програмного коду в проекті LLVM у простій RISC-формі;

LSP – Language Server Protocol, протокол взаємодії інструменту роботи з програмним кодом та серверу, який підтримує базовий набір операцій для роботи із програмним кодом;

NLP – Natural Language Processing, обробка природної мови;

RISC – Reduced Instruction Set Computer, парадигма обчислень, яка надає спрощену архітектуру процесора, порівняно з іншими архітектурами (x86, arm);

RSI – Repetitive Strain Injury, запалення нервових закінчень, спричинене неергономічною роботою із клавіатурою;

TU – Translation Unit, одиниця компіляції;

VUI – Voice User Interface, голосовий інтерфейс користувача;

БНФ – Бекуса-Наура форма;

ОС – операційна система.

ВСТУП

Оцінка сучасного стану об’єкту розробки. Ми бачимо тенденції застосування досягнень штучного інтелекту у роботі з натуральними мовами: GPT-3 [1], BERT [2]. Машинний переклад став рушієм прогресу у цій галузі. Для успішного застосування методів машинного перекладу із мовою користувача необхідно представити її у зручному для машини вигляді, тобто розпізнати.

Мовлення можна декодувати за допомогою ЕЕГ і схожих методів [3], але такі методи поки що тільки розробляються. Наразі стало можливим і доступним розпізнавання мови через звукові сигнали. Мова є єдиним способом безконтактно працювати із машиною так, щоб передати стисло значне когнітивне навантаження.

Компанії збирають голосові дані [4] та розробляють специфікацію для голосового інтерфейсу у браузерах [5].

Одним цікавим застосуванням досягнень у роботі із голосовим введенням може бути використання голосу користувача для виконання повсякденних задач.

Ця робота полягає на результати в галузі розпізнавання мови, програмні рішення, які знаходяться у відкритому доступі.

Актуальність роботи та підстави для її виконання. На сьогодні голосові інтерфейси мобільних пристроїв надають можливості для користування пошуком в Інтернеті (Siri, Google Assistant, Microsoft Cortana, Amazon Alexa/Echo) та деякими іншими функціями, що традиційно були доступні за допомогою сенсорних взаємодій користувача і пристрою. Актуальність застосування цієї системи може бути зумовлена полегшенням повсякденної роботи людей, які хворіють на RSI або не мають можливості користуватися кінцівками для традиційного введення.

Мета й завдання роботи. Мета роботи полягає в створенні інструменту для роботи із програмним кодом, який сприймає голос користувача. Для досягнення мети було поставлено наступні завдання:

– дослідження відомих голосових інтерфейсів – їх функціональних характеристик та використовуваних в них технологій;

– вирішення, які функціональні характеристики можуть бути надані користувачу у рамках роботи;

– планування архітектури додатку – розміщення функціоналу по компонентах;

– розробка компонентів.

Об'єкт і методи розробки. Об'єктом розробки є мод для текстового редактора Emacs, а також набір виконуваних компонентів, які використовуються модом. Було використано наступні програмні засоби для виконання роботи:

– Emacs, середовище для програмування мовою Emacs Lisp;

– середовище Linux для програмування мовою C++;

– інструменти для компіляції Clang, GCC;

– бібліотеку та допоміжні ресурси від Facebook Flashlight для роботи із розпізнаванням мовлення людини.

– ALSA, набір утиліт для роботи із звуковою картою на робочій машині, де відбувається розгортання системи [14].

Можливі сфери застосування. Виконана робота може бути наступним кроком у поширенні голосових інтерфейсів серед кінцевих користувачів. Ця робота може бути використана для надання можливостей людям, які не здатні чи не вважають за доцільне користуватися кінцівками для введення інформації у комп'ютер.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМНОЇ РОБОТИ ЗІ ЗВУКОМ

1.1 Формат WAV. Представлення аудіо хвилі

Існує формат WAV (англ. waveform audio format) кодування аудіо файлу. Такий файл містить вимірювання амплітуди звукової хвилі у часі (див. рис. 1). Зазвичай вимірюють напругу струму, який відображає акустичні коливання джерела звуку. Існує два параметри для вимірювання: частота виміру (sampling rate, Гц), точність виміру (скільки інформації може закодувати вимір; наприклад, за допомогою двох байт можна закодувати значення від -32768 до 32767). WAV-файл може містити інформацію про різну кількість джерел звуку (наприклад, монозапис або стереозапис).

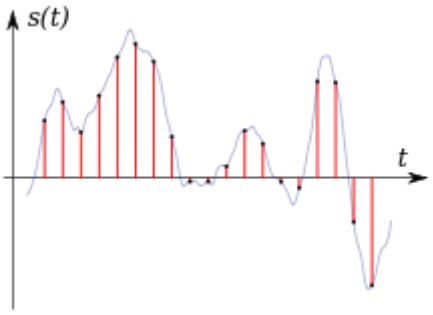

Рисунок 1 – Вміст WAV-файлу

Значна амплітуда звукової хвилі означає велику гучність.

1.2 Способи візуального представлення звуку

Деякі підходи до розпізнавання мовлення, які розглянуті у розділі 2, спираються на візуальне представлення фрагменту звуку, яке будується за допомогою перетворення Фур'є. Методи, які дозволяють побудувати таке представлення, доступні у ряді бібліотек, наприклад, librosa [13]. Згорткові

нейронні мережі працюють із таким візуальним представленням. Тобто для роботи згорткових нейронних мереж попередня обробка може включати в себе наступні кроки: визначити вікно, яке має змісти для даної задачі (наприклад, в задачі розпізнавання ключових слів це може бути 100 мс [35]), побудувати за ними спектрограми. У роботі [35] розглянуто можливі способи графічного представлення звукового сигналу на невеликому проміжку часу..

1.2.3 Спектрограми

Зазвичай працюють із представленням, яке називають спектрограма. Вхідний аудіофайл у форматі WAV або іншому (OGG, MP3 або ін.) обробляється за допомогою перетворення Фур'є таким чином, щоб отримати спектри частот, розкладені у проміжки часу.

1.2.4 Мел-частотні коефіцієнти та септрограми

Для наочного представлення звукової хвилі протягом певного короткого проміжку часу перетворюють результат попереднього кроку – спектрограму – у Mel-шкалу [35]. Після цього кроку можливо використати згорткові нейронні мережі для роботи з візуальним представленням вихідної аудіо хвилі на заданому проміжку часу.

1.3 Акустична одиниця

Поняття акустичної одиниці (utterance) є невід'ємною складовою усіх систем розпізнавання мови, починаючи із систем, які не використовували глибоке машинне навчання (див. розділ 3.1). Воно означає неподільну одиницю мовлення, задану в контексті (фонема або слово, наприклад).

РОЗДІЛ 2 ВИКОРИСТАННЯ СИНТАКСИЧНОГО ПРЕДСТАВЛЕННЯ ПРОГРАМИ ДЛЯ ПОБУДОВИ ІНСТРУМЕНТІВ РОБОТИ З ПРОГРАМНИМ КОДОМ

Багато текстових редакторів містять функцію пошуку входження текстового рядку у тестовому файлі. Такий функціонал містять навіть прості текстові редактори, наприклад Notepad у ОС Windows. Більш традиційні редактори, такі як Vim та Emacs, теж містять функціонал пошуку текстової стрічки у файлі. Трохи більш потужним є інструмент пошуку за регулярним виразом. Такий пошук дозволяє шукати вирази, які можна описати детермінованим скінченним автоматом. Але детермінований скінченний автомат не дозволяє шукати вирази, які визначаються абстрактними машинами, які потребують наявності стекової пам’яті.

Наприклад, вираз, який визначає правильну послідовність дужок (ППД), вимагає зліченної кількості станів для розпізнавання. Тому неможливо використати регулярний вираз для побудови інструменту для пошуку конструкцій мови програмування C++, оскільки вони визначаються контекстно-незалежними граматиками [41]. Для роботи із C++ існує декілька інструментів – від Майкрософт, GNU та Clang/LLVM. Проект Clang надає драйвер (інструмент для виконання необхідних задач для перетворення вхідного файлу в виконуваний файл), який сумісний із інструментом cl від Майкрософт, також gcc від GNU. Ці інструменти працюють із внутрішнім представленням програми, але не всі надають можливості для програмування із внутрішнім представленням, яке зазвичай називають AST.

РОЗДІЛ 3 ПОСТАНОВКА ЗАДАЧІ ASR І ТЕОРЕТИЧНІ ОСНОВИ ЇЇ РОЗВ'ЯЗАННЯ.

Нижче поставлена задача ASR та розглянуто підходи до її розв'язання.

3.1 Складові системи ASR

Загалом для розв'язання задачі отримання транскрипції аудіо файлу використовують моделі машинного навчання, які складаються із компонентів, описаних нижче. Деякі складові застарілі (див. розділ 3.3, 3.4).

3.1.1 Фонетична модель

Під фонемою розуміються логічно окремі елементи мови. Моделювання фонем було невід'ємною частиною процесу розпізнавання мови до початку появи так званих end-to-end підходів. [7]. Відповідна модель призначена для розпізнавання таких одиниць.

3.1.2 Акустична модель

Акустична модель створена для розпізнавання символів із вхідних даних, які представлять аудіо хвилю.

3.1.3 Мовна модель

Мовна модель використовується для формування змістовних конструкцій, характерних для мови, яку система намагається розпізнати. Така система може бути статистичною і враховувати імовірність послідовностей слів для виведення найбільш імовірної послідовності слів. Статистична модель зазвичай складається із n-грам, де n – це параметр. Така модель розраховує імовірність знаходження в

тексті послідовності із n слів.

Мовна модель (language model) використовується в деяких випадках явно (як у класифікаторі, який тренується за методом зниження оцінки кросс-ентропії і за критерієм CTC [11] [15]), і інших випадках неявно (у випадку RNN-T [15]). Використовуються також мовні моделі, побудовані на нейронних мережах [16]. У цій роботі використана n-грамна модель KenLM [17].

3.3 Традиційні підходи до вирішення задачі ASR

Моделі глибокого навчання стали використовувати у задачі ASR в 2013-му році. Тоді стало можливим позбутися від HMM/GMM (Hidden Markov Model, Gaussian Mixture Model [6]) моделей. Система Dragon Voice використовувала саме такі підходи до 2016 року, коли команда розробників почала працювати над заміною деяких компонентів у своєму продукті на більш сучасні.

Традиційно використовували наступні складові: модель мови, акустична модель, фонетична модель. Але зараз використовують лише акустичну та мовні моделі. Причому мовна модель буває явною, як у декодері при використанні CTC-підходу, так і неявною (у випадку RNN-T модель може запам’ятовувати видані результати, відповідно згідно з ними і визначати умовні ймовірності послідовності символів з алфавіту).

Компанії, такі як Yandex у 2013, використовували підходи, які не можна назвати end-to-end підходами [8], але із появою досліджень [9] стало зрозуміло, що використання глибоких нейронних мереж (DNN) може покращити роботу акустичних моделей у таких системах. З цього почався процес переходу до так званого end-to-end підходу до вирішення задачі розпізнавання мови. У 2014 році відбувся перехід так званого end-to-end підходу, у рамках якого відмовилися від фонетичного моделювання. Першою спробою створити таку спрощену модель розпізнавання мови була робота [10], яка базувалася на техніці CTC [11].

Ранні системи використовували так звані MFCC-критерії для роботи зі звуком. Наразі стало можливим тренувати моделі цілком на даних про форму

звукової (акустичної) форми.

Системи, що були побудовані раніше, полягають на розпізнавання фонем, а не окремих графем. Ранні системи також виглядали схожими на FNN [12].

Також вони використовували HMM (Hidden Markov models), GMM (Gaussian Mixture models) [12], також на техніку FST (finite state transduction).

Першим застосуванням нейронних мереж на глибокого навчання в області ASR стало використання їх в акустичних моделях.

3.4 Сучасні підходи до вирішення задачі ASR

В цілому задача розпізнавання мовлення за заданим аудіо файлом зводиться до розпізнавання одиниць мовлення, які розгортаються послідовно в часі. Для розпізнавання базових одиниць використовують перетворення Фур'є для побудови візуального зображення розкладеної енергії сигналу на певному проміжку у частоти. Після того використовують згорткові нейронні мережі, щоб класифікувати базові одиниці мови. Раніше використовували більш складну фонетичну модель, яка вимагала більш детального опрацювання фонетичної складової і класифікації фонетичних одиниць у ній (дифтони, трифони). Зараз прийнято розпізнавати базові одиниці мови – тобто літери напряму. Послідовність літер накладає необхідність працювати з рекурентними нейронними мережами на наступному етапі класифікації – для побудови коректного представлення мовлення зважаючи на відсутність чітких обмежень на довжину в часі, яку займають в аудіофайлі базові одиниці мовлення (можна вимовити літеру швидко, можна повільніше, і системи ASR намагаються працювати за різних швидкостей вимови тексту, який розпізнають). Задача ASR споріднена до задачі розпізнавання рукопису, оскільки обидві задачі характерні тим, що відсутня можливість побудувати розмітку даних. Після отримання представлення послідовності вхідних символів з'являється необхідність у мовній моделі, яка корегує отриману на вході послідовність так, щоб вона відповідала нормам і граматиці мови тексту, що розпізнається. Такі моделі зазвичай статистичні [17] або побудовані із

використанням можливостей NLP – такі підходи побудовані на моделях BERT, Transformer.

3.5 Відомості про вибрані моделі машинного навчання

Для розуміння технічної літератури потрібно розуміти деякі базові моделі машинного навчання, які розглянуті нижче.

3.5.1 Різниця між RNN та Transformer

Принципова різниця між підходами, що базуються на рекурентних нейронних мережах (RNN), та Transformer [18], полягає в тому, що, по-перше, механізм позиційного кодування в останньому дозволяє паралелити тренування за рахунок уникнення необхідності чекати, поки кожний об’єкт у послідовності буде опрацьований, як це робиться у рекурентному енкодері [24], по-друге, існує верхня межа на розмір елементів, які подаються на вхід моделі Transformer, тобто вони працюють не з послідовністю, а з вектор із розміром, що гарантовано менший за деяку фіксовану константу.

У [19] було вперше написано про RNN-T, тобто про спосіб генерувати послідовності довільних символів за заданою послідовністю довільних символів.

3.5.2 Архітектура Seq2Seq

Архітектура Seq2Seq передбачає наявність компоненту, який у цій роботі називається енкодером, та компоненту, який тут названо декодером (з англ. “encoder” та “decoder” відповідно). Причина в тому, що згідно до парадигми Seq2Seq, спочатку енкодер будує внутрішнє представлення послідовності, яку йому подано на вхід; після цього декодер працює із цим представленням, генеруючи послідовність символів (у широкому розумінні символів як елементів вихідного алфавіту), які мають бути відповіддю на задачу. Виглядає цей процес

схематично як на рисунку 2:

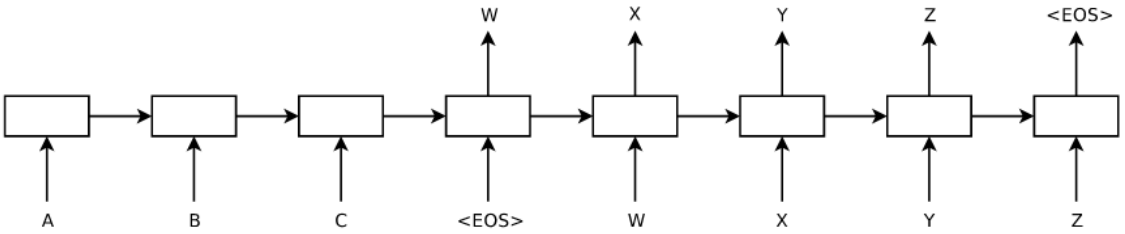

Рисунок 2 – Архітектура Seq2Seq

Спочатку на вхід до енкодеру подається послідовність символів (у цьому випадку A, B, C), після цього надається токен <EOS>, що означає завершення вхідної послідовності. Після цього починається генерація послідовності вихідних символів.

Серед різних моделей, призначених для вирішення задачі seq2seq, виділяють [20] ті, що побудовані на використанні CTC; механізму уваги; RNN-T [19]; також RNN-T із використанням механізму уваги.

3.5.3 Рекурентні нейронні мережі

Рекурентні нейронні мережі представляють із себе один чи декілька слоїв, тобто зазвичай такі системи не роблять глибокими. Причиною цього є так званий затухаючий градієнт під час тренування. Такі нейронні мережі Тьюрінг-повні [21], тобто дозволяють вирішувати довільну задачу, яку можна вирішити на сучасному процесорі, теоретично. Звичайні рекурентні нейронні мережі схожі на нейронні мережі прямого поширення. Їх зображують за допомогою розгортання їх у часі. Складні системи, побудовані на рекурентних нейронних мережах, важко тренувати.

Таким чином, рекурентні нейронні мережі зберігають у собі “відбиток” елементів послідовності, які було пропущено через неї. Елементи, які були відносно давно у послідовності, мають менший і менший ефект на прихований

стан рекурентної мережі (в даному випадку вважаємо, що кожен шар рекурентної нейронної мережі можна назвати самою рекурентною мережею).

Зауважимо, що рекурентні нейронні мережі теж свого роду містять пам’ять та за визначенням відображають зміну вхідних символів на своїх внутрішніх станах. Проте, існує декілька проблем із таким підходом. По-перше, проблема затухаючого градієнту унеможливлює ефективне тренування. По-друге, порівняно із механізмами уваги, така нейронна мережа не може вибірково зважати на стани із більшою чи меншою мірою. Рекурентні нейронні мережі, які вміють вибірково запам’ятовувати чи підсилювати певний сигнал (LSTM [22], GRU [23]), більш здатні до ефективного аналізу послідовності, проте ціною більш складного тренування, знову ж таки через проблему затухаючого градієнту. Результатом є те, що зазвичай рекурентні нейронні мережі не роблять дуже глибокими (достатньо до 4-х слоїв зазвичай), інакше тренувати таку модель буде важко.

3.5.4 Connectionist Temporal Classification

CTC (Connectionist Temporal Classification) був описаний у роботі [11] як необхідність вирішувати задачу, коли існує необхідність натренувати модель класифікації фрагментів, що розміщені одне за одним, але можуть мати довільні розміри. Першопочатково використовувався цей підхід у задачі розпізнавання рукописів: на фотографіях рукописів робили розпізнавання за допомогою згорткових мереж, після цього використовували архітектуру Seq2Seq (пізніше вона була так названа), яка складалася з енкодеру та декодеру. CTC-моделі тренують на мінімізацію кросс-ентропії (так званий log-loss), як і інші моделі, результатом роботи яких є імовірнісний розподіл (зазвичай це означає використання функції softmax), таким чином виконуючи задачу класифікації.

Іншими словами, такі моделі мають мету надавати правильний імовірнісний розподіл, і мета тренування полягає в тому, щоб мінімізувати похибку між спостережуваним імовірнісним розподілом і розподілом моделі на вибірці для тренування.

Неможливо зрозуміти наперед, де саме починається і закінчуються вимовляння однієї акустичної одиниці, тому з'явився підхід Connectionist Temporal Classification [11]. Цей підхід до вирішення задачі класифікації на даних, що представляють послідовність, але не можуть бути розділені на конкретні фрагменти в силу довільної протяжності кожного елемента послідовності, вперше став у нагоді у вирішенні задачі розпізнавання рукописного тексту. У роботі [28] використано згорткові нейронні мережі для виділення особливостей рукописних літер, які належить розпізнати, та після цього використовується рекурентна нейронна мережа, що може розпізнавати послідовність символів.

3.5.5 Механізм уваги

Існують також механізми уваги, який набув поширення у задачі Seq2Seq і називається Seq2Seq з доданим механізмом уваги. Механізм уваги може бути реалізовано різними способами. Першопочатково цей механізм був запропонований як нормалізований за допомогою softmax розподіл, який можна було трактувати як імовірнісний. Задача цього розподілу була вплинути покомпонентно на вектор тієї ж розмірності [18]. Іншим способом створити схожий за призначенням механізм стала робота [25], яка ввела адитивну увагу.

У рамках роботи із механізмом уваги приховані стани останнього слою рекурентної нейронної мережі мають бути запам'ятовані і використані на етапі генерації послідовності вихідних символів на кожному етапі видачі нового символа рекурентним декодером.

Механізм уваги використовується у розпізнаванні мови [37] і це один із відомих інструментів для роботи із розпізнаванням акустичних елементів мови. Існують також механізми уваги, які значно краще адаптовані до розпізнавання мови [26], вони дозволяють розпізнавати мову в режимі реального часу (див. розділ 3.6). Модель під назвою Transformer, що була запропонована у [18], містить так званий механізм рефлексивної уваги (self-attention).

3.5.6 Transformer

Окремо варто звернути увагу на модель Transformer, яка знайшла використання також в суміжній галузі, в роботі із натуральними мовами. Ця модель може працювати із послідовністю, яка не перевищує деяку задану наперед константу, на якій модель було натреновано. У противагу до цього, рекурентні нейронні мережі здатні опрацьовувати нескінченну послідовність символів і зберігати “відбиток” цієї послідовності у пам’яті.

3.5.7 RNN-T

RNN-T з’явилися у роботі [27, 15, 28, 19] як розв’язок задачі sequence transduction.

Застосування двосторонніх рекурентних нейронних мереж робить неможливим роботу в режимі онлайн моделей, тренованих за критерієм CTC, а також RNN-T. Щоб працювати в режимі реального часу, їх енкодерам необхідно не мати змоги дивитися на майбутні дані [20]. За роботи в режимі, коли доступна вся послідовність одразу, двосторонні рекурентні нейронні мережі використовувати можна.

Існує певна проблема із RNN-T моделями: їх важко тренувати на практиці, тобто вони можуть стати непридатними для практичних потреб. Тим не менше, ця область розвивається і є результати.

3.4.8 Time-Depth Separable Convolution

TDS (Time-Depth Separable Convolution) показав найкращі результати у 2019-му році, реалізований у Flashlight та описаний у роботі [33]. Він базується на згорткових нейронних мережах, які застосовуються до задачі розпізнавання мови. Згорткові нейронні мережі корисні для того, щоб розпізнати характерні для тих чи інших акустичних одиниць деталі. За аналогією із розпізнаванням об’єктів,

згорткові нейронні мережі моделюють ті особливості об’єктів, які треба класифікувати, на які мозок звертає увагу при розпізнаванні.

3.6 Розпізнавання в режимі реального часу

У цій роботі не розглядаємо задачу розпізнавання мови в режимі реального часу. Зазвичай такі модели мають балансувати між метрикою WER (word error rate), що вимірює точність розпізнавання моделі, та довжиною затримки акустичної моделі до того, як вона здійснить класифікацію, яка потрібна для більшої впевненості.

Якщо поставлена задача розпізнавання в режимі онлайн, модель має розпізнавати паузи між розпізнуваними одиницями мови (це можуть бути слова або послідовності слів, в залежності від потреби додатку, у даному випадку це слова). У такому випадку необхідно розпізнавати, коли видати так званий end-of-sentence токен з акустичної моделі. Цей токен може бути виданий, наприклад, за умови наявності тиші у аудіозаписі (наприклад, модель, яка натренована за критерієм CTC, може почути тишу на одному вхідному фреймі і видати “пустий” символ декодеру, який у свою чергу буде рахувати, скільки вже пустих символів було отримано: коли він отримає достатню кількість таких символів, то зможе видати end-of-sentence токен). Іншим підходом є використання так званої EP [30] – допоміжної частини в акустичній моделі, яка може розпізнавати паузи раніше, ніж на етапі декодування сигналу. Іншими словами, така конструкція має певну пам’ять і може визначати імовірнісний розподіл зважаючи на попередньо розпізнані символи. Така конструкція називається RNN-T у роботі [31]. Проблема такої конструкції в тому, що її важно натренувати [32]. Саме тому у цій роботі використаний простіший підхід, а саме такий, що автор не вимагає розпізнавання в режимі онлайн, навпаки, він дозволяє користувачу вимовити команду і зупинится, оскільки він має фіксований час на одну команду.

3.7 Сучасні інструменти для вирішення задачі ASR. End-to-end моделі.

Безпосереднім (end-to-end) розпізнаванням мови називають підхід, який замінив ті підходи, що були запропоновані раніше: використовувати окремо акустичну модель, окремо модель фонеми, окремо модель натуральної мови. Деякі сучасні інструменти для вирішення задачі прибирають проміжні ланки з процесу навчання і тренують моделі, використовуючи напряму акустичну і мовну моделі. Такі інструменти працюють із аудіо хвилею напряму.

3. 4.1 Facebook Flashlight

Flashlight [34], платформа для машинного навчання, яка належить Facebook, працює за допомогою платформи для автоматичного диференціювання Arrayfire, надає готові рішення розпізнавання мови. Методи, які використовують Flashlight, включають у себе згорткові мережі, розроблений ними критерій ASG [36], Seq2Seq [28] із використанням механізму уваги [37] і Transformer [18].

Flashlight також надає можливість декодувати вивід акустичної моделі не за допомогою променевого пошуку, а жадібно (алгоритм Вітебрі). Це може бути корисно для швидкого тестування акустичної модели, коли не так потрібно опрацьовувати її результати змістовно на високому рівні.

Променевий пошук – це евристичний алгоритм пошуку найоптимальнішого шляху серед певного постійно підтримуваного набору варіантів, який обмежений величиною, що називається розміром променю.

Користувачу Flashlight надає можливість вказувати архітектуру для акустичної модели, також він надає існуючі моделі. Для того, щоб отримати повноцінну готову для розміщення модель розпізнавання мови, потрібно спочатку запустити процес тренування акустичної моделі. Після цього можна підключити сторонню мовну модель, яка буде містити інформацію про найбільш ймовірні комбінації слів у мові, що розпізнається [36].

Arrayfire, система з відкритим вихідним кодом, яку використовує Flashlight

для автоматичного підрахунку похідних під час тренування за допомогою методу градієнтного спуску, працює наступним чином: основним об’єктом у ній є вектор, який містить також посилання на змінні, від який залежить його значення [38]. Arrayfire надає можливості ефективної низькорівневої роботи із CPU, GPU, лінивого обчислення та JIT-компіляції. Основний об’єкт Flashlight є обгорткою масиву ArrayFire. Система Arrayfire за призначенням є аналогом PyTorch та Tensorflow, але виділяється від попередніх високою швидкістю роботи, оскільки є більш новим проектом та написана на низькорівневому C++.

Для представлення лексичної структури мови Flashlight використовує бор, звичну структуру даних для роботи зі строками. Завдяки цій структурі даних декодер може визначати, які слова можуть містити продовження і динамічно вирішувати, чи потрібно йому більше інформації, щоб розпізнати слово.

3.4.2 Mozilla Deep Speech

Проект Mozilla Deep Speech імплементував систему, про яку йдеться у публікації від Baidu [40], яка стала першим відомим і вдалим застосуванням CTC.

3.4.3 LibreASR

LibreASR був започаткований не так давно. Він, як і Deep Speech, працює на прикінцевому пристрої, тобто не потребує підключення до хмарного середовища. Тренувати моделі можна також на іншому сервісі, який надає платні можливості для тренування моделей. Це швидше, ніж тренувати на власному GPU.

РОЗДІЛ 4 АНАЛІЗ ІСНУЮЧИХ ГОЛОСОВИХ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ ДЛЯ ОФІСНОЇ РОБОТИ ТА ПРОГРАМУВАННЯ

Для розуміння можливого рівня розвитку технологій надання голосових користувацьких інтерфейсів було проведено аналіз існуючих систем, які вирішують цю задачу. Вони включають у себе програмне забезпечення, здатне сприймати звук, незалежно від того, чи призначене воно для програмування (наприклад, Dragon Naturally Speaking, програмне рішення для ОС Windows, призначене для роботи із текстовими процесорами), а також спеціалізовані системи для програмування.

VUI (voice user interface) називають такі інтерфейси, які використовують голос користувача для інтерактивної взаємодії.

Частою причиною цікавості до програмування за допомогою голосу є травма сухожиль, яка призводить до незручності користування клавіатурою [42].

4.1 Dragon Naturally Speaking

Одним можливим застосуванням системи Dragon (розширена назва Dragon Naturally Speaking) може бути робота із документами в текстовому процесорі (Word, Google Docs). Застосування цієї системи можуть бути, наприклад, медичні записи або ведення документів, але для програмування така система не придатна.

Dragon NaturallySpeaking – це програма для Windows, яка розпізнає мову, надиктовану користувачем. Створена була компанією Nuance. Перші моделі були побудовані на HMM/GMM. Потім стали працювати (2016) із глибоким навчанням. Dragon не призначений для розпізнавання коду. Наприклад, для того щоб надиктувати програмний код, зображений нижче, потрібно було вимовити “spell F O R space open paren I N T space I equals zero semicolon” [42]:

```
for (int i=0; i<5; i++) {
    console.log(i);
}
```

Рисунок 3: невеликий фрагмент програмного коду, який незручно диктувати, якщо використовувати Dragon.

Багато з перелічених далі проектів знаходяться у відкритому доступі.

4.2 Програми з відкритим вихідним кодом

Natlink з'явився у 1999-му році як розширення для Dragon, що дозволяє користувачеві створювати користувачу власні команди використовуючи мову програмування Python. Той самий фрагмент, що зображений на рисунку 3, можна було б надиктувати наступним чином: “for loop five, log 1”.

Тревіс Руд презентував у 2013-му році на конференції PyCon систему, яка працювала із редактором Emacs та реагувала на голосове введення користувача. Незважаючи на те, що вона потребує вимови слів, які незвичні для людської мови, у презентації було наведено мотив такий самий, як у розробників AT&T, що раніше за нього створили систему ShortTalk/EmacsListen.

Aenea використовує віртуалізацію для роботи із Dragonfly (система, яка написана мовою Python і працює із ОС Windows) для отримання подій у ОС хоста (Linux, MacOS).

Програмне середовище Voice Code використовує Dragon Naturally Speaking.

Системи, побудовані на Kaldi, включають в себе додаток Osprey, який має більш широке використання, ніж суто програмування: він інтегрується з операційною системою (у тому числі за допмогою systemd, дозволяючи користувачу користуватися браузером, набирати текст програмного коду, виконувати дії, для яких традиційно потрібна клавіатура.

4.3 Talon Voice

Talon Voice може працювати не використовуючи Dragon Naturally Speaking. Замість нього користувач може використовувати програмне забезпечення з

відкритим вихідним кодом, опублікованим Facebook AI Research, Flashlight. Підтримує платформи Windows, Linux, macOS High Sierra.

4.4 Serenade

Serenade представляє готове рішення для розробників, це програма із закритим вихідним кодом. Програмне забезпечення інтегроване з платформами для розробки програмного коду, такими як Microsoft Visual Studio Code, Intellij IDEA, також із браузером Chrome.

Serenade використовує також і методи обробки натуральної мови (NLP), окрім ASR. Порівняно із можливостями програмних продуктів, таких як Serenade, розроблений у цій роботі програмний продукт не може сприймати натурально мову користувача і робити змістовні дії виходячи із семантики команди. У даній системі команди мають чітку структуру і більше нагадують мови програмування. Можна провести паралель із областью розробки компіляторів: інструменти такі як Serenade включають в себе не тільки лексинг (тобто отримання токенів, якими у випадку розпізнавання мови є слова або літери, в залежності від уточнення задачі), а ще і аналіз лексем на більш високому рівні (тобто парсинг у вираз і визначення змісту комбінації токенів). Такий підхід дозволив Serenade створити інтерфейс високого рівня абстракції, який слабко прив’язаний до конкретної мови програмування, тому що представляє у собі концепції, характерні для більшості мов програмування, основними такими поняттями є “функція”, “клас” тощо.

Невід’ємною складовою роботи програміста є комунікація із колегами через сервіси, які часто мають веб-інтерфейс, такими як Jira, GitHub). Serenade на момент написання роботи інтегрували своє програмне забезпечення у браузер Chrome, крім цього, в інтегроване середовище для розробки програмного коду від Intellij, також у популярні редактори програмного коду (Microsoft Visual Studio Code та ін.).

Незважаючи на це, у Serenade є окремий недолік порявняно із системою Talon Voice: перша не надає користувачу можливість отримати інтеграцію із

операційною системою та графічною оболонкою системи повноцінно, тому користуватися довільними додатками користувачі Serenade не можуть.

Потрібно зауважити, що розпізнавання натуральної мови можливе тільки після того, як мова сприйнята машиною. Людина може набрати текст за допомогою клавіатури або миші (soft keyboard), але також вона може його надиктувати. Через велику кількість акцентів, варіації голосів, через незрозумілість пауз між словами задача автоматичного розпізнавання голосу стає важкою для вирішення. Але це все ж можливо.

РОЗДІЛ 5 РОЗРОБЛЕНА СИСТЕМА

5.1 Опис функцій

Розроблений додаток містить функціонал, який дозволяє користувачу виконати розумний пошук за файлом. За роботи із відкритим файлом, який представляє із себе програмний код, який відповідає стандарту мови C++ [Посилання на стандарт], можна отримати змістовні результати пошуку мовних конструкцій за файлом. Додаток працює у текстовому редакторі Emacs.

5.2 Концепція стислої команди

Найбільш концептуально просте рішення задачі кодування інформації за допомогою мовлення було б визначати наявність чи відсутність сигналу у аудіо файлі. Наприклад, можна було в визначити тишу або нетишу. Деякі системи (зокрема Talon Voice) використовують шуми як спосіб взаємодіяти із пристроєм. Це простий спосіб, позбавлений такої семантики, як натуральна мова, але в деяких простих задачах (наприклад, симулювати натиснення лівою чи правої кнопки миші) це працює.

Таким чином, можна було б бінарно кодувати повідомлення і передавати його іншим програмам для більш високорівневого аналізу. Проте, такий підхід представляє із себе крайній випадок нестислості повідомлення. Людська мова може передавати порівнято велике когнітивне навантаження, чим більше ми аналізуємо мову: тим стисліше ми можемо собі дозволити передавати повідомлення за допомогою неї. У подальшому логічним продовженням виконаної роботи є використання методів роботи із натуральними мовами для розроблення складної граматики мови команд користувача.

5.3 Використані технології

Для досягнення поставленої в роботі задачі було використано наступні технології:

– текстовий редактор Emacs та мову програмування Emacs Lisp, яка дозволяє програмувати текстовий редактор;

– інструменти для компіляції мови C++ – GCC, Clang;

– середовище Linux із налаштованим доступом до аудіокарти за допомогою системи ALSA, яка інтегроване в використовуваній версії Linux;

– інструменти для програмування мовою Python – інтерпретатор мови CPython;

– Bash для написання скриптів для роботи на ОС Linux.

Нижче описані детально, як вибрані технології були використані.

5.4 Дизайн програми

Система була спроектована таким чином, щоб кожен компонент вирішував поставлену перед ним задачу. Виділено декілька компонентів програмної системи, які описані нижче.

Програма розроблена у форматі програмного коду, призначеного для виконання у середовищі текстового редактору Emacs та набору бінарних компонентів, які використовують файлову систему для міжпроцесорної взаємодії. Бінарні компоненти включають в себе:

– модуль розпізнавання мовлення, який використовує інструмент Facebook Flashlight;

– модуль коригування помилки в розпізнавання;

– модуль роботи з програмним кодом, який використовує бібліотеки Clang.

Для запису звуку використовується ALSA.

5.5 Задача розпізнавання тексту в аудіо файлі

Проектуючи рішення для розпізнавання тексту в аудіо файли, було вирішено не використовувати методи для розпізнавання в режимі реального часу (див. розділ 3.6). Це зумовлено тим, що команда опрацьовується одразу після того, як поступає до системи. В майбутньому за необхідності із системою Facebook Flashlight можливо налаштувати модель для розпізнавання в режимі реального часу. Такий підхід можна застосувати, якщо записувати звук користувача в окремому потоку від того, який займається розпізнаванням мовлення і записом розпізнаного фрагменту. Це можна реалізувати, наприклад, за допомогою бібліотеки для роботи із звуком sounddevice.

Для розпізнавання мовлення користувача було використано інструмент Facebook Flashlight. Оскільки він потребує акустичну і мовну моделі, було завантажено відповідні файли, які постачаються самим проектом. Після завантаження мовної та параметрів для акустичної моделі можна розпочати процес розпізнавання.

У цьому проекті використана 2-грамна статистична мовна модель.

Flashlight підтримує репозиторій із претренованими моделями, які показати свою успішність на відомих наборах даних, таких як LibriSpeech. Звідси було взято існуючу претреновану акустичну модель, яка називається Wav2Letter@anywhere.

Вдалий застосунок має бути незалежним від хмарної платформи, він має обходитися прикінцевим пристроєм і не покладатися на хмарні сховища та обчислення під час розпізнавання. Це пов’язано із небажаністю підтримувати мережеве з’єднання між клієнтом – робочою машиною користувача – і сервером – хмарним середовищем.

У вирішенні цієї задачі не потрібно використовувати низькорівневі засоби для роботи з даними, звуковими хвилями (ffmpeg та подібні). Потрібно використовувати надані бібліотекою Flashlight можливості для роботи зі звуковими файлами (у форматі FLAC або MP3).

Wav2letter та Wav2letter++ від Facebook -- продукт із відкритим програмним

кодом, який було обрано. Також у 2020-му році з’явився новий підхід до вирішення задачі розпізнавання людської мови, і він спирається на так звані Conformers. Проект також використовує згорткові нейронні мережі для розпізнавання, наприклад, перша версія wav2letter полягала саме на такий підхід.

Тренування власної моделі у flashlight має наступний вигляд: користувач повинен вибрати модулі, із яких складається нейронна мережа (подібно до існуючих фреймворків PyTorch, Tensorflow).

Зважаючи на те, що звуки різних мов часто нічого спільного між собою не мають, а також на різницю між вимовами дітей та дорослих, чоловіків та жінок, приходимо до висновку, що необхідно тренувати на різних наборах даних. Система буде краще натренована для певного користувача, які вона буде чути від нього мову безпосередньо. Тому для кращої роботи розпізнавання користувачу рекомендується тренувати модель власноруч на своїх даних. Для цього достатньо виконати процес тренування, описаний у документації проекту. на свої даних у відповідній формі.

5.6 Програмування текстового редактора

Використано текстовий редактор Emacs. В Emacs різниця між командою та функцією в тому, що команда позначається спеціальним символом у вихідному коді, “interactive”. Задача додатку в тому, щоб дозволити користувачу користуватися інтерактивними функціями, тобто командами. Для цього використано механізм Emacs, який дозволяє виконувати команди користувача в інтерактивному режимі. Основна задача додатку полягає в тому, щоб перетворити текст, розпізнаний модулем розпізнавання мови, у назву команди (інтерактивної функції), яка буде виконуватися.

Emacs – це редактор коду і водночас інтерпретатор мови Emacs Lisp. За допомогою цієї мови користувач може керувати роботою редактора інтерактивно з моменту початку роботи в запущеному редакторі.

За бажання користувач може налаштувати конфігурацію редактора Emacs

таким чином, що його можливості будуть такими ж, як і в середовищі інтегрованої розробки. Будь-який контроль над редактором здійснюється за допомогою виклику команд.

5.7 Інструмент роботи із програмним кодом

Для роботи із програмним кодом було використано можливості вибраних бібліотек Clang. Спосіб роботи із ними та деякі додаткові можливості описані нижче.

5.7.1 Можливості бібліотек Clang/LLVM

Clang має модульну структуру і надає можливість програмісту користуватися модульністю для побудови інструментів для роботи із програмним кодом. Модулі представляють із себе бібліотеки, які працюють на певних етапах компіляції. Наприклад, для роботи із генерацією коду LLVM IR (проміжне представлення, яке поступає на вхід до модулю генерування коду для конкретної архітектури процесора, наприклад, x86_64), можна використовувати libcodegen. Для роботи із синтаксичною структурою програмного коду такий модуль не використовують, оскільки для задач, поставлених у роботі, достатньо зупинитися на етапі побудови синтаксичного дерева. Нижче наведені компоненти, які необхідно окреслити для розуміння процесу побудови програмної частини роботи.

5.7.1.1 Отримання програмного доступу до AST вихідного коду

Для отримання програмного доступу до AST існує комплекс модулів, які працюють разом до отримання представлення одиниці компіляції у вигляді дерева, яке містить ідентифікатори із посиланнями на ділянки коду, де вони розміщені. libAST містить програмний код, який визначає AST як структуру даних. libASTMatchers надає функціонал для пошуку вузлів у синтаксичному

дереві.

Варто зауважити, що в цій роботі важливий зв’язок між вузлами AST та позиціями ідентифікаторів у вихідному коді. Існує можливість отримати інформацію про знаходження ідентифікатора із вузла синтаксичного дерева у вигляді пари цілих чисел за допомогою метода getSourceLocation.

5.7.1.2 Можливості модифікації вихідного програмного коду

Бібліотека Clang libRewrite надає клас Rewriter для роботи із модифікації вихідного коду, який представлений класами FileManager, SourceBuffer. В цій роботі не було використано цей функціонал. Модифікуючі операції над вихідним кодом (наприклад, рефакторинг) можуть потребувати використання цих класів.

5.7.1.3 Написання інструментів для роботи із одиницями трансляції

В стандарті C++ сказано про одиницю трансляції як про найменший об’єкт вихідного коду, із яким може працювати компілятор. Зазвичай це файли із розширенням “.cpp”. Бібліотека libFrontend знаходиться в дереві проекту LLVM та дозволяє користувачу будувати інструменти для роботи із кодом.

За необхідності працювати із більше ніж однією одиницею компіляції можна використовувати функціонал, який надається класом CrossTU. Ericsson Codecheker – система для статичного аналізу вихідного коду, яка використовує такий функціонал. Для роботі із багатьма одиницями компіляції необхідно підготувати базу даних компіляції у форматі JSON.

5.7.1.4 Пошук вузлів в AST

Для пошуку вузлів в синтаксичному дереві використана бібліотека LibASTMatchers, що надає можливість шукати вирази у вихідному коді за допомогою спеціально розробленої DSL. Існує декілька видів виразів, які можна

комбінувати. В цій роботі використано клас виразів, який вибирає вузли із дерева, виходячи з їх типу (Node Type matchers).

РОЗДІЛ 6 ПОРІВНЯННЯ З ІСНУЮЧИМИ СИСТЕМАМИ

Існують різні способи роботи вирішення поставленої в цій роботі мети. В таблиці 6.1 наведено порівняння розробленої системи та двох інших – Talon Voice та Serenade. З таблиці видно, що можливі способи будувати користувацький інтерфейс варіюються в залежності від міри інтеграції з операційною системою, яка обслуговує користувача. Іншим параметром, за яким виконано порівняння, є можливість конфігурувати і налаштовувати систему таким чином, щоб кінцевий користувач міг, наприклад, додавати свої команди в систему (це можливо із Talon Voice, але неможливо із Serenade).

Таблиця 6.1 – Порівняння функціональних характеристик роботи із системою Serenade та Talon Voice

	Talon Voice	Serenade	Розроблена система
Інтеграція із операційною системою	Linux, macOS, Windows	Linux, macOS, Windows	Linux
Ліцензія	Закритий вихідний код; безкоштовне користування	Закритий вихідний код; комерційний продукт	Відкритий вихідний код; безкоштовне користування
Конфігурованість	Можна конфігурувати мовою Python	Можна додавати плагіни для роботи із конкретними програмами	Можна конфігурувати мовою Emacs Lisp

РОЗДІЛ 7 НАСТУПНІ КРОКИ У РОЗВИТКУ ГОЛОСОВИХ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ

7.1 Спеціальні можливості для мови програмування C++

Робота із заданою мовою програмування може потребувати характерної саме для заданої мови підтримки у середовищі програмування. Нижче наведено перелік можливих розширень для роботи із C++:

– може бути корисно візуалізувати роботу компілятора зі спеціалізацією та інстанціюванням шаблонів, як це робить metashell;

– дослідження композиції між класами, враховуючи типи змінних класу;

– підтримка пошуку за визначенням оператора.

7.2 Інтеграція із операційною системою

У повсякденній роботі розробника виникає необхідність працювати з терміналом, браузером та інтерактивним середовищем розробки.

В цій роботі було вирішено не виконувати інтеграцію із операційною системою, оскільки це потребує більших знань та додаткового часу. Далі розглянуто можливості використання інструментів для досягнення більшого рівня інтеграції із операційною системою.

7.2.1 Linux

Можна навести різні підходи для інтеграції програмної системи з операційною системою Linux. Нижче наведені можливі напрямки для роботи:

– демонізація процесів для запуску їх при старті системи за допомогою сервісів system. Це може бути корисно для побудови інтерфейсу для диктування, який сприймає звук постійно протягом робочої сесії користувача;

– використання розширеннями, які надає X Server. Наприклад розширити

можливості контролю віконного середовища дозволом до надсилання команд за допомогою мережевого з’єднання до віддаленого X клієнту, наприклад за допомогою протоколу SSH. Передача аудіо хвилі мережею спричинить велике навантаження на мережу, що не потрібно, зважаючи на те, що потрібно контролювати буфер;

– інтеграція на рівні DRM (Direct Rendering Infrastructure) для побудови влаштованої в систему системи користувацької графіки;

– використання композиторів віконного середовища. Існує два підходи до композиції в такій задачі – Wayland та X Window Server. Композитори дозволяють створити цікаві інтерфейси, оскільки передбачають програмну маніпуляцію інформацією про піксель, яка надається відеокарті для відображення, що дозволяє надати більше часу програмній системі для обчислень над вихідною сукупністю пікселів.

7.2.2 Windows

Microsoft надає можливість працювати із аудіокартою за допомогою Windows SDK, що може бути використано на практиці для портування частини роботи із записом голосового введення користувача.

ВИСНОВКИ

В рамках виконаної роботи було розроблено систему, яка дозволяє користувачу керувати текстовим редактором за допомогою голосу. Було використано поточний рівень розвитку можливостей для розпізнавання голосу. Також було виконано інтеграцію із бібліотекою Clang/LLVM для роботи із синтаксичним деревом вихідного коду мовою C++.

У подальшому хотілося б розширити систему інтеграцією із рішеннями NLP, які могли б допомогти побудувати граматику для мови, яку можна було в використовувати в інтерфейсі. Крім того, можна розширити систему шляхом додавання нових функціональних можливостей для роботи з програмним кодом; додаванням контекстного меню для збільшення рівня інтерактивності системи та іншими способами надавати спеціальні можливості для користувачів, у яких немає можливості набирати текст на клавіатурі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. T. Brown, B. Mann, N. Ryder Language Models are Few-Shot Learners / T. Brown, B. Mann, N. Ryder [et al] [Edited by: H. Larochelle and M. Ranzato and R. Hadsell and M.F. Balcan and H. Lin]. – 2020. – Advances in Neural Information Processing Systems 33. – V. 33. – ISBN: 9781713829546
2. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. – BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. – 2019. – V. 1. – P. 4171-4186.
3. E. Musk An integrated brain-machine interface platform with thousands of channels / E. Musk. – 2019. – (Препринт bioRxiv)
4. Mozilla Common Voice : проект, що займається збиранням голосових даних на різних мовах для формування вибірки для тренування інструментів машинного навчання [Електронний ресурс]. – Режим доступу до ресурсу: https://commonvoice.mozilla.org/
5. Web Speech API : робота над специфікацією голосового інтерфейсу для браузерів [Електронний ресурс]. – Режим доступу до ресурсу: https://wicg.github.io/speech-api/
6. L. R. Rabiner, B. H. Juang An Introduction to Hidden Markov Models / L. R. Rabiner, B. H. Juang. – 1986. – January 1986. – P. 4-16. – ISSN 1558-1284
7. S. Goldwater, D. Jurafsky, C. D. Manning Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates / S. Goldwarer, D. Jurafsky, C. D. Manning. – 2009. – (Препринт / Elsevier Speech Communication ISSN: 0167-6393)
8. Блог команії Yandex [Електронний ресурс]. – Режим доступу до ресурсу. – https://habr.com/en/company/yandex/blog/198556/
9. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups / G. Hinton, L. Deng, D. Yu [et al]. – 2012. – V. 29, I. 6. – IEEE Signal Processing Magazine, ISSN 1558-0792

10. Speech recognition with deep recurrent neural networks / A. Graves, A. Mohamed, G. Hinton – 2013. – International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2379-190X
11. Connectionist temporal classification: Labeling unsegmented sequence data with recurrent neural networks / A. Graves [et. al]. – 2006. – Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006
12. Speech recognition with weighted finite-state transducers / M. Mohri, F. Pereira, M. Riley. – 2008. – Springer Handbook of Speech Processing
13. librosa : програмний код для роботи із аудіо файлами для перетворення їх у змістовне для подальшого опрацювання представлення [Електронний ресурс]. – Режим доступу до ресурсу: https://librosa.org/
14. ALSA : програмне забезпечення для роботи зі звуковою картою на пристроях під ОС Linux [Електронний ресурс]. – https://github.com/alsa-project
15. A. Graves Sequence Transduction with Recurrent Neural Networks / A. Graves. – 2012. – (Препринт arXiv)
16. K. Jing, J. Xu A Survey on Neural Network Language Models / K. Jing, J. Xu (Препринт arXiv)
17. KenLM : програмний код статистичної мовної моделі [Електронний ресурс]. – Режим доступу до ресурсу: https://kheafield.com/code/kenlm/
18. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin Attention is All you Need. / A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. // Advances in Neural Information Processing Systems. – 2017. – V. 30. – ISBN 9781510860964
19. A. Graves Generating Sequences With Recurrent Neural Networks / A. Graves. – 2014. – (Препринт arXiv)
20. A comparison of Sequence-to-Sequence Models for Speech Recognition / R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, N. Jaitly. // Interspeech

2017. – Stockholm, Sweden. – ISSN 1990-9772

21. H. T. Siegelmann Computation Beyond the Turing Limit / H. T. Siegelmann [Department of Information Systems Engineering, Faculty of Industrial Engineering, Technion, Haifa 32000, Israel]. – Science. – 1995. – V. 268, I. 5210. – P. 545-548. – ISSN 1095-9203.
22. S. Hochreiter, J. Schmidhuber Long Short-Term Memory / S. Hochreiter, J. Schmidhuber. // Neural Computation. – 1997. – V. 9, I. 8. – P. 1735-1780. – ISSN 1735–1780.
23. K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation / K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2014. – P. 1724-1734
24. A. Graves Supervised Sequence Labelling with Recurrent Neural Networks / A. Graves. – 2012. – ISBN 978-3-642-24797-2
25. D. Bahdanau, K. Cho, Y. Bengio Neural Machine Translation by Jointly Learning to Align and Translate / D. Bahdanau, K. Cho, Y. Bengio. – 2014. – (Препринт arXiv)
26. C.-C. Chiu, C. Raffel Monotonic Chunkwise Attention / C.-C. Chiu, C. Raffel. – 2017. – (Препринт arXiv)
27. A. Graves, N. Jailty Towards End-to-End Speech Recognition with Recurrent Neural Networks / A. Graves, N. Jailty. // Proceedings of the 31st International Conference on International Conference on Machine Learning. – 2014. – V. 34.
28. I. Sutskever, O. Vinyals, Q. V. Le Sequence to Sequence Learning with Neural Networks, I. Sutskever, O. Vinyals, Q. V. Le Sequence to Sequence Learning with Neural Networks // Proceedings of the 27th International Conference on Neural Information Processing Systems. – 2014. – V. 2.
29. An All-Neural On-Device Speech Recognizer [Електронний ресурс] : стаття у блозі Google про досягнення у розпізнаванні голосу на мобільних пристроях

без використання хмарного середовища за допомогою моделі RNN-T / Schalkwyk. – 2019. – Режим доступу до ресурсу: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

30. B. Li, Shuo-yiin Chang, T. N. Sainath Towards Fast and Accurate Streaming End-To-End ASR / B. Li, Shuo-yiin Chang, T. N. Sainath [et al] // International Conference on Acoustics, Speech, and Signal Processing (ICASSP). – 2020. – ISSN 2379-190X
31. J. Li, R. Zhao, Z. Meng Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability / J. Li, R. Zhao, Z. Meng [et al] // Interspeech 2020. – Shanghai, Chine. – ISSN 1990-9772
32. S. Wang, P. Zhou, W. Chen Exploring RNN-Transducer for Chinese speech recognition / S. Wang, P. Zhou, W. Chen // Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). – 2019. – ISSN 2640-0103
33. A. Hannun, A. Lee, Q. Xu, R. Collobert Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolution / A. Hannun, A. Lee, Q. Xu, R. Collobert // Interspeech 2019. – Graz, Austria. – ISSN 1990-9772
34. Facebook Flashlight [Електронний ресурс] : бібліотека для машинного навчання за допомогою мови C++. – Режим доступу до ресурсу: https://github.com/flashlight/flashlight
35. Пилипенко В. О., Слюсарь І. І., Слюсар В. І. Варіант використання нейронної мережі в системі «Smart Home». // IV Міжнародна науково-практична конференція «Інтеграція інформаційних систем і інтелектуальних технологій в умовах трансформації інформаційного суспільства», що присвячена 50-ій річниці кафедри інформаційних систем та технологій, 21-22 жовтня 2021 р., Полтава: Полтавський державний аграрний університет. — С. 93 — 96. DOI: 10.32782/978-966-289-562-9.
36. R. Collobert, C. Puhrsch, G. Synnaeve Wav2Letter: an End-to-End ConvNet-based Speech Recognition System / R. Collobert, C. Puhrsch, G. Synnaeve. – 2016. – (Препринт arXiv)

37. J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio Attention-Based Models for Speech Recognition / J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio // Proceedings of the 28th International Conference on Neural Information Processing Systems. – 2015. – V. 1 – P. 577-585
38. J. G. Malcolm ArrayFire: a GPU acceleration platform / J. G. Malcolm // Proceedings of SPIE - The International Society for Optical Engineering. – 2012. – V. 8403.
39. Mozilla DeepSpeech [Електронний ресурс] : програмний продукт, який використовує підхід CTC для вирішення задачі розпізнавання мовлення. – Режим доступу до ресурсу: https://github.com/mozilla/DeepSpeech
40. A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng Deep Speech: Scaling up end-to-end speech recognition / A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng. – 2014. – (Препринт arXiv)
41. B. Stroustrup The C++ Programming Language / B. Stroustrup. – [4-е видання]. – [Б. м.] : Addison-Wesley Professional, 2013. – ISBN: 9780133522884
42. L. Rosenblatt, P. Carrington, K. Hara, J. P. Bigham Vocal Programming for People with Upper-Body Motor Impairments / L. Rosenblatt, P. Carrington, K. Hara, J. P. Bigham // Proceedings of the 15th International Web for All Conference. – 2018. – A. No. 30. – Pages 1-10. – ISBN 978-1-4503-5651-0