КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота

за спеціальністю122 Комп'ютерні науки
на тему:

ІНДЕКСАЦІЯ ТА ПОШУК АУДІО-ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ТЕКСТОВИХ ЗАПИТІВ

Виконала студентка 4 курсу
Зайцева Єлизавета Едуардівна

(підпис)

Науковий керівник:
Асистент кафедри математичної інформатики
Бобиль Богдан Володимирович

Консультант:
доцент кафедри математичної інформатики, кандидат фіз.-мат. наук
Деревянченко Олександр Валерійович

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

Студент _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 41 сторінка, 17 ілюстрацій, 19 джерел посилань.

ВЕКТОРНЕ ПРЕДСТАВЛЕННЯ СЛІВ, ДАТАСЕТ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, МУЛЬТИКЛАСОВА КЛАСИФІКАЦІЯ , РЕКУРЕНТНА НЕЙРОННА МЕРЕЖА, СЕМАНТИЧНИЙ ПОШУК, СПЕКТРОГРАМА, ТРЕНОВАНА МОДЕЛЬ, ТОКЕНІЗАЦІЯ.

Об’єктом дослідження є створення системи, яка буде індексувати аудіо-інформацію у текстовий формат та робити пошук необхідних аудіо-файлів за текстовими запитами від користувачів.

Предметом роботи є проектування програмного засобу для пошуку аудіо-файлів через текстові запити.

Метою роботи є проектування програмного засобу для пошуку аудіо-файлів.

Методи розроблення: навчання нейронної мережі для перетворення аудіо-файлів у текстовий формат, використання нейронної мережі для конвертації тексту в ембедінги, проектування семантично-пошукової системи.

Інструменти розроблення: хмарне середовище розробки Google Colaboratory, мова програмування Python 3.8.0.,

Результати роботи: проаналізовано існуючі методи конвертування аудіо-інформації в текстовий формат та на їх основі створено нейронну мережу, спроектовано програмний продукт для пошуку аудіо-файлів за допомогою текстових запитів.

Програмний продукт для пошуку аудіо-файлів за допомогою текстових запитів може застосовуватися, у таких комерційних проектах як, музичні стрімінгові сервіси, додатки з аудіо-книгами або подкастами. Окрім цього

можливе використання для полегшення аналізу зворотнього зв’язку клієнтів/користувачів через колл-центри.

ЗМІСТ

	С.
ВСТУП	4
РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ	7
1.1 Огляд основних підходів до опрацювання аудіо-інформації	7
1.2 Огляд основних технік семантичного пошуку	14
РОЗДІЛ 2 АНАЛІЗ ТА ПОПЕРЕДНЯ ОБРОБКА ДАТАСЕТА	20
2.1 Аналіз датасета	20
2.2 Попередня обробка датасета	24
РОЗДІЛ 3 ТРЕНУВАННЯ НЕЙРОННОЇ МЕРЕЖІ	28
3.1 Вибір архітектур нейронної мережі для поставленої задачі	28
3.2 Результати використання архітектур та вибір найоптимальнішої з них	33
РОЗДІЛ 4 ПРОЕКТУВАННЯ ПРОЦЕДУРИ ПОШУКУ ЗА ТЕКСТОВИМ ЗАПИТОМ	37
ВИСНОВКИ	39
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	40

ВСТУП

Оцінка сучасного стану об’єкта розробки. Станом на сьогодні лідери ринку у сфері технологій розробили досить багато програмних рішень для перекладу аудіо-записів у текстову форму. Більшість з цих рішень розроблена для англійської мови, проте найбільш популярні мають також опції для багатьох інших мов.

Одним з найвідоміших засобів є “Speech-to-Text” від компанії Гугл. Цей продукт дає змогу користувачам застосовувати попередньо треновані нейронні мережі для розпізнавання тексту на декількох мовах у найбільш популярних форматах аудіо-файлів. Окрім цього, велику популярність мають сервіси для запису під диктовку, які використовують дані, отримані з мікрофона пристрою та транскрибують їх у текстовий документ. На даний момент, в результаті дослідження існуючих рішень, не було знайдено моделей, які б розпізнавали українську мову.

Стосовно розробок у сфері пошуку аудіо записів за текстовими запитами, таких засобів дуже мало. Найбільш простим способом залишається використання пошукових систем, зокрема, такий спосіб є найбільш ефективним для пошуку пісень. Іншою опцією є додаток та сайт Genius, що представляє собою велику базу текстів пісень. Додаток дозволяє знаходити тексти в яких буде присутнім текстовий запит користувача. Варто відмітити, що в результати пошуку також включено тексти, у яких запит присутній у якості підстроки, що, у більшості випадків сповільнює процес пошуку та дає зайві результати. Досить потужним засобом є пошук по аудіо від Deepgram. В його основу закладена індексація аудіо файлу, та подальший пошук в отриманому тексті. В процесі дослідження, не було знайдено пошукових додатків, або існуючих баз з аудіо, які б дозволяли

проводити не просто пошук конкретних слів або фраз у тексті, а знаходити семантично близькі до пошукового запиту тексти.

Актуальність роботи та підстави для її виконання. Після аналізу та дослідження існуючих розробок перекладу аудіо-формату в текстовий та наявності баз або додатків для здійснення семантичного текстового пошуку аудіо файлів, було зроблено висновки, що створення подібної системи, яка буде сприймати українську мову є досить актуальною задачею.

Враховуючі той факт, що зараз можна спостерігати стрімке зростання україномовного контенту в багатьох напрямах, таких як музика, кінематограф, подкасти, література та аудіо книги. Цілком очевидною є потреба у системі, яка б давала змогу проводити швидкий та гнучкий пошук контенту на різних платформах.

Така система у рази пришвидшує пошук необхідної інформації, адже дозволяє економити час, прибираючи необхідність прослуховувати частини контенту, щоб знайти необхідні файли.

Мета й завдання роботи. Метою дипломної роботи є створення системи для індексування україномовних аудіо файлів у текстовий формат та здійснення семантичного пошуку серед аудіо файлів за текстовими запитами. Для досягнення цієї мети поставлено такі завдання.

1) Дослідити існуючі підходи для переведення мовлення в текст.
2) Визначити найбільш релевантний підхід, враховуючі специфіку поставленої задачі.
3) Створити датасет та провести необхідний препроцесинг для тренування нейронної мережі.
4) Підібрати архітектуру нейронної мережі та провести її навчання на зібраному датасеті.

5) Розробити процедури, для семантичного пошуку аудіо файлів.

Об'єкт, методи й засоби розроблення. Об’єктом розроблення системи є індексація україномовних аудіо файлів та семантичний пошук серед цих файлів за текстовими запитами.

Розробці програмного засобу передувало створення набору навчальних даних, та вибір моделей нейронних мереж, які найкраще підходять для розв’язання задач даного типу.

Під час розробки програмного продукту використана еволюційна модель, заснована на таких принципах. Розробляється початкова версія продукту, у нашому випадку нейронна мережа, яка потім оцінюється за допомогою тестових даних, після чого, у разі недостатньої степені точності, модель корегується та знову перевіряється її якість.

В якості інструменту створення програмного засобу було обрано Jupyter Notebook 6.3.0 - який є безкоштовним, вільно поширюваним, з відкритим вихідним кодом. У якості мови програмування обрано Python 3.8.0.

Python надає широкий набір бібліотек машинного та глибокого навчання, а також набір бібліотек для маніпуляції різноманітними даними та їх візуалізації.

Можливі сфери застосування. Дана система може застосовуватися, як додаткова функція у стрімінгових сервісах з піснями, подкастами, аудіо-книгами.

Окрім цього, можливою сферою застосування є автоматизація та спрощення обробки зворотнього зв’язку від користувачів/клієнтів.

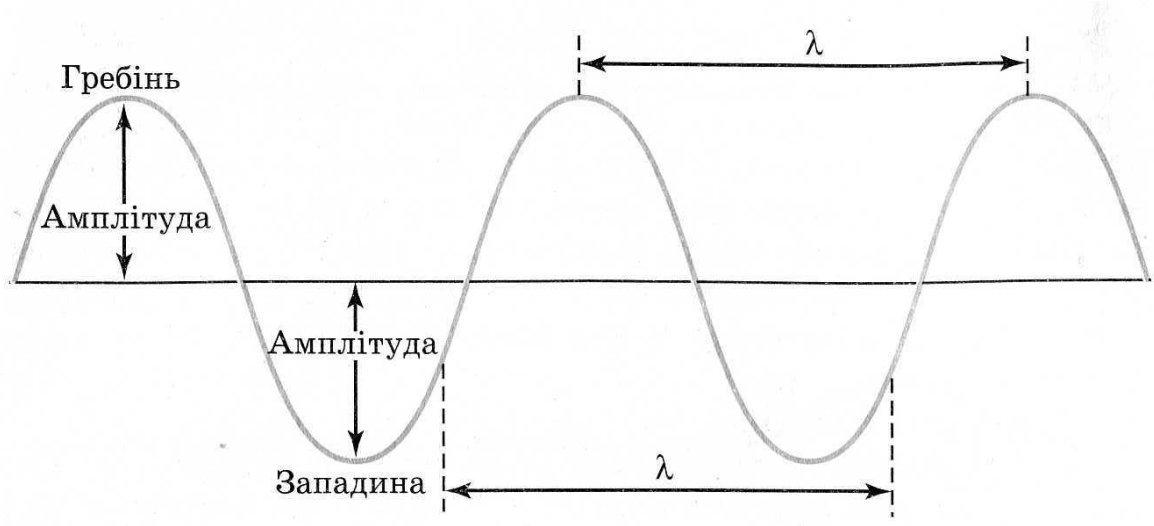
РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Огляд основних підходів до опрацювання аудіо-інформації

Щоб працювати з аудіо-інформацією необхідно розуміти, що вона собою представляє. У якому вигляді аудіо опрацьовується електронними пристроями, якими параметрами може бути описана та у яких форматах може зберігатися

Основними параметрами звукових хвиль є:

- Амплітуда - максимальне відхилення молекул зі стану спокою
- Довжина – відстань між двома найближчими точками в просторі, в яких коливання відбуваються в однаковій фазі
- Частота – кількість коливань в секунду



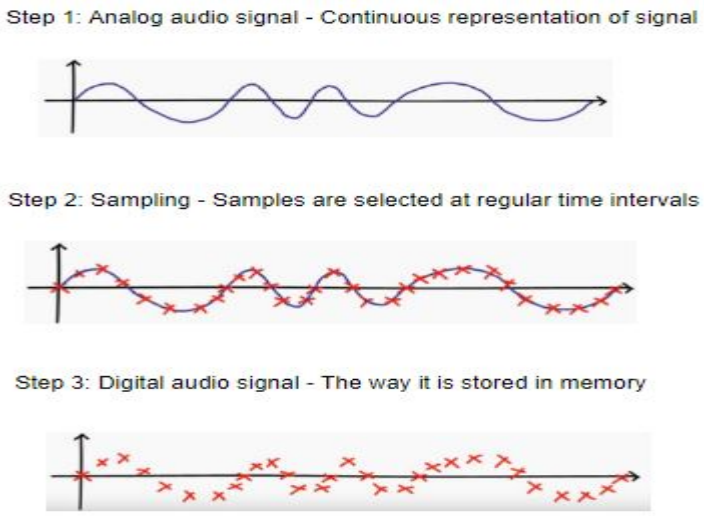
Малюнок 1

Існують два типи сигналів - аналоговий і цифровий. Цифровий тип є репрезентацією сигналів у дискретній формі, тобто на будь-якому проміжку сигналу буде існувати скінчена кількість точок.

Аналоговий же сигнал це безперервна хвиля, на проміжку якої існує нескінчена кількість точок.

Природньо звукові хвилі існують у вигляді аналогових сигналів, проте для роботи зі звуком треба використовувати цифрову форму. Найпростішим

способом для конвертування безперервної хвилі в дискретну є вибір деякої кількості точок за певний проміжок часу



Малюнок 2

Важливим є вибір форми, у якій буде вестись робота з аудіо-сигналом. Основними варіантами репрезентації сигналів є амплітуда-час або амплітуда-частота. Значним обмеженням першого варіанту є те, що він повністю ігнорує інформацію про частоту сигналу. Другий же варіант навпаки ігнорує послідовність сигналу у часі, що є критично важливим для задачі розпізнавання мовлення.

За допомогою швидкого перетворення Фур’є можливо трансформувати дискретні дані у окремі спектральні компоненти і таким чином отримати з представлення сигналу у форматі амплітуда-час, формат амплітуда-частота.[1] Однак, як було зазначено вище, ми не можемо нехтувати часовою компонентою, адже одним з ключових елементів є саме послідовність звуку. Певною мірою при роботі з аудіо-інформацією її варто сприймати як часовий ряд, адже ми при дослідженні орієнтуємось на аналіз змін тих чи іншим параметрів протягом часу і послідовність значень цих параметрів є невід’ємним елементом аналізу аудіо-інформації.

В такому разі доцільно використовувати віконне перетворення Фур'є[2]. Суть віконного перетворення заключається в тому, що сигнал розбивається на підінтервали та до цих підінтервалів застосовується швидке перетворення Фур'є. Математична формула віконного перетворення має такий вигляд:

$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n} = \mathrm{DTFT}_\omega(x \cdot \textsc{Shift}_{mR}(w)),$$

де:

$$\begin{aligned} x(n) &= \text{input signal at time } n \\ w(n) &= \text{length } M \text{ window function } (e.g., \text{Hamming}) \\ X_m(\omega) &= \text{DTFT of windowed data centered about time } mR \\ R &= \text{hop size, in samples, between successive DTFTs.} \end{aligned}$$

При виконанні умови $\boxed{\sum_{m=-\infty}^{\infty} w(n-mR) = 1,\ \forall n \in \mathbf{Z}}$ $(w \in \textsc{Cola}(R))$

Отримуємо наступну рівність:

$$\begin{aligned} \sum_{m=-\infty}^{\infty} X_m(\omega) &\triangleq \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \underbrace{\sum_{m=-\infty}^{\infty} w(n-mR)}_{1 \text{ if } w\in\textsc{Cola}(R)} \\ &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\ &\triangleq \mathrm{DTFT}_\omega(x) = X(\omega). \end{aligned}$$

[3]

На практиці віконна трансформація рахується як послідовність дискретних перетворень Фур'є, де вікно ковзає вперед по інтервалам часу. Реалізація обчислення з теоретичної формули виглядає наступним чином:

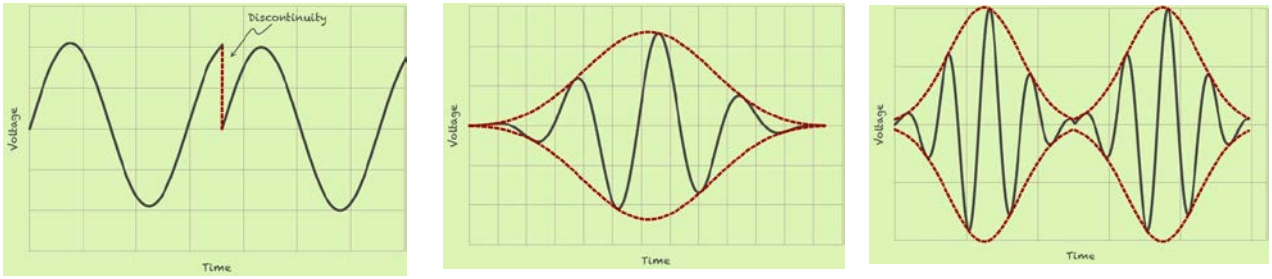
$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x(n+mR)w(n)e^{-j\omega(n+mR)}$$

$$= e^{-j\omega mR} \sum_{n=-\infty}^{\infty} x(n+mR)w(n)e^{-j\omega n}$$

$$= e^{-j\omega mR}\mathrm{DTFT}_{\omega}(\textsc{Shift}_{-mR}(x) \cdot w).$$

В цій формі, дані, що знаходяться в околі mR переводяться у нульовий час та потім множаться на вікно w, і після цього виконується дискретне перетворення Фур'є.

При розбитті аудіо на підінтервали, кордони виділених сегментів видно як розриви, яких в реальності в сигналі немає. Задля того, щоб зменшити вплив сегментації використовуються віконні функції. За рахунок того, що віконні функції на кордоні прямують до нуля, розриви на кордонах сегментів стають непомітними, тобто віконні функції дозволяють змінювати сам сигнал, але зміни сигналу мінімально впливають на його статистику.[4]



Малюнок 3

Основними функціями вікон є:

- Вікно Кайзера – це вікно дозволяє визначити сигнали, близькі до мінімального рівню шуму, які інші вікна можуть приховати, через це вікна Кайзера часто використовуються у багатьох аналізаторах спектру за замовчуванням.
- Вікна Хеммінга та Ханна – ці вікна будуються шляхом множення прямокутного вікна на один період косинуса. Завдяки швидкому

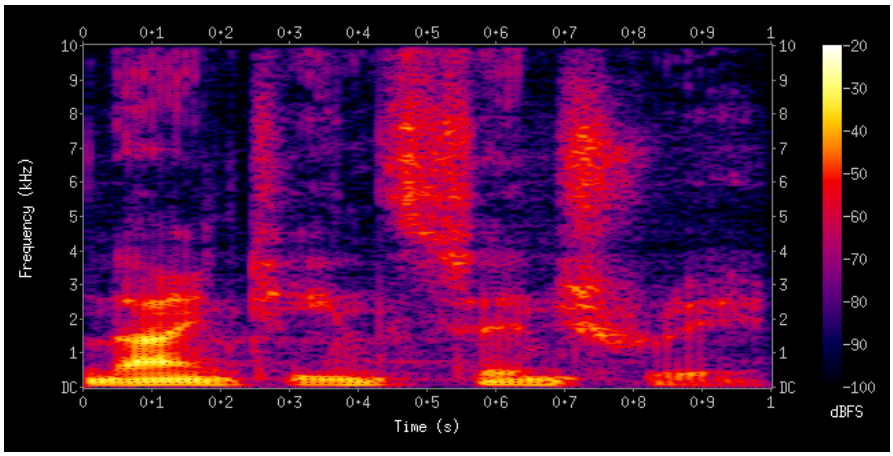
обчисленню та високій якості обробки шумів ці вікна використовуються в роботах пов’язаних з аналізом та обробкою мовлення.

- Прямокутне вікно – це фактично, відсутність вікна, яка створює враження наче форма хвилі вмикається та вимикається

Коли визначено яке саме вікно використовувати треба обрати два важливих параметри вікна: його ширину та міру перекриття з наступним кроком. Зазвичай для задач розпізнавання мовлення використовується ширина в двадцять – тридцять мілісекунд, приблизно стільки часу займає вимовлення одного звука.

Для запобігання втраті частини даних при використанні вікон додатково використовується перекриття, тобто вікно на наступному кроці буде містити якусь частину інтервалу з попереднього кроку. Загально прийнятою практикою є перекриття від двадцяти п’яти до семидесяти п’яти відсотків, для вікон сімейства Хемінга прийнято брати п’ятдесят відсотків. При виборі перекриття важливо не обрати занадто великий показник, в результаті занадто великого рівня перекриття у діаграмі віконного перетворення може виникнути картина даних, яка не буде відповідати дійсності.[5]

Для візуального відображення віконного перетворення Фур’є використовується спектрограма – візуальне відображення графіка між частотою та амплітудою, де кожна точка відображає амплітуду для конкретного моменту часу і частоти за допомогою шкали кольору.



Малюнок 4

Дослідження показали, що людина сприймає звуки не лише в лінійній шкалі. Ми краще сприймаємо відмінності у звуці на нижчих діапазонах частот. Наприклад людина легко розрізнить п’ятсот і тисячу герц, але майже не помітить різницю між десятьма тисячами та одинадцятьма тисячами герц.

У тисяча дев’ятсот тридцять сьомому році Стівенс, Фолькманн і Ньюманн Запровадили нову одиницю висоти звуку – мел. Ідея цієї одиниці полягає в тому, щоб звуки, які мають однакову відстань за шкалою Мел, сприймалися людиною як така сама однакова відстань. Тобто, щоб для людини різниця у сто герц для ста і двохсот герц сприймалась так само як різниця між тисячею та тисячею сто герц.[6]

Перехід з герців до мел може здійснюватись за декількома формулами, але їх суть однакова – логарифмування.

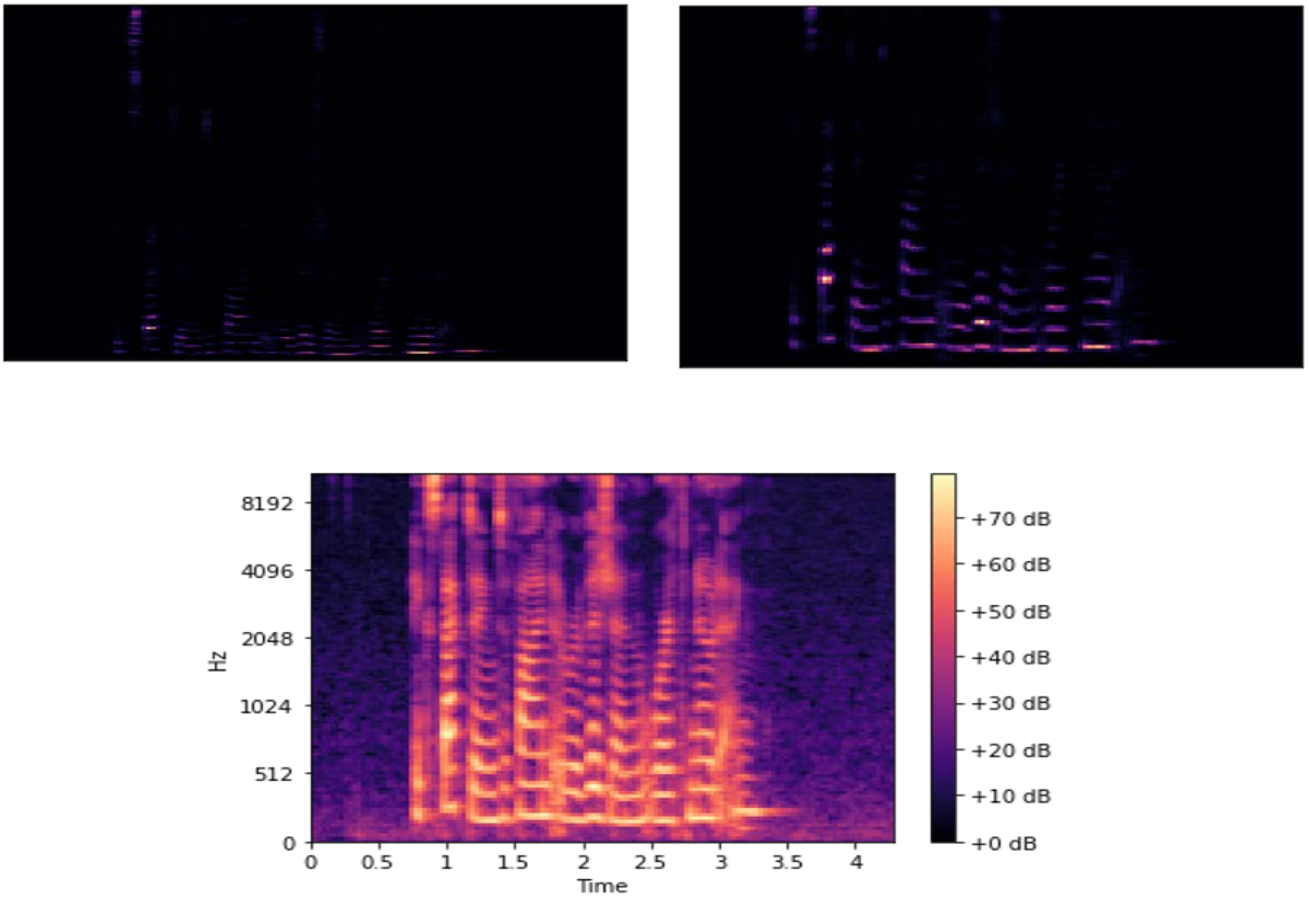
Приклади формул: $$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$

Якщо ми будемо використовувати у якості вхідних даних для моделі спектрограму у сирому вигляді – отриману після використання віконного перетворення Фур’є, то ризикуємо навчати модель на малоінформативних даних.

Вирішити цю проблему може шкала Мел, а саме провести перетворення частот з герців в мели та додатково замість амплітуди, яка використовується за замовчанням, використати шкалу децибелів, що також є логарифмічною шкалою. Десять децибелів гучніше нуля у десять разів, двадцять децибелів гучніше десяти децибелів у сто разів.

Нижче приведено приклади трьох спектрограм:

- Проста спектрограма з частотою в герцах та амплітудою
- Спектрограма з частотою в шкалі Мел та амплітудою
- Спектрограма з частотою в шкалі Мел та децибелами



Малюнок 5

З зображень добре видно, що приведення параметрів до шкали Мел дозволяє отримати більше інформативніше зображення аудіо-інформації.[7]

Мел спектрограми можуть бути використані, як вхідні дані для моделей, а можуть бути проміжним етапом у обробці даних для моделі. У другому випадку річ йде про MFCC – Mel frequency cepstral coefficient.[8] Це представлення спектру потужності звуку, заснованого на лінійному косинусному перетворенні логарифмічного спектру потужності на нелінійній шкалі частоти мел.

MFCC обчислюються шляхом застосування фільтра попереднього наголосу до аудіо сигналу, використання віконного перетворення Фур’є,

застосування фільтрів на основі шкали Мел, використання дискретного косинусного перетворення і нормалізації виходу.

Таким чином на виході отримується вектор розмірності тринадять – двадцять ознак, який можна використовувати в тренуванні моделі.

1.2 Огляд основних технік семантичного пошуку

На відміну від лексичного пошуку, де шукаються збіги конкретних слів запиту або їх комбінації, семантичний пошук дозволяє знаходити дані які б мали схоже з запитом значення.

Семантичний пошук використовує наміри, контекст і концептуальні значення запиту користувача, щоб знайти контент який би відповідав змісту запиту.

Такий спосіб відрізняється від чистого пошуку за ключовими словами. Чистий пошук за ключовими словами працює через зіставлення слів запиту зі словами в документах. І навпаки, семантичний пошук часто повертає результати, у яких немає збігів слів, але вміст все одно явно відповідає тому, що шукає користувач.

Дуже важливу роль відіграє контекст, у якому відбувається пошук. Інтелектуальна пошукова система може використовувати контекст як на особистому, так і на груповому рівні. Особистий рівень впливу на результати називається персоналізацією і використовує попередні пошуки окремого шукача та попередні взаємодії з пошуковою системою, щоб повернути вміст, який найкраще підходить для поточного запиту.

На рівні групи пошукова система може переранжувати результати, використовуючи інформацію про те, як усі шукачі взаємодіють з результатами пошуку, наприклад, які результати натискають найчастіше, або навіть сезонність того, коли певні результати популярніші за інші.

Одним з ключових моментів успішного пошуку є правильно сформований запит. Від того, у який спосіб користувач сформує запит, буде залежати чи вдасться знайти шукану інформацію. Так запит з помилками може значно знизити коло пошуку, тому першим кроком у системі має бути перевірка запиту на наявність помилок та їх автоматичне коригування або пропонування можливих варіантів корекції користувачу.

Ще одна проблема, яка виникає при пошуку запиту - це виявлення зв’язків між словами в запиті, тобто які зі слів більш імовірно використовуються як словосполучення та які зв’язки слова можуть мати одне з одним.

Можливі ситуації появи у запиті таких слів, яких не має в морфологічному словнику системи. У такому разі система повинна вивчити нове слово самостійно скануючи ряд відомих словників. Можливе інше рішення проблеми - пошук контекстів, у яких слово використовується, для визначення його семантики. Для цього зазвичай використовують метод скіп-грам. Це один із методів навчання без нагляду, який використовується для пошуку найбільш споріднених слів для певного слова. Він використовується для передбачення контекстного слова для даного цільового слова.

Унікальність тексту також важлива для пошукових систем. Під час відбору результатів пошуку алгоритм повинен враховувати оригінальність, визначення якої може зайняти багато часу та ресурсів, якщо порівнювати абсолютно всі тексти. Тому задля оптимізації цього процесу варто використовувати спеціальні методи. Наприклад, можна використовувати метод н-грам, який працює з ймовірностями і, таким чином, може не сканувати всі слова, а пропускати деякі з них з певним кроком. Інший підхід полягає в тому, щоб надати більшу ваги рідкісним словам, таким чином припускаючи, що вони є індикаторами унікальності.

Після цього виконується стемінг – шукається початкова основа слова та лематизація – слово приводиться до своєї нормальної форми, а саме виконується вилучення основи слова. Наступним кроком є позбавлення тексту від стоп-слів, слів які зустрічаються в текстах найчастіше, різні сполучники, прийменники, займенники, слова що часто повторюються або ж навпаки слова які зустрічаються дуже рідко. Видалення стоп-слів проводиться задля того, щоб зменшити розмір словника.

Після попередніх кроків робляться певні припущення, такі як: текст є згенерована ймовірнісним розподілом послідовність слів і порядок слів у цій послідовності не впливає на тематику. Інше припущення базується на тому, що всі слова належать до певної теми. Останнє припущення – документ містить невелику кількість тем, а теми у свою чергу характеризуються невеликою кількістю ключових слів.

Після того як попередня обробка тексту завершена, у тексті виділяються ознаки за допомогою представлення слів у векторній формі. Це означає, що всі слова будуть перекодовані у числові вектори певної довжини. Головною ідеєю представлення слів у векторній формі є те, що над словами можна виконувати певні математичні операції і таким чином отримувати нові семантичні значеняя. Зазвичай у задачах семантичного аналізу використовується метод bag of words. Цей метод будує вектор з розмірністю що дорівнює довжині текста. Також вказується або бінарна ознака слова - наявне дане слово чи ні, або рахується скільки разів це слово зустрічається в тексті.

Після застосування bag of words використовується метод TF-IDF. Суть TF-IDF у тому, що робиться розрахунок відносної частоти кожного слова у тексті. TF рахується як частка використання конкретного слова від загального обсягу тексту. IDF рахується як частка кількості текстів від текстів в яких зустрічалося конкретне слово.

Перемноження цих двох чисел призводить до оцінки TF-IDF слова в документі. Чим вищий бал, тим релевантніше це слово в цьому конкретному документі. Говорячи більш формально математичними словами, оцінка TF-IDF для слова t в документі d з набору документів D розраховується наступним чином:

$$tf\ idf\ (t, d, D) = tf\ (t, d)\ .\ idf\ (t, D)$$

$$tf\ (t, d) = log\ (1 + freq\ (t, d))$$

$$idf\ (t, D) = log\ (\ \frac{N}{count\ (d \in D{:}t \in d)}\ )$$

Наступним кроком є безпосереднє вирішення задачі класифікації чи кластеризації. Відмінність цих двох методів полягає у тому, що на відміну від класифікації, де класи відомі, у кластеризації класи невідомі. Вони стають відомими безпосередньо в результаті навчання моделі. Для того щоб алгоритм кластеризації працював максимально ефективно, зразки одного класу мають бути схожими один на одного і при цьому досить вагомо відрізнятися від зразків інших класів.

Досить часто у задачах класифікації застосовується класифікатор Баєса. Оптимальний класифікатор Байєса є імовірнісною моделлю, яка робить найбільш вірогідний прогноз для нового прикладу. Він описується за допомогою теореми Байєса, яка забезпечує принциповий спосіб обчислення умовної ймовірності. Він також тісно пов’язаний з апостеріорним максимумом: імовірнісною структурою, званою MAP, яка знаходить найбільш імовірну гіпотезу для набору навчальних даних. На практиці оптимальний класифікатор Байєса є дорогим з точки зору обчислень і замість цього для наближенного обчислення результату можна використовувати такі спрощення, як алгоритм

Гіббса та наївний Байєс. Наївний Баєс припускає що ознаки в даних не корелюють одна з одною, тобто є незалежними.
Основними недоліками даного класифікатора є факт того, що умовна незалежність ознак не завжди виконується. У більшості ситуацій вони демонструють певну форму залежності. Та так звана проблема з нульовою ймовірністю: коли ми зустрічаємо слова в даних тесту для певного класу, яких немає в навчальних даних, ми можемо отримати нульову ймовірність класу.

Алгоритми кластеризації бувають багатьох типів. Перший з них - методи на основі розподілу: це кластеризаційна модель, в якій ми підбираємо дані про ймовірність того, що вони можуть належати до того самого розподілу. Зроблене групування може бути звичайним або гаусовим. Гауссовий розподіл є більш помітним, коли ми маємо фіксовану кількість розподілів, і всі майбутні дані вписуються в нього таким чином, що розподіл даних може бути максимальним.

Наступним є метод на основі підключення - основна ідея моделі на основі зв'язку схожа на модель на основі центроїда, яка в основному визначає кластери на основі близькості точок даних. Тут ми працюємо над ідеєю, що точки даних, які знаходяться ближче, мають подібну поведінку порівняно з точками даних, які знаходяться далі. Це не єдине розділення набору даних, натомість воно забезпечує широку ієрархію кластерів, які зливаються один з одним на певних відстанях. Тут вибір функції відстані суб’єктивний. Ці моделі дуже легко інтерпретувати, але їм бракує масштабованості. [9]

Кластеризація підпростору: це навчання без нагляду, яке спрямовано на групування точок даних у кілька кластерів, щоб точки даних в одному кластері лежали у низьковимірному лінійному підпросторі. Алгоритм кластеризації підпростору локалізує пошук релевантних вимірів і дозволяє знайти кластер, який існує в декількох підпросторах, що перекриваються. Кластеризація підпросторів була створена для вирішення дуже специфічних проблем

комп’ютерного зору, які мають об’єднання структури підпростору в даних, але вона привертає все більше уваги серед статистики та спільноти машинного навчання. Люди використовують цей інструмент у соціальних мережах, рекомендаціях фільмів та наборах біологічних даних. Існують дві гілки кластеризації підпростору на основі їх стратегії пошуку.

Алгоритми зверху вниз знаходять початкову кластеризацію в повному наборі вимірів і оцінюють підпростір кожного кластера.
Підхід знизу вгору знаходить щільну область у низьковимірному просторі, а потім об’єднує, утворюючи кластери.

Методи на основі центроїдів: це один з алгоритмів ітераційної кластеризації, в якому кластери формуються за рахунок близькості точок даних до центру кластерів. Тут центр кластера, тобто центроїд, утворений таким чином, що відстань точок даних є мінімальною від центру. Таким чином, рішення зазвичай апроксимуються під час ряду випробувань. Алгоритм к-середніх є одним із найбільш популярних прикладів цього алгоритму. Найбільша проблема методів на основі центроїдів полягає в тому, що нам потрібно вказати K заздалегідь. Вони також мають проблеми з кластеризацією розподілів на основі щільності. [10]

Окрім класичних методів машинного навчання, набули популярності та широкої вживаності методи глибинного навчання – нейронні мережі. Нейронні мережі показують високу точність на великих наборах даних та є досить гнучкими, завдяки можливості змінювати архітектуру в залежності від конкретної задачі. З іншого боку, недоліком використання нейронних мереж є висока ресурсоємність під час навчання – час, високі вимоги до обчислювальних потужностей, можуть потребувати великих об’ємів пам’яті.

РОЗДІЛ 2 АНАЛІЗ ТА ПОПЕРЕДНЯ ОБРОБКА ДАТАСЕТА

2.1 Аналіз датасета

На відміну від різноманіття датасетів для найпопулярніших мов, для української мови, нажаль, ситуація гірша. У вільному доступі було знайдено два підготовлених датасети. Один з них від The M-AILABS Speech Dataset. Він базується на аудіокнигах та налічує вісімдесят сім годин аудіо даних.

Другий датасет зібраний українською спільнотою спеціалістів з розпізнавання мовлення. Цей датасет містить декілька джерел даних, а саме: звукові доріжки з різних програм телебачення, аудіокниги, звукові доріжки з україномовних ютуб каналів та інші джерела. Датасет налічує приблизно одну тисячу двісті годин аудіо даних у форматі wav. Окрім цього, для датасета було додатково створено словник з усіма лейблами для кожної звукової дорожки.

Звісно завжди можна створити власний датасет, але в такому випадку існує шанс стикнутися з рядом проблем як от: обмежена кількість даних у вільному доступі, розрізненість даних великі часові затрати для підготовки датасета.

Так як поставлена у цій роботі задача, не є дуже специфічною, доцільніше використати готовий датасет, ніж збирати власний.

Отже далі буде проведено ознайомлення з обраним датасетом. Робота з даними буде виконуватись за допомогою Python у хмарному середовищі Google Colab. Основними бібліотеками для аналізу даних будуть numpy та pandas.

Нижче можемо бачити загальний вигляд файлу у якому зберігаються усі лейбли.

```
data=pd.read_csv('Ukrainian_Open_Speech_To_Text Dataset.csv')
print(data)
```

```
                                                      path  \
0        ../input/ukrainian-open-speech-to-text-dataset...
1        ../input/ukrainian-open-speech-to-text-dataset...
2        ../input/ukrainian-open-speech-to-text-dataset...
3        ../input/ukrainian-open-speech-to-text-dataset...
4        ../input/ukrainian-open-speech-to-text-dataset...
...                                                    ...
2171203  ../input/ukrainian-open-speech-to-text-dataset...
2171204  ../input/ukrainian-open-speech-to-text-dataset...
2171205  ../input/ukrainian-open-speech-to-text-dataset...
2171206  ../input/ukrainian-open-speech-to-text-dataset...
2171207  ../input/ukrainian-open-speech-to-text-dataset...

                                text        dataset  \
0                    я на хвилинку             11/
1                     віддати тобі             11/
2                  назавжди тисяча             11/
3                        є с батька             11/
4                    по-перше не             11/
...                              ...             ...
2171203          боях над братською  TOLOKAFILMS/
2171204  могилою колишнього союзу  TOLOKAFILMS/
2171205                 доведеться  TOLOKAFILMS/
2171206               відбудовувати  TOLOKAFILMS/
2171207               над ними влади  TOLOKAFILMS/
```

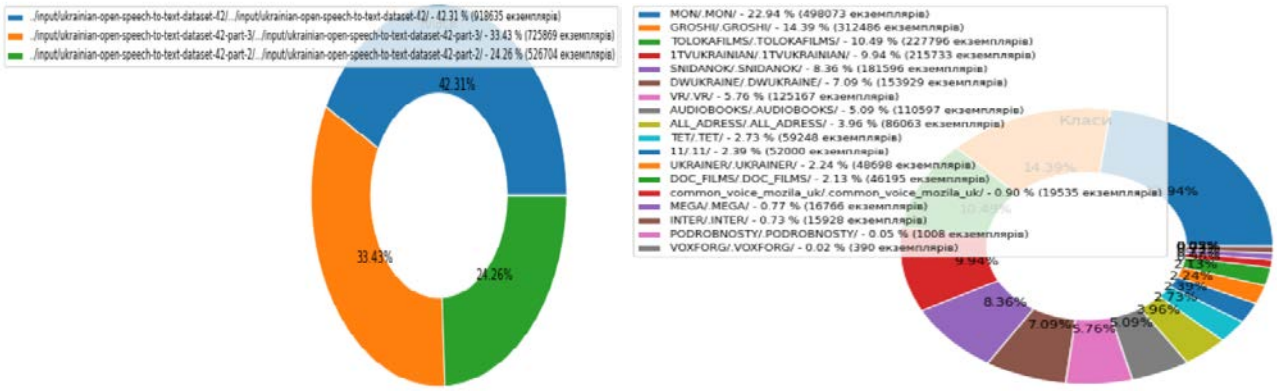
Далі дослідимо з яких джерел даних складається датасет, та у якому співвідношенні дані розподілені між цими джерелами

```
Main_datasets=data['kaggle_dataset'].unique()
print('main_datasets:',main_datasets)
main_datasets = {main_datasets[i] : main_datasets[i] for i in range(0, len(m
ain_datasets) ) }

datasets=data['dataset'].unique()
print('datasets:',datasets)
datasets = { datasets[i] : datasets[i] for i in range(0, len(datasets) ) }
```

```
main datasets: ['../input/ukrainian-open-speech-to-text-dataset-42/'
 '../input/ukrainian-open-speech-to-text-dataset-42-part-2/'
 '../input/ukrainian-open-speech-to-text-dataset-42-part-3/']
datasets: ['11/' '1TVUKRAINIAN/' 'ALL_ADRESS/' 'DOC_FILMS/' 'DWUKRAINE/' 'GROSHI/'
 'INTER/' 'MEGA/' 'common_voice_mozila_uk/' 'AUDIOBOOKS/' 'PODROBNOSTY/'
 'SNIDANOK/' 'TET/' 'UKRAINER/' 'VOXFORG/' 'VR/' 'MON/' 'TOLOKAFILMS/']
```

```
data_class_display(dataframe=data,class_column='kaggle_dataset',expl_lables=
kaggle_datasets)
```

```
data_class_display(dataframe=data,class_column='dataset',expl_lables=dataset
s)
```



Малюнок 6

У наступному коді ми обробляємо лейбли, а саме проводимо токенізацію, приводимо слово до початкової форми, та видаляємо стоп-слова

```
def ua_tokenizer(text,ua_stemmer=True,stop_words=[]):
    tokenized_list=[]
    text=re.sub(r"""['’"`�]""", '', text)
    text=re.sub(r"""([0-9])([\u0400-\u04FF]|[A-z])""", r"\1 \2", text)
    text=re.sub(r"""([\u0400-\u04FF]|[A-z])([0-9])""", r"\1 \2", text)
    text=re.sub(r"""[\-.,:+*/_]""", ' ', text)
    for word in nltk.word_tokenize(text):
        if word.isalpha():
            word=word.lower()
            if ua_stemmer is True:
                word=UkrainianStemmer(word).stem_word()
            if word not in stop_words:
                tokenized_list.append(word)
    return tokenized_list
```

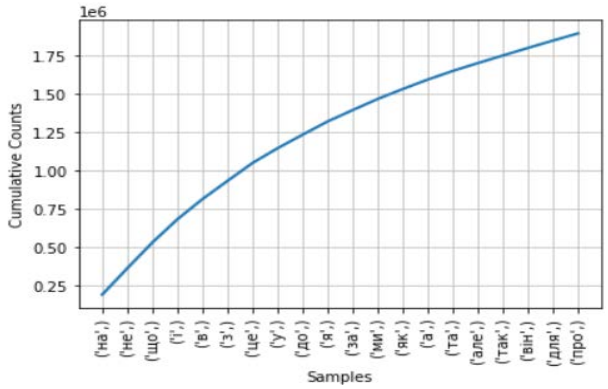
За допомогою даної функції можна буде проаналізувати датасет на різних рівнях на найбільш вживані слова та словосполучення

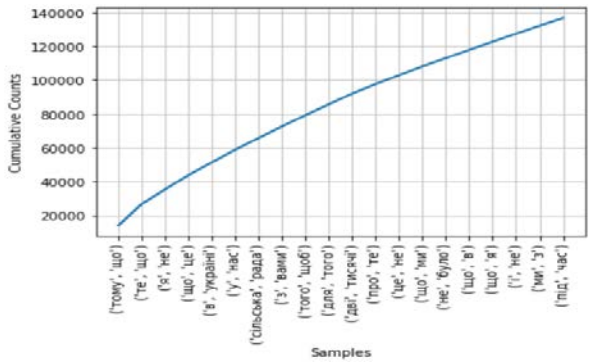
```
def ngrams_info(series,n=1,most_common=20,ua_stemmer=True,stop_words=[]):
    print (n,'- grams')
    print ('ua_stemmer:',ua_stemmer)
    words=series.str.cat(sep=' ')
    print ('Кількість символів: ',len(words))
    words=nltk.ngrams(ua_tokenizer(words,ua_stemmer=ua_stemmer,stop_words=st
op_words),n)
    words=nltk.FreqDist(words)
    print ('Кількість токенів: ',words.N())
    print ('Кількість унікальних токенів: ',words.B())
    print ('Найбільш уживані токени: ',words.most_common(most_common))
    words.plot (most_common, cumulative = True)
```

```
for n in (1,2):
    ngrams_info(data['text'],n=n,ua_stemmer=False)
```

На даних діаграмах видно найбільш вживані слова та словосполучення при загальному аналізі датасету. Очікуваним результатом є те що, більшість цих слів є частками, сполучниками, прийменниками.

```
1 - grams
ua_stemmer: False
Кількість символів:  58897395
Кількість токенів:  9287947
Кількість унікальних токенів:  293149
Найбільш уживані токени:  [(('на',), 187816), (('не',), 172010), (('що',), 168306),
(('і',), 152326), (('в',), 131841), (('з',), 118902), (('це',), 117474), (('у',), 9
6428), (('до',), 88522), (('я',), 86789), (('за',), 74187), (('ми',), 72897), (('як
',), 64848), (('а',), 61960), (('та',), 56385), (('але',), 51347), (('так',), 49644
), (('він',), 49275), (('для',), 48336), (('про',), 46852)]

2 - grams
ua_stemmer: False
Кількість символів:  58897395
Кількість токенів:  9287946
Кількість унікальних токенів:  3417312
Найбільш уживані токени:  [(('тому', 'що'), 14209), (('те', 'що'), 12671), (('я', '
не'), 8650), (('що', 'це'), 8441), (('в', 'україні'), 7544), (('у', 'нас'), 7472),
(('сільська', 'рада'), 6914), (('з', 'вами'), 6858), (('того', 'щоб'), 6589), (('дл
я', 'того'), 6431), (('дві', 'тисячі'), 6373), (('про', 'те'), 5609), (('це', 'не')
, 5265), (('що', 'ми'), 5181), (('не', 'було'), 4866), (('що', 'в'), 4863), (('що',
'я'), 4799), (('і', 'не'), 4740), (('ми', 'з'), 4669), (('під', 'час'), 4546)]
```



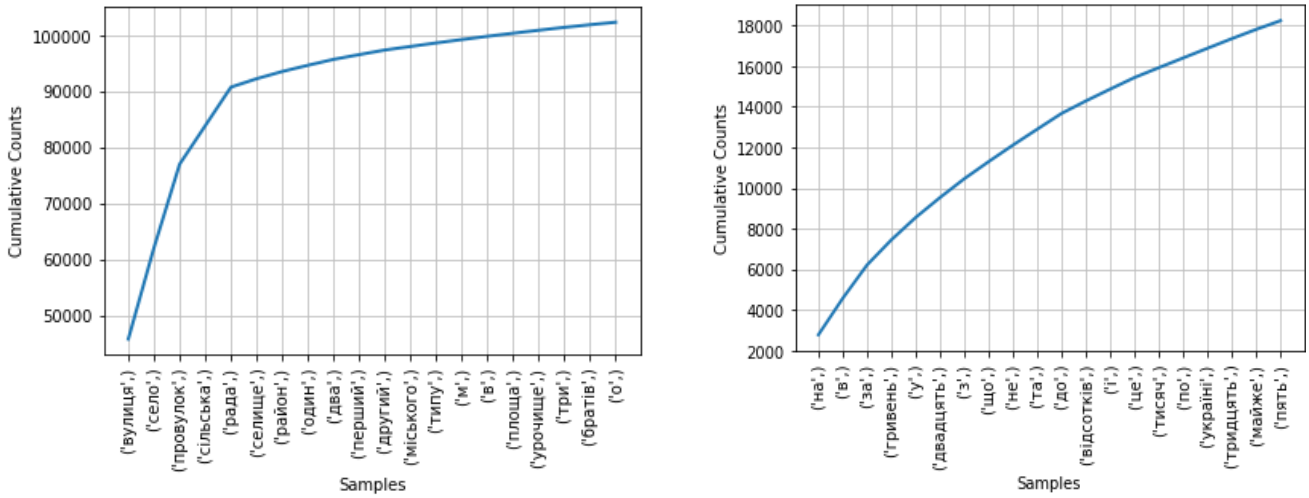


Малюнок 7

Якщо ж проаналізувати датасет зі специфічною лексикою, як от ALL_ADRESS, можна побачити зовсім інший розподіл.

```
ALL_ ADRESS/
1 - grams
ua_stemmer: False
Кількість символів:  1762601
```

```
Кількість токенів:  217347
Кількість унікальних токенів:  59688
Найбільш уживані токени:  [(('вулиця',), 45790), (('село',), 16330), (('провулок',)
, 14929), (('сільська',), 6887), (('рада',), 6887), (('селище',), 1503), (('район',
), 1293), (('один',), 1109), (('два',), 1061), (('перший',), 848), (('другий',), 82
3), (('міського',), 634), (('типу',), 634), (('м',), 600), (('в',), 586), (('площа'
,), 547), (('урочище',), 538), (('три',), 515), (('братів',), 469), (('о',), 449)]

INTER/
1 - grams
ua_stemmer: False
Кількість символів:  560919
Кількість токенів:  81373
Кількість унікальних токенів:  10435
Найбільш уживані токени:  [(('на',), 2768), (('в',), 1796), (('за',), 1645), (('гри
вень',), 1237), (('у',), 1104), (('двадцять',), 982), (('з',), 927), (('що',), 842)
, (('не',), 816), (('та',), 784), (('до',), 777), (('відсотків',), 615), (('і',), 5
83), (('це',), 570), (('тисяч',), 486), (('по',), 479), (('україні',), 477), (('три
дцять',), 473), (('майже',), 456), (('пять',), 430)]
```



Малюнок 8

Загалом з даного дослідження можна припустити, що в подальшому, модель, яка буде навчатися на цих даних буде краще розпізнавати саме найбільш вживані токени та словосполучення. Якщо ми маємо на меті розроблення моделі для загального розпізнавання мовлення, тоді доцільніше буде навчати її на всіх даних.

Якщо ж наприклад ми будуємо модель, яка буде використовуватись в системах пов’язаних з навігацією, то матиме сенс використовувати дані ALL_ADRESS

2.2 Попередня обробка датасета

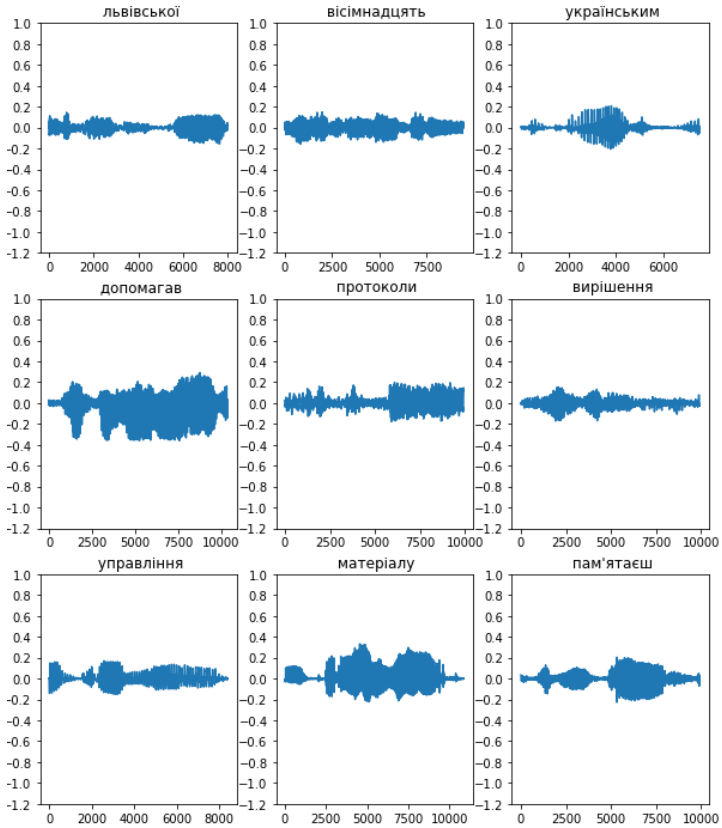
Першим кроком необхідно поділити датасет на три частини: дані для тренування моделі, валідації та тестування

```
val_size=0.2#0.2
test_size=0.1
from sklearn.model_selection import train_test_split

train_val_files_pd,test_files_pd,_,__= train_test_split(data,data['text'],
test_size=test_size,stratify=data['text'], random_state=10)

train_files_pd,val_files_pd,_,__= train_test_split(train_val_files_pd,trai
n_val_files_pd['text'],test_size=val_size,stratify=train_val_files_pd['tex
t'], random_state=11)

train_files=train_files_pd.to_numpy()
val_files=val_files_pd.to_numpy()
test_files=test_files_pd.to_numpy()

print('Training set size', len(train_files))
print('Validation set size', len(val_files))
print('Test set size', len(test_files))


1050
Training set size 735
Validation set size 210
Test ser size 105
```

Після цього починаємо процесс декодування аудіо у формат часового домену

```
def decode_audio(audio_binary):
  audio, _ = tf.audio.decode_wav(audio_binary)
  return tf.squeeze(audio, axis=-1)

def get_waveform_and_label(file_path):
    label = file_path[1]
    audio_binary = tf.io.read_file(file_path[0])
    print(audio_binary)
    waveform = decode_audio(audio_binary)
    return waveform, label
```

```
rows = 3
cols = 3
n = rows*cols
fig, axes = plt.subplots(rows, cols, figsize=(10, 12))
for i, (audio, label) in enumerate(waveform_ds.take(n)):
  r = i // cols
  c = i % cols
  ax = axes[r][c]
  ax.plot(audio.numpy())
  ax.set_yticks(np.arange(-1.2, 1.2, 0.2))
  label = label.numpy().decode('utf-8')
  ax.set_title(label)

plt.show()
```

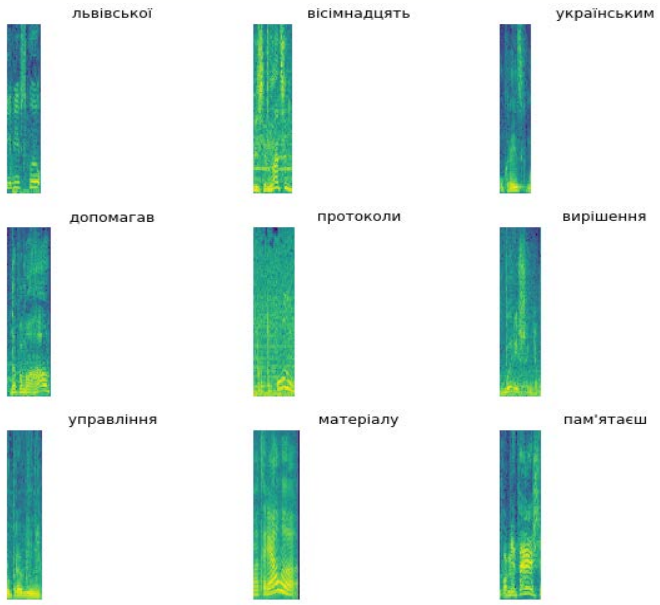


Малюнок 9

Коли маємо дані у вигляді амплітуда-час, можемо за допомогою віконного перетворення Фур'є, для кожного зразка отримати спектрограму, яка і буде в подальшому подаватися у якості вхідних даних для нейронної мережі[11, 12]

```
def get_spectrogram(waveform):
  print (([50000] -tf.shape(waveform))<0)
  if ([50000] -tf.shape(waveform))<0:
```

```
    waveform=tf.slice(waveform, [0], [50000])
  zero_padding = tf.zeros([50000] - tf.shape(waveform), dtype=tf.float32)

  waveform = tf.cast(waveform, tf.float32)
  equal_length = tf.concat([waveform, zero_padding], 0)
  spectrogram = tf.signal.stft(
      equal_length, frame_length=255, frame_step=128)

  spectrogram = tf.abs(spectrogram)

  return spectrogram
```



Малюнок 10

Власно кажучи, нижче представлена функція, яка й буде виконувати всі перелічені вище кроки для підготовки даних.

```
def preprocess_dataset(files):
  files_ds = tf.data.Dataset.from_tensor_slices(files)
  output_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTU
NE)
  output_ds = output_ds.map(
      get_spectrogram_and_label_id,  num_parallel_calls=AUTOTUNE)
  return output_ds
```

РОЗДІЛ 3 ТРЕНУВАННЯ НЕЙРОННОЇ МЕРЕЖІ

3.1 Вибір архітектур нейронної мережі для поставленої задачі

На сьогоднішній день існує дуже багато архітектур нейронних мереж. Деякі з них можуть бути навчені на власному комп’ютері за годину, а інші потребують використання потужних обчислювальних процесорів впродовж значного часу. Варто також не забувати про навчені нейронні мережі, які готові можна одразу використовувати або ж перенавчити під свої потреби. Різні архітектури можуть по різному демонструвати рівень точності у залежності від того для якої задачі вони застосовуються. До того ж, в залежності від підходу до розв’язання задачі, представлення даних та власне самих даних результати використання певних архітектур також можуть суттєво відрізнятись.

Також не варто забувати про ресурсні можливості дослідника, а саме, наявність пам’яті та обчислювальних потужностей. Адже обравши надто глибоку нейронну мережу можна зіштовхнутись з великими часовими затратами, при недостатній потужності процесорів, а враховуючі той факт, що для досягнення високих показників процес тренування нейронної проходить у декілька ітерацій навчання – маємо значне збільшення тривалості тренування.

Окрім того не варто забувати про пам’ять, чим більше параметрів буде в моделі, тим більше пам’яті буде необхідно. Тож при побудові моделі варто адекватно оцінювати кількість необхідних параметрів.

Підсумуємо сказане вище, та зробимо список факторів, які будуть впливати на те, якою буде побудована модель:

- Рід задачі

- Підхід до її розв’язання
- Дані в датасеті
- В якому вигляді дані будуть подаватись в модель
- Наявність обчислювальних ресурсів
- Обмеження в пам’яті

Опишемо кожен з пунктів для поставленої задачі:

Рід задачі – переклад мовлення в текстовий формат

Підхід до її розв’язання – застосування нейронної мережі до фрагментів аудіо-запису, де фрагмент це вимовлення одного слова. Використання ідеї мультикласової класифікаціїї: нейронна мережа буде обраховувати ймовірність належності аудіо-фрагменту до того чи іншого класу, де класи – це заздалегідь визначений набір слів.

Дані в датасеті – датасет налічує тридцять чотири класи, тобто нейронна мережа буде розрізняти тридцять чотири різних слова. Загальний розмір датасета п’ятсот шістдесят шість аудіо-зразків.

В якому вигляді дані будуть подаватись в модель – спочатку аудіо-файли конвертуються у векторне представлення методами TensorFlow, після цього з цих векторів створюються спектрограми методом віконного перетворення Фур’є, з шириною вікна двісті п’ятдесят п’ять та кроком сто двадцять вісім.

Наявність обчислювальних ресурів – тренування моделі проводиться в Google Colab Pro з графічними процесорами NVIDIA P100, NVIDIA T4 і об’ємом оперативної пам’яті двадцять п’ять гігабайтів. Проте існує часове обмеження – після безперервного використання ресурів протягом двадцяти чотирьох годин, сесія буде автоматично зупинена.

Обмеження в пам’яті –

1) Обмеження під час безпосередньо навчання моделі - датасет завантажується з гугл диска, де кожний користувач має п’ятнадцять гігабайтів доступної

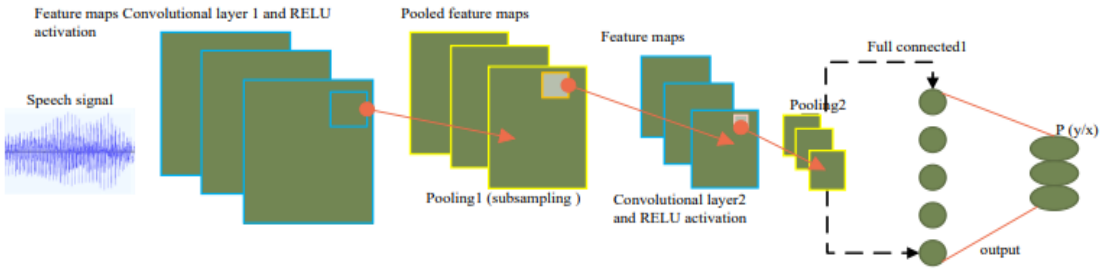
пам’яті. З цього витікає, що розмір датасету не може перевищувати п’ятнадцять гігабайтів.

2) Обмеження під час використання навченої моделі - при використанні моделі в програмному засобі чим менше пам’яті вона буде займати, тим краще буде. Тобто тут постає питання балансу між якістю моделі та об’ємом місця, яке вона буде займати.

Фактично, якщо подивитись на кінцевий результат препроцесингу даних – маємо роботу з зображеннями, а для цього найбільш оптимальними є згорткові нейронні мережі.[13]

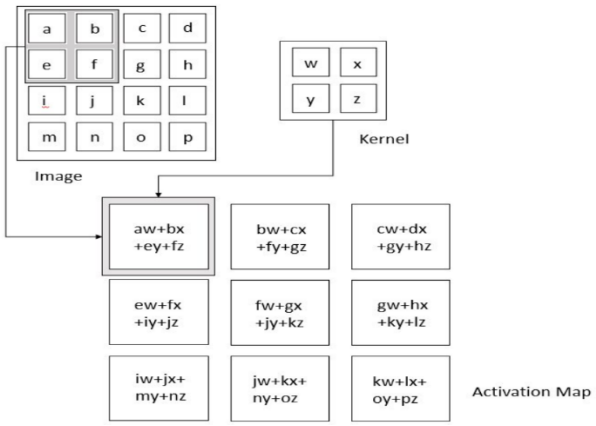
При цьому з іншого боку маємо зображення певних характеристик у часі, тобто можемо припускати, що дані на певному часовому проміжку певною мірою залежать від даних з попереднього часового проміжку. Для задач такого роду, коли треба враховувати попередні дані використовуються рекурентні нейронні мережі.

Типова архітектура згорткової нейронної мережі складається з кількох згорткових та агрегувальних шарів із повністю з’єднаними шарами для класифікації. [14]



Малюнок 11

Згортковий шар відповідає за основну частину обчислювального навантаження мережі. Під час прямого проходу ядро ковзає вздовж зображення, створюючи двовимірне представлення зображення, відоме як мапа активації.



Малюнок 12

Подібно до згорткового шару, агрегувальний шар відповідає за зменшення просторового розміру згорнутого елемента. Таким чином, за рахунок зменшення розмірності відбувається зниження кількості обчислень, що в свою чергу впливає на швидкість навчання мережі. Крім того, агрегувальний шар є корисним для виділення домінуючих ознак, виділення яких дозволяє ефективно навчати модель.

Існує два типи агрегації: максимум та середнє значення. Максимальний пул повертає максимальне значення з частини зображення, охопленої ядром. З іншого боку, середній пул повертає середнє значення всіх значень з частини зображення, охопленої ядром.

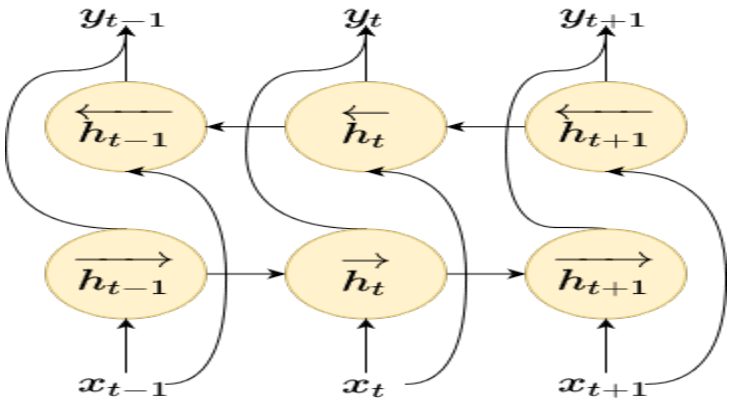
Максимальна агрегація також виконує роль шумозаглушувача. Вона видаляє шуми разом із зменшенням розмірності. З іншого боку, середня агрегація просто виконує зменшення розмірності. Отже, ми можемо сказати, що у більшості випадків максимальна агрегація працює набагато краще, ніж середня.

Додавання повністю з’єднаного шару є дешевим способом вивчення нелінійних комбінацій ознак, отриманих від згорткових шарів. Після того як, вхідне зображення було перетворено у відповідну форму для багаторівневого персептрона, воно зрівнюється у вектор-стовпець. Зведений вектор подається в

нейронну мережу з прямим зв’язком і зворотне поширення застосовується на кожній ітерації навчання. Протягом ряду епох модель навчається розрізняти домінуючі та певні низькорівневі ознаки в зображеннях та класифікувати зображення на основі цих ознак.

Рекурсивні нейронні мережі виконують обчислення в послідовності часу, так як їх поточний прихований стан залежить від усіх попередніх прихованих станів. Точніше, вони розроблені для моделювання сигналів часових рядів, а також для визначення довгострокових і короткострокових залежностей між різними часовими проміжками вхідних даних.[15]

Що стосується програм розпізнавання мовлення, то вхідний сигнал передається через рекуренту мережу для обчислення прихованих послідовностей. Одним з основних недоліків простої форми такої моделі є те, що вона генерує наступний вихід тільки на основі попереднього контексту.



Малюнок 13

Рекурентні мережі обчислюють послідовність прихованих векторів як:

$$h_t = H\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$
$$y_t = W_{hy}h_t + b_y,$$

де W — ваги, b — вектори зміщення, а H — нелінійна функція.

Однак у розпізнаванні мовлення зазвичай інформація майбутнього контексту є такою ж важливою, як і минулий контекст. Ось чому замість використання

однонаправленої мережі зазвичай вибирають двонаправлені моделі, щоб усунути цей недолік. Двонаправлені нейронні мережі обробляють вхідні вектори в обох напрямках, тобто вперед і назад, і зберігають приховані вектори стану для кожного напрямку, як показано на малюнку вище.[16]

3.2 Результати використання архітектур та вибір найоптимальнішої з них

У рамках дослідження було протестовано дві архітектури на трьох варіантах датасету.

Тестувалися згорткова нейронна мережа та двонаправлена рекурентна нейронна мережа.

Датасети складалися зі ста, тридцяти чотирьох та шести класів.

У якості метрики використовувалась sparce categorical accuracy. Це метрика яка зазвичай застосовується у задач мультикласової класифікації, де лейбли представлені одним числом. Рахується вона як відношення кількості вірно визначених зразків до загальної кількості зразків.

Втрати визначались через sparce categorical crossentropy. Рахується вона як:

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

У якості оптимізатора використовувався Адам.[17]

Нижче будуть продемонстровані згорткова та двонаправлена рекурентна моделі

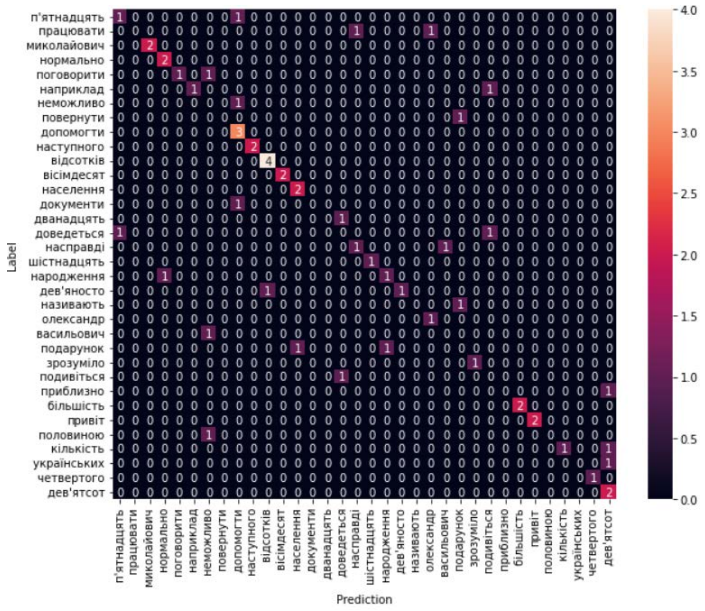
```
Input shape: (389, 129, 1)
num_labels: 34
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 resizing (Resizing)         (None, 96, 96, 1)         0

 normalization (Normalizatio  (None, 96, 96, 1)        3
 n)

 conv2d (Conv2D)             (None, 94, 94, 512)       5120

 max_pooling2d (MaxPooling2D  (None, 47, 47, 512)      0
 )

 dropout (Dropout)           (None, 47, 47, 512)       0

 conv2d_1 (Conv2D)           (None, 45, 45, 256)       1179904

 max_pooling2d_1 (MaxPooling  (None, 22, 22, 256)      0
 2D)

 dropout_1 (Dropout)         (None, 22, 22, 256)       0

 flatten (Flatten)           (None, 123904)            0

 dense (Dense)               (None, 256)               31719680

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 34)                8738

=================================================================
Total params: 32,913,445
Trainable params: 32,913,442
Non-trainable params: 3


Layer (type)                Output Shape              Param #
=================================================================
bidirectional_23 (Bidirectio (None, 50, 256)          285696
_________________________________________________________________
dropout_34 (Dropout)        (None, 50, 256)           0
_________________________________________________________________
bidirectional_24 (Bidirectio (None, 256)              395264
_________________________________________________________________
dropout_35 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_23 (Dense)            (None, 128)               32896
_________________________________________________________________
dropout_36 (Dropout)        (None, 128)               0
_________________________________________________________________
dense_24 (Dense)            (None, 80)                10320
=================================================================
Total params: 724,176
Trainable params: 724,176
Non-trainable params: 0
```

Для перших двох датасетів де було сто та тридцять чотири класи були отримані досить невисокі значення точності. У датасеті на сто класів загальна кількість зразків була тисяча п’ятдесят при умові, що на клас приходиться мінімум п’ять зразків. У другому на тридцять чотири класи було п’ятсот сімдесят шість зразків за умови, що для кожного класу є мінімум десять зразків.

Основна проблема навчання на цих датасетах була в оверфітингу, бо вони були сильно незбалансовані і моделі навчалися класифікувати одні класи набагато краще або гірше за інші.

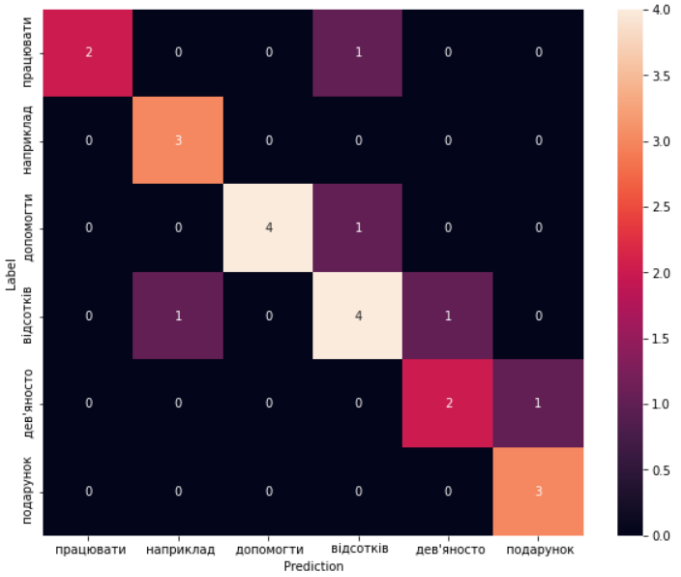
Нижче можемо бачити матрицю невідповідностей для тридцяти чотирьох класів:



Малюнок 14

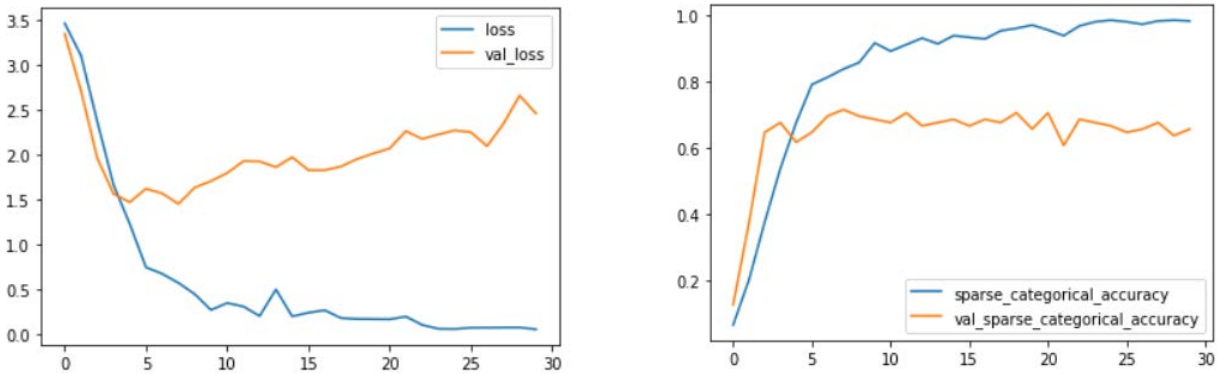
Тому для того щоб вирішити цю проблему було вирішено істотно зменшити кількість класів та зробити їх максимально збалансованими. Таким чином було отримано датасет на шість класів з загальною кількістю зразків сто п’ятдесят шість, при умові що в одному класі буде не менше двадцяти зразків.

Для цього датасету отримали наступні результати навчання моделей:



Малюнок 15

	precision	recall	f1-score	support
0	1.000000	0.666667	0.800000	3.000000
1	0.750000	1.000000	0.857143	3.000000
2	1.000000	0.800000	0.888889	5.000000
3	0.666667	0.666667	0.666667	6.000000
4	0.666667	0.666667	0.666667	3.000000
5	0.750000	1.000000	0.857143	3.000000
accuracy	0.782609	0.782609	0.782609	0.782609
macro avg	0.805556	0.800000	0.789418	23.000000
weighted avg	0.804348	0.782609	0.782057	23.000000



Малюнок 16

При цьому точність згорткової мережі склала шістдесят сім відсотків, а двонаправленої рекурентної шістдесят два відсотки.

З дослідження можемо зробити висновок, що найоптимальнішою є згорткова мережа, та що для досягнення високо рівня точності варто мати датасет більших розмірів та в частості більшої кількості зразків для кожного класу.

РОЗДІЛ 4 ПРОЕКТУВАННЯ ПРОЦЕДУРИ ПОШУКУ ЗА ТЕКСТОВИМ ЗАПИТОМ

Ідея пошуку аудіо-файлів за користувацьким текстовим запитом має у якості фундаменту базу, в якій будуть зберігатися аудіо-файли та їх зміст у текстовому форматі. Від користувача робиться запит і повертається аудіо-файл, який мав співпадіння з запитом.

Задачу пошуку можна розв’язувати багатьма шляхами, починаючі з примітивного варіанту порівняння з усіма наявними в базі текстами, що є повільним та неефективним пошуком та закінчуючи використанням різних технік семантичного пошуку з першого розділу.

Будемо проектувати систему, у якій використовується векторне представлення слів.[18] Такий підхід до задачі дозволить суттєво прискорити пошук та зробити його більш інтелектуальним та гнучким. Ще одним плюсом, в

залежності від реалізації, може бути зменшення обсягу пам’яті за рахунок того, що текст буде зберігатися у числовому форматі.

Проте у нашій схемі допустимо, що користувач також хоче мати текстовий зміст аудіо-файлу.

Таким чином у базі для кожного аудіо-файла має зберігатися його текстова версія, векторне представлення текстової версії, та загальна тематика тексту. Загальна тематика необхідна щоб на початку пошуку відкинути значну кількість варіантів, які не мають відношення до користувацького запиту, і проводити пошук все безпосередньо серед необхідних. Як варіант, замість загальної тематики можна використовувати ключові та найбільш значущі слова з тексту.

Сам пошук можна організувати через обчислення косинусу подібності[19] між запитом та шуканим текстом, спочатку за загальною тематикою а потім за записами зі знайденої тематики, або ж за ключовими словами. У якості результату повертати користувачу 3-5 найбільш підходящих аудіо-файлів з або без текстової версії.

Для створення подібної системи необхідно вибрати наступні інструменти:

- База в якій будуть зберігатися дані
- Бібліотека/фреймворк яка буде обчислювати косинус подібності
- Модель яка буде перетворювати текст у векторну форму
- Модель яка буде визначати загальну тематику тексту або знаходити ключові слова

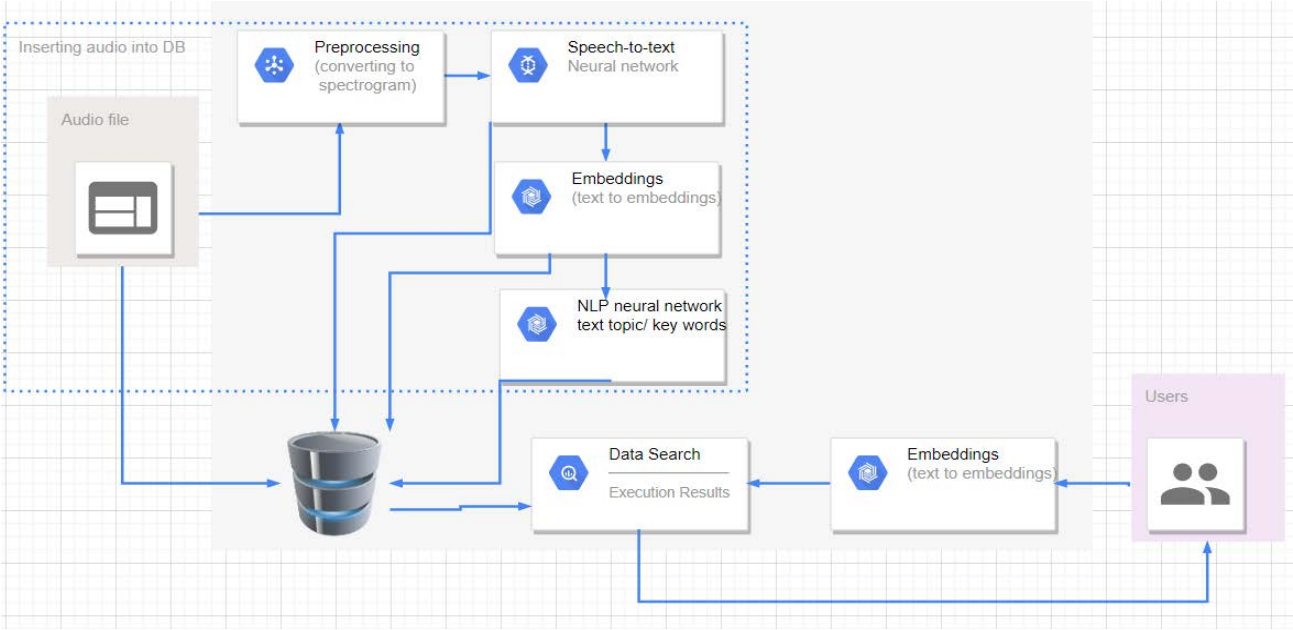
Так як планується зберігати аудіо-файли, в якості бази варто обрати NoSQL рішення, а конкретніше – MongoDB. Основними плюсами є її швидкість та гарна інтеграція з Python, основний мінус – це використання великих об’ємів пам’яті.

Потім треба буде обрати моделі для роботи з текстом. Можна використати вже існуючі або ж створити свої власні. Для перетворення тексту у векторну

форму. Станом на сьогодення існують готові ембедінги для української мови на основі Word2Vec, GloVe, LexVec від спільноти lang-uk.

І для обчислення косинуса подібності існує досить багато готових рішень, наприклад бібліотека torch.

Нижче наведена загальна схема системи:



Малюнок 17

ВИСНОВКИ

У даній роботі було розібрано ключові етапи створення нейронної мережі та проектування системи пошуку по текстовим запитам. Під час виконання роботи були вдало вирішені такі проблеми:

Пошук необхідних для тренування нейронної мережі даних та раціональне формування датасету, подальша обробка та підготовка датасету.

Розробка архітектури, підбір параметрів та тренування моделі на малому датасеті.

Цей програмний продукт може бути покращено побудовою більш точної моделі, яка буде тренована на датасеті з більшою кількістю даних, або ж розширенням простору слів, які розпізнає модель. У перспективі нейронна

мережа може бути перенавчена на розпізнавання аудіозаписів, які складаються зі словосполучень або навіть речень.

З точки зору системи пошуку аудіозаписів, може бути впроваджений функціонал для пошуку через голосові команди або ж розширені параметри текстового пошуку на вибір користувача як от: пошук конкретного співпадіння запиту та текстового відображення аудіо або ж пошук близьких за значенням зразків. Також можливим впровадженням може стати створення особистих кабінетів користувачів у яких вони зможуть відстежувати історію пошуків та додавати в існуючу базу власні аудіофайли, або ж мати свою добірку аудіофайлів серед якої потім будуть проводити пошук

Після належного покращення діагностичних складових та функціоналу, проект може бути випущений, як комерційний продукт, який може використовуватися у стрімінгових додатках для музики або аудіокниг або ж у великих компаніях які зберігають частину даних в аудіоформаті, як от наприклад записи розмов з клієнтами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SPECTRAL AUDIO SIGNAL PROCESSING JULIUS O. SMITH III Center for Computer Research in Music and Acoustics (CCRMA) Режим доступу до веб-сторінки: https://ccrma.stanford.edu/~jos/sasp/sasp.html
2. J. B. Allen, ``Short term spectral analysis, synthesis, and modification by discrete Fourier transform," IEEE Transactions on Acoustics, Speech, Signal Processing, vol. ASSP-25, pp. 235-238, June 1977.
3. J. B. Allen and L. R. Rabiner, ``A unified approach to short-time Fourier analysis and synthesis," Proc. IEEE, vol. 65, pp. 1558-1564, Nov. 1977.
4. PREPROCESSING TECHNIQUE IN AUTOMATIC SPEECH RECOGNITION Yakubu A. Ibrahim, Juliet C. Odiketa, Tunji S. Ibiyemi Режим доступу до веб-сторінки : https://anale-

informatica.tibiscus.ro/download/lucrari/15-1-23-Ibrahim.pdf

5. https://wiki.aalto.fi/display/ITSP/Windowing
6. B. C. Moore, An introduction to the psychology of hearing. Brill, 2012.
7. S. F. Dodge and L. J. Karam, “Understanding how image quality affects deep neural networks,” CoRR, vol. abs/1604.04004, 2016.
8. Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013, 2013.
9. Гитис Л. Кластерный анализ в задачах классификации, оптимизации и прогнозирования
10. Тараскина А. Нечеткая кластеризация по модифицированному методу ссредних и ее применение для обработки микрочиповых данных
11. Role of Windowing Techniques in Speech Signal Processing For Enhanced Signal Cryptography Aparna Ramdoss Режим доступу до веб-сторінки : https://www.researchgate.net/publication/323127358_Role_of_Windowing_Techniques_in_Speech_Signal_Processing_For_Enhanced_Signal_Cryptography
12. Preference for 20-40 ms window duration in speech analysis Kuldip K. Paliwal, James G. Lyons and Kamil K. Wojcicki Режим доступу до веб-сторінки:https://maxwell.ict.griffith.edu.au/spl/publications/papers/icsps10_window.pdf
13. AUTOMATIC TAGGING USING DEEP CONVOLUTIONAL NEURAL NETWORKS Keunwoo Choi, Gyorgy Fazekas, Mark Sandler Режим доступу до веб-сторінки : https://arxiv.org/pdf/1606.00298.pdf
14. Speech Recognition using Convolution Deep Neural Networks Ayad

Alsobhani, Hanaa M A ALabboodi and Haider Mahdi Режим доступу до веб-сторінки:https://iopscience.iop.org/article/10.1088/1742-6596/1973/1/012166

15. Theory, Concepts and Methods of Recurrent Neural Networks and Soft Computing Jeremy Rogerson – 2015
16. Bidirectional recurrent neural networks M. Schuster, K.K. Paliwal
17. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.
18. Lebret, Rémi; Collobert, Ronan (2013). "Word Emdeddings through Hellinger PCA". Conference of the European Chapter of the Association for Computational Linguistics (EACL). Vol. 2014.
19. Text Similarity Measures in News Articles by Vector Space Model Using NLP Ritika Singh, Satwinder Singh Режим доступу до веб-сторінки: https://www.researchgate.net/publication/346772467_Text_Similarity_Measures_in_News_Articles_by_Vector_Space_Model_Using_NLP