Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів семантичного аналізу аудіодоріжок
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1

Біблий Д.В.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник д.т.н. проф. Четвериков Г.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет_Комп'ютерних наук_____
Кафедра_Програмної інженерії_____
Рівень вищої освіти – другий (магістерський)

Спеціальність _121 – Інженерія програмного забезпечення_____ (код і повна назва)
Тип програми освітньо-наукова програма_____
Освітня програма_Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Біблому Дмитру Валерійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів семантичного аналізу аудіодоріжок затверджена наказом університету від "___" _____20 ___ р. № _____ заповнюється вручну після отримання наказу
2. Термін подання студентом роботи до екзаменаційної комісії
18 травня 2020 р.
3. Вихідні дані до роботи розпізнавання мовлення, семантичний аналіз тексту, пояснювальна
записка_____
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів розпізнавання мовлення, методів семантичного аналізу._____

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецрозділ	д.т.н. проф. Четвериков Г.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1	Аналіз предметної галузі	01.03.2020	
2	Методи та алгоритми розпізнавання іменованих сутностей і аналізу емоційності текстів	12.03.2020	
3	Прототипування	20.03.2020	
4	Підготовка пояснювальної записки	26.03.2020	
5	Спецчастина	07.04.2020	
6	Підготовка презентації та доповіді	21.04.2020	
7	Попередній захист	28.04.2020	
8	Нормоконтроль, рецензування	29.04.2020	
9	Занесення диплома в електронний архів	05.05.2020	
10	Допуск до захисту у зав. кафедри	13.05.2020	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____ 2019 р.

Студент _____
(підпис)

Керівник роботи _____ __д.т.н. проф. Четвериков Г.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 66 сторінок, 16 рисунків, 3 додатки, 14 джерел інформації.

СЕМАНТИЧНИЙ АНАЛІЗ, ВІДЕОФАЙЛ, АУДІОДОРІЖКА, ТРАНСКРІБАЦІЯ, РОЗПІЗНАВАННЯ, ТЕКСТ, ПОШУК.

Об’єктом дослідження є процес визначення смислового навантаження та тематики мовлення в аудіодоріжках.

Метою роботи є розробка програмної системи для розпізнавання тексту з відео та подальшого його семантичного аналізу з метою виділення основних тегів відео та пошуку заданого моменту відео.

Методи розробки базуються на інструментах для створення веб-додатків на платформі .NET.

В результаті роботи було виконано програмну систему для розпізнавання мовлення та семантичного аналізу.

SEMANTIC ANALYSIS, VIDEO FILE, AUDIO TRACK, TRANSCRIPTION, RECOGNITION, TEXT, SEARCH.

The object of research is the process of determining the semantic load and the subject of speech in audio tracks.

The aim of the work is to develop a software system for text recognition from video and its subsequent semantic analysis in order to highlight the main video tags and search for a given moment of the video.

Development methods are based on tools for creating web applications on the .NET platform.

As a result, a software system for speech recognition and semantic analysis was implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	application program. interface (інтерфейс програм. додатків)
avi	Audio Video Interleave (чергування аудіо і відео)
DNN	Deep Neural Network
flv	Flash Video
FLAC	Free Lossless Audio Codec
G2P	Grapheme-To-Phoneme
GMM	Gaussian Mixture Models (модель суміші Гауссіан)
HMM	Hidden Markov Model (прихована марківська модель)
JSON	JavaScript Object Notation
MFCC	Mel Frequency Cepstral Coefficients
mp3	MPEG-3 (кодек третього рівня)
RBM	Restricted Boltzmann Machines (обмежені машини Больцмана)
SDK	Software Development Kit (комплект засобів розробки)
wav	Waveform Audio (у формі хвилі)
WFST	weighted finite-state transducer
WMP	Windows Media Player
ВПК	відкритий програмний код
ЗПК	закритий програмний код
МТ	мовні технології
ПЕОМ	персональна електронно-обчислювальна машина
ПЗ	програмне забезпечення

ВСТУП

Питаннями автоматичного розпізнавання мови вчені стали займатися з моменту появи перших комп'ютерів, так як текстовий інтерфейс взаємодії з ЕОМ не забезпечував прийнятною швидкості і природності роботи. На протязі багатьох років досліджень був розроблений широкий спектр комп'ютерних програм і методів, спрямованих на вирішення проблем розпізнавання мови.

На різних етапах розвитку дослідження розпізнавання мовлення мало різні цілі. Все на що була здатна перша система з розпізнавання – це зрозуміти названі людиною цифри. Наступним етапом стало розуміння певних команд. Наступним етапом розвитку систем розпізнавання мовлення стало перетворення людської мови в надрукований текст, які спочатку не давали гарного результату та з кожним днем системи стають все досконаліші та досконаліші.

В наш час наукове співтовариство вкладає гігантську кількість грошей в науково-дослідні розробки та ноу-хау для вирішення таких проблем як розуміння і автоматичне розпізнавання мовлення. Це також підсилюється через потреби в створенні систем комерційного та військового призначення, хоча автоматична обробка мови вже набула широкого вживання як у військовій та космічній техніці так і в різних сферах бізнесу. Незважаючи на всі старання тисяч науковців системи яка б працювала в оточенні шумів, наприклад в ресторанах, аеропортах та інших шумних громадських містах досі немає на ринку, тож питання розпізнавання мовлення як і раніше стоїть на часі.

Метою даної роботи є створення програмного сервісу для розпізнавання слів з відео або аудіофайлу наданого користувачем, та для подальшого пошуку збігів, обробки отриманої інформації, семантичним аналізом тексту та словником термінів для визначення тематики ресурсу.

Об’єктом дослідження є процес визначення смислового навантаження та тематики мовлення в аудіо доріжках.

Предметом дослідження є методи розпізнавання мовлення та семантичного аналізу аудіодоріжок, які можуть бути використані для створення системи семантичного аналізу аудіодоріжок.

У роботі досліджуються методи розпізнавання та семантичного аналізу мовлення які побудовані на алгоритмах машинного навчання, комп'ютерної лінгвістики та цифрової обробки сигналів.

Якщо поглянути на саме по собі розпізнавання мовлення то воно не представляє нічого нового, так як перші спроби розпізнавання мовлення буди зроблені близько 70 років тому. Саме тому крім розпізнавання мови, суть даної роботи полягає в пошуку слів і фраз в відео або аудіо записах. Такий спосіб пошуку може використовуватись звичайним користувачем, задля пошуку фрагменту, в якому звучить фраза або слово, задане ним самим. Наприклад, при перегляді відео-уроків, які на даний момент стають дедалі актуальнішими, користувач може використати навігацію по відео за допомогою пошуку за ключовими словами. Таким чином програмна система може бути використана для пошуку фраз і слів з відео або аудіо, та може сильно спростити пошук відео та навіть необхідної частини відео в відео хостингах, соціальних мережах, а також в глобальних пошукових системах. Так само є можливість використання даної програми великими компаніями, таким як, Google, Яндекс, ВКонтакте, Однокласники. Вона надає можливість індексувати велику кількість відеозаписів за тематикою за допомогою визначення словника термінів кожного відео. Тобто запровадивши даний спосіб пошуку в пошукову систему, знаходження відео вже буде здійснюватися не тільки за назвою, а за тегами, і навіть по тексту, що звучить в відео.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Розпізнавання мовлення

Розпізнавання мовлення – одна з найбільш складних але в той же час і цікавих задач штучного інтелекту. В цьому процесі задіяні досягнення різних областей: від цифрової обробки сигналів до комп’ютерної лінгвістики.

Забезпечення надійного та точного розпізнавання мовлення є надзвичайно важливим завданням в реаліях сучасного світу. Кількість приладів що володіє можливістю хоча б примітивного розпізнавання мовлення неможливо підрахувати та йде на мільярди.

В останній час ми можемо спостерігати гігантський крок вперед в галузі розпізнавання та аналізу мовлення. Якщо ще в дев’яності роки системи в основному були спроможні тільки на перетворення мовлення в текст, то зараз системи вже досить вправно аналізують сказане та можуть виконати задані команди або ж, навіть, надати вам смислову відповідь та поспілкуватись з вами.

З кожним днем дедалі більше об’єктів з якими ми взаємодіємо так чи інакше використовують розпізнавання мовлення. Це може бути звичайна іграшка яка керується голосом або складний робот з безліччю деталей. Автоматична обробка мови вже набула широкого вживання у військовій, космічній техніці, автомобілебудуванні та робототехніці. Надалі автоматична обробка мови буде тільки частіше з’являтись в нашому побуті, так як за таким інтерфейсом взаємодії майбутнє. Також застосування технологій розпізнавання мовлення досить цікаве навіть для одного з розділів мовознавства, а саме прикладної лінгвістики. Так автоматична обробка мови є одним з ключових методів в дослідженнях. Фонетичні знання, поряд з такими дисциплінами, як цифрова обробка сигналів і математичні основи розпізнавання образів створюють велику теоретичну базу комп'ютерних мовних технологій, важливість яких для сучасного інформаційного та комп'ютеризованого суспільства неможливо переоцінити.

На даний момент основними проблеми які з’являються в системах розпізнавання мовлення це:

– спонтанне мовлення, яке наповнене різним мовним сміттям та аграмматизмами;
– наявність в мовленні акустичних перешкод та спотворень в тому числі мінливих;
– мовні перешкоди.

Саме через такі перешкоди створення досконалих систем з розпізнавання мовлення являється досить складною задачею з якою намагаються впоратися багато провідних компаній гігантів в світі. Також хотілося б додати що наступним щаблем в еволюції розпізнавання мови є безмовне розпізнавання мовлення. Його суть полягає в зчитуванні, розумінні та аналізі мовних сигналів ще до їх безпосереднього вимовлення.

Загалом існує багато алгоритмів, моделей за якими можливо виконати автоматичне розпізнавання мовлення та хотілось би зупинитись на декількох з них.

Однією з моделей яка використовуються для розпізнавання мовлення являється прихована марківська модель. Вона являє собою статистична модель, у якій модельована система, береться до уваги як марківський процес із прихованими станами. Модель може бути виражено як найпростішу баєсову динамічну мережу. ПММ відомі в першу чергу завдяки тому що вони використовуються в розпізнаванні часових шаблонів, а саме рукописного введення, розпізнавання мовлення, морфологічної розмітки, жестів, часткових розрядів та навіть в біоінформатиці.

При аналізі людської мови перш за все необхідно визначити до якої частини мови належить кожне з слів у реченні. Для англійській мови це завдання має назву Part-Of-Speech tagging. Тож як саме можливо визначити частину мови члена речення? Розглянемо речення англійською мовою: «The can will rust». В даному реченні, the – артикль або частка «тим», can – може одночасно бути і іменником, і модальним дієсловом; will – модальне дієслово, rust – іменник або

дієслово. У статистичному підході необхідно побудувати таблицю з ймовірностями використання слів у кожному з граматичних значень. Цю задачу можна вирішити на основі тестових текстів, проаналізованих вручну. Відразу після цього можливо виділити одну з проблем, а саме що слово «can» частіше за все використовується в якості дієслова, але іноді він може бути і іменником. Враховуючи цей недолік, була створена модель, яка бере до уваги той факт, що після артикля буде дієслово, прикметник або іменник. Розрахунок ймовірності наведено у формулі 1:

$$\underset{t1..tn}{\arg\max} \prod_{t=1}^{n} p(w_i|t_i)p(t_i|t_{i-1}) \qquad (1)$$

де, t – таг (іменник), w – слово з тексту (table, have ...), p(w|t) –ймовірність відповідності слова тагу t, p(t1|t2) – ймовірність що t1 йде після t2.

Ця статистична модель може бути описана як ергодична прихована марківська модель, наприклад рисунок 1.1

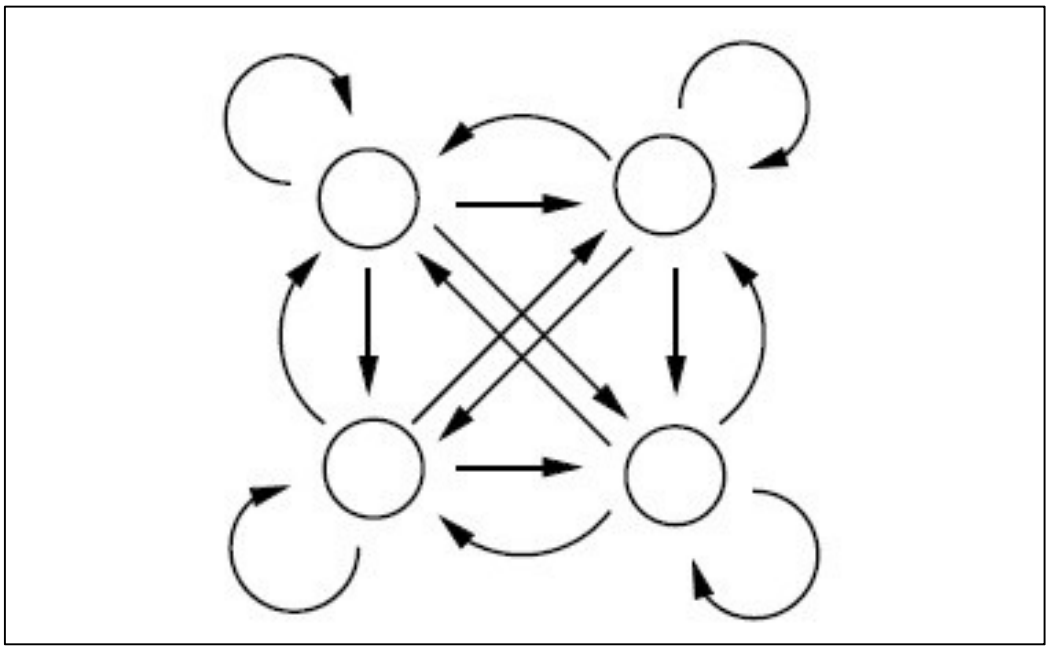


Рисунок 1.1 – Ергодична Марківська модель

За формулою видно, що таги підбираються так, щоб слово відповідало тагу, і таг був влучний до попереднього тагу. Даний метод дозволив нам визначити, що «can» виступає в ролі іменника, а не як модального дієслова. Також можемо поглянути на приклад Ердогичної Марківської моделі на рисунку 1.2

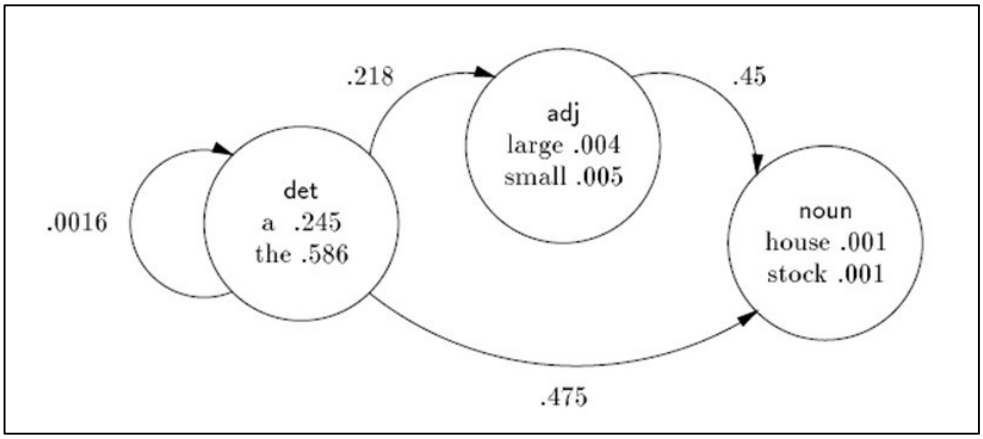


Рисунок 1.2 – Ергодична Марківська модель на практиці

Кожна вершина в даній схемі означає окрему частину мови, в якій записуються пари (слово; ймовірність, що слово належить до певної частини мови). Переходи відображають можливу ймовірність знаходження однієї частини мови за іншою. Так ймовірність того, що 2 артиклі будуть йти один за другим, за умови, що зустрінеться артикль, буде дорівнює 0,0016. Даний етап розпізнавання мови досить важливий, тому що вірне визначення граматичної структури речення дозволяє підібрати правильну граматичну конструкцію для відображення експресивного забарвлення відтворюваного речення.

Також існують n-грамні моделі які також використовуються для розпізнавання мовного потоку. Вони закладені на припущенні, що ймовірність ще одного вживання слова в реченні залежить від n-1 слів. Сьогодні найбільш популярними моделями являються біграмні і триграмні моделі мови. Пошук в таких моделях виконується за великою таблицею (корпусу). Незважаючи на швидкість роботи алгоритму яка досить висока, такі моделі не можуть розпізнати семантичні та синтаксичні зв'язки, якщо залежні слова знаходяться на відстані 5

слів. Використання n-грамних моделей, де n більше за 5, потребує величезних потужностей.

Найпопулярнішою моделлю на сьогоднішній день є триграмна модель. Варто відзначити, далеко діючу триграмну модель, в якій аналіз проводиться не тільки за двома попередніми словами, а по будь-якій парі слів, якщо вони знаходяться поруч. Така модель може пропускати недостатньо інформативні слова, тим самим покращуючи передбачуваність сполучуваності в моделі.

Ще один алгоритм що використовується в розпізнаванні мови є алгоритм Вітербі. Він являється алгоритмом пошуку найбільш належного списку станів, що контексті ланцюгів Маркова отримує найймовірнішу послідовність подій, що відбувалися. Алгоритм Вітербі є представником алгоритмів з динамічного програмування. Його використовують в синтезі мови, біоінформатиці а також комп'ютерній лінгвістиці. При розпізнаванні мови звуковий сигнал буде сприйнятий як послідовність подій і рядок тексту є «прихований сенс» акустичного сигналу. Даний алгоритм знаходить найймовірніший рядок тексту за даними сигналу.

Алгоритм робить декілька припущень:

– події повинні бути послідовністю. Послідовність частіше за все впорядкована за часом;
– дві послідовності має бути вирівняні: кожна зі спостережуваних подій повинна відповідати тільки одній прихованій події;
– обчислення найвірогіднішої прихованої послідовності до моменту t мусить залежати тільки від спостережуваної події в момент часу t, і найвірогіднішої послідовності до моменту t – 1.

Серед основних існуючих систем які зараз присутні на ринку розпізнавання мовлення можливо виділити такі як:

– Google Speech Recognition API – система розроблена компанією Google, яка надає просте в використанні API яке використовує нейронні мережі для перетворення звуків в текст;

– Yandex Speech Kit – система від компанії Яндекс яка працює на основі прихованих марківських моделей;

– Microsoft Speech API – система від компанії Microsoft.

Yandex SpeechKit дозволяє користуватися тим API, який успішно використовують мобільні застосунки Яндекса. При цьому розпізнавання відбувається досить швидко й зазвичай не займає сильно більше секунди. Також процент розпізнавання слів досить високий та наведений на рисунку 1.3:

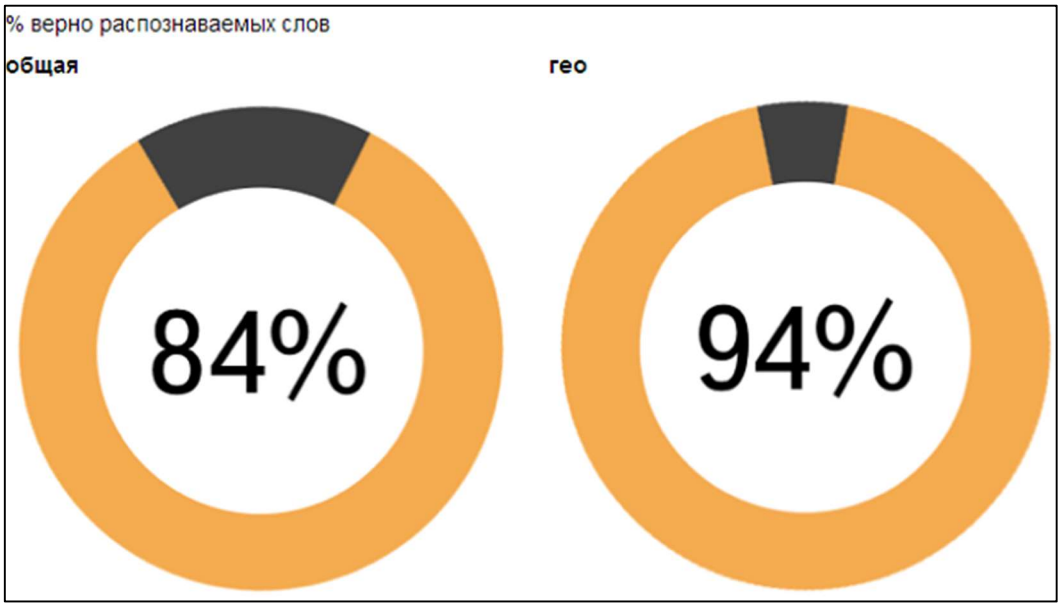


Рисунок 1.3 – Відсоток розпізнаних слів

На мою думку це досить гарний результат та найближчим часом ми зможемо спостерігати що голосові інтерфейси майже не будуть відлічатись за надійністю від класичних методів вводу тексту.

Google Cloud Speech API являється прямим конкурентом для Yandex SpeechKit від компанії Google. Google Cloud Speech API дозволяє розробникам розпізнавати аудіо та отримувати текст, застосовуючи потужні нейронної мережі у простому використанні API. API має змогу розпізнавати більш ніж 80 мов та їх варіації, щоб була змога підтримати глобальну базу користувачів, також можливо фільтрувати небажаний контент в текстових результатах. Ви можете записувати текст просто диктуючи його на мікрофон, ввімкнути голосовий контроль,

транскрибувати аудіофайл та ще безліч інших засобів використання. Розпізнати аудіо в запиті, а також інтеграція Google Cloud Storage зі збереженням аудіо, використовуючи ту ж саму технологію.

Застосування машинного навчання та передових алгоритмів глибокого навчання нейронної мережі дає користувачеві розпізнавання мови з неймовірною точністю. Точність Speech API з кожним днем тільки росте, так само, як Google покращує технологію розпізнавання мовлення, що використовують продукти Google.

Speech API має змогу надсилати текстові результати, повертаючи частини з результатами розпізнавання. Розпізнаний текст видимий відразу під час розмови. В якості альтернативи, Speech API може повертати розпізнаний текст з аудіо, збереженого у файлі. Також вам навіть не знадобляться передові технології з обробки сигналу та шумозаглушення перед відправкою аудіо до Speech API. Служба здатна успішно впоратися з гучним аудіо з любих середовищ.

Розпізнавання мови може бути адаптивним до контексту, надаючи окремий набір слів-термінів що являється особливо корисним, для пристроїв управління.

1.2 Семантичний аналіз

Для того щоб краще розуміти в тому, що таке семантичний аналіз, насамперед потрібно з’ясувати значення самої по собі семантики. Семантика – це дисципліна, котра вивчає взаємозв'язок слів між собою та людською реальністю, визначає значення слова згідно контексту фрази. Семантична модель включає в собі слово, його значення, поєднання з другими словами, створення з нього пропозицій та фраз.

Семантичний аналіз це взагалі складна математична задача, рішення якої застосовується у процесі створення штучного інтелекту, при цьому складність полягає необхідності обробки природної мови. Також складним являється те, що

комп'ютер не вміє вірно пояснювати образи, котрі людина передає за допомогою символів. Дані гарного семантичного аналізу можуть бути використані торговими мережами для аналізу попиту за отриманими відгуками, в різних пошукових системах, системах автоматичного переводу та багато іншого.

Статистичними показниками являються:

– кількість символів без пробілів та з ними;

– кількість слів, в тому числі значущих та унікальних, а також стоп-слів;

– кількість граматичних помилок та води;

– відсоток академічної та класичної нудоти;

– семантичне ядро.

При підрахунку враховується кількість унікальних слів, кількість значущих слів, слів позбавлених жодного сенсу.

Семантичний аналіз тексту оцінює його семантичне ядро, а саме кількість фраз або слів, що визначають основний зміст тексту, а також і основні статистичні показники. Вірно сформоване семантичне ядро спроможне швидко просувати статтю в системі пошуку. Для того щоб створити текст, який буде ефективно впливати на цільову аудиторію, спонукаючи її саме до тих дій, в яких зацікавлені власники досить лиш комбінувати слова та складати грамотно фрази. Пошукові системи також виконують семантичний аналіз, визначаючи сутність тексту, та згодом у відповідь на запит пропонують обрані матеріали.

В рамках даної атестаційної роботи розглядається застосування семантичного аналізу, для спрощення пошуку аудіо та відеозаписів. Це досягається завдяки тому що семантичний аналіз дозволяє визначати теги відео, за якими можливо виконати пошук, та визначити тематику цього відео.

Для того щоб написати алгоритм, який зможе відрізняти новини про народження кошеняти від геополітичних новин (так само як відрізняти тему відео), перше, що спадає на думку, це обрати слова, які можна зустріти виключно у статтях (відео, аудіо) кожного виду і використовувати їх для класифікації. Головною проблемо такого підходу являється неможливість перелічити всі слова, та невизначеність що робити в випадку якщо в статті є слова з різних класів. Ще

одним викликом стають слова омоніми, тобто слова які мають більше одного значення. Наприклад, слово «замок» в одному контексті може означати замок на дверях, а в іншому контексті це може бути королівська споруда.

Латентно-семантичний аналіз показує окремі слова та документи в «семантичний простір», в якому надалі виробляються всі наступні порівняння. Робляться такі припущення:

– документи це не більш ніж набір слів. Порядок слів не має ніякого значення. Важливо тільки те, скільки разів кожне слово зустрічається в даному документ;
– семантичне значення документу визначається комплектом слів, які зазвичай йдуть поряд. Наприклад, у спортивних зведеннях, часто зустрічаються слова: «спортсмен», «змагання», «перемога»;
– кожне слово має максимум одне значення. Це, сильне спрощення, та саме воно дає можливість вирішити задачу.

Якщо брати до уваги українську мову то в даному випадку на допомогу приходить алгоритм Стаммер Портера. Виділення основи слова це одне з завдань, що не рідко виникає в процесі обробки тексту. Наша мова володіє особливою складністю й якщо в англійській мові буде досить зробити декілька звичайних перетворень (наприклад видалення ing-ового закінчення) в результаті чого ми вже з скоріш за все отримуємо необхідну основу, то в українській мові ця задача набагато складніша. В кращому випадку, вона вирішується за допомогою складання великих морфологічних словників, але це не може бути шляхом програміста, оскільки, словники потрібно підтримувати, адже мова постійно змінюється, а також потрібно окремо обробляти граматичні та друкарські помилки. Шлях для програмістів зовсім інший, спеціально для них, у 1980 році Мартіном Портером було винайдено алгоритм стеммінгу.

Головний плюс Стеммера Портера полягає в тому, що використання словників не потрібне, а виділення основи здійснюється шляхом безпосереднього перетворення слова згідно з окресленими правилами.

2 ДОСЛІДЖЕННЯ

2.1 Дослідження методів розпізнавання мовлення

Звучна мова для нас – це, насамперед цифровий сигнал. Тож якщо поглянути на запис сигналу, то не помітимо там ні слів, ні чітко виражених фонем – різні мовні події поступово перетікають з одного в інше, не утворюючи чітких меж та границь. Одна і та ж сама фраза, що вимовлена декількома людьми або в за різних обставин, на рівні сигналу буде виглядати зовсім по-різному. Але все ж таки, люди вміють розпізнавати мову один одного, тож, існують сценарії згідно з якими за сигналом можливо відрізнити, що ж саме малося на увазі. Пошук таких сценаріїв й є завданням акустичного моделювання.

Припустимо, що мовлення кожного з нас складається з фонем (це грубе спрощення, але в першому наближенні воно являється вірним). Визначимо фонему як найменшу змісторозрізнювальну одиницю мовлення, а саме звук, зміна якого можливо призведе до зміни сенсу слова або фрази. Візьмемо невеликий відрізок сигналу, наприклад, 30 мілісекунд. Для нас це відрізок буде «кадром». Яка саме фонема була сказана протягом цього кадру? Це досить складне питання яке не має однозначної відповіді – велика кількість фонем дуже схожі одна на одну. Якщо неможливо дати точну відповідь, то можемо взяти ймовірність. Так для даного сигналу одні фонеми більш ймовірні, інші менш, а деякі взагалі неможливі й їх можна не брати до уваги. Акустична модель – це насамперед функція, що отримує на вхід невеликий відрізок акустичного сигналу, або його називають фрейм та видає розподіл ймовірностей різних фонем на даному фреймі. Отже, акустична модель надає нам можливість відновити що було сказано за звуком з тою чи іншою впевненістю.

Ймовірність переходу між різними фонемами являється ще одним досить важливим моментом акустики. Як ми знаємо з нашого досвіду, деякі поєднання фонем досить легко вимовляються і не рідко зустрічаються, інші являються вже складнішими для вимови та на практиці використовуються не так часто. Таким

чином можливо узагальнити інформацію та врахувати її при оцінці ймовірності тієї чи іншої послідовності фонем.

На цьому етапі ми можемо сконструювати один з основних компонентів автоматичного розпізнавання мовлення – приховану марківску модель яку ми розглядали напередодні. Для цього нам потрібно уявити, що вирішується протилежна задача розпізнавання мови, а саме перетворення тексту в мову. Наприклад, треба отримати вимову слова «Університет». Нехай слово «Університет» складається з набору фонем, скажімо, [у][н][і][в][е][р][с][и][т][е][т]. Побудуємо кінцевий автомат для слова «Університет», в якому кожна з фонем представлена окремим станом. В кожний момент часу перебуваємо в одному з цих станів та вимовляємо характерний для цієї фонеми звук (як саме вимовляється кожна з фонем ми можемо знати завдяки акустичній моделі). Але деякі фонеми тривають довго, інші практично незамітні. Саме тут нам й знадобиться інформація про ймовірності переходів між фонемами. Згенерувавши звук, який відповідає поточному стану, приймаємо ймовірнісне рішення: залишатися в цьому ж самому стані чи переходити до наступного. Наведемо автомат слова «Університет» на рисунку 2.1.

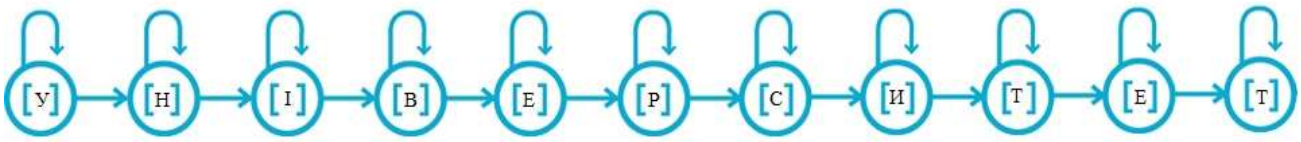


Рисунок 2.1 – Автомат

Більш формально приховану марківску модель можна представити наступним чином. Насамперед, введемо поняття емісії. Кожне з станів моделі породжує звук, який характерний саме для цього стану тобто фонеми. На кожному кадрі звук розігрується з розподілу ймовірностей, відповідного даній фонемі. Далі враховуємо що між станами можливі переходи, що також слідують заздалегідь визначеним ймовірнісним закономірностям. Наприклад, ймовірність того, що фонема [е] буде тягнутися досить висока, чого неможливо сказати про

фонему [н]. Матриця переходів та матриця емісій абсолютно точно задають приховану марківську модель.

Було розглянуто, як марківська модель може бути використана для породження мовлення. Тепер за допомогою алгоритму Вітербі ми можемо застосувати її до зворотної задачі, а саме розпізнавання мови. Ми маємо комплект спостережуваних величин а саме звук та ймовірнісна модель, співвідношення приховані стану і спостережувані величини. За допомогою Алгоритму Вітербі ми можемо відновити найймовірнішу послідовність прихованих фонем.

Нехай наш словнику розпізнавання має всього два слова: «Так» ([т][а] [к]) й «Ні» ([н][і]). Отже, ми маємо дві приховані марківські моделі. Далі, нехай, ми маємо запис голосу людини, що каже «так» або «ні». За допомогою Алгоритму Вітербі ми зможемо отримати відповідь на питання, яка саме з гіпотез розпізнавання більш ймовірна.

Зараз нам потрібно відновити найбільш ймовірну послідовність станів(фонем) прихованої марківської моделі, яка могла б породити або породила пред'явлений нам аудіозапис. Якщо людина говорить «так», то відповідна послідовність станів на 10 кадрах може бути, наприклад, [т][т][т][а][а][а][а][к][к][к] або [т][т][а][а][а][а][к][к][к][к]. Аналогічно, можливі різні варіанти вимови для «ні» – наприклад, [н][н][н][н][і][і][і][і][і][і] або [н][н][н][і][і][і][і][і][і][і]. Тепер знайдемо найбільш ймовірний, спосіб вимовляння кожного зі слів. На кожному кадрі будемо запитувати нашу акустичну модель, наскільки велика ймовірність того, що тут звучить конкретна фонема (наприклад, [н] і [і]), крім того, будемо враховувати ймовірності переходів ([н]->[н], [н]->[і], [і]->[і]). Таким чином ми отримаємо найймовірніший спосіб виголошення кожного зі слів-гіпотез. Більше того, для кожного ми отримаємо міру, наскільки ймовірно, що було сказане саме це слово (міру можливо ототожнювати з довжиною найкоротшого шляху через відповідний граф). Найбільш ймовірна гіпотеза буде повернута в результаті розпізнавання.

В любому випадку Алгоритм Вітербі є досить простим в реалізації алгоритмом, для його реалізації використовується динамічне програмування й час

його роботи є пропорційним кількості фреймів множених на кількість станів прихованої марківської моделі. Та все ж таки не завжди нам достатньо знати найбільш ймовірніший шлях, наприклад для, тренування акустичної моделі використовується алгоритм Forward-Backward так як для моделі потрібні оцінки ймовірності всіх станів протягом всіх кадрів.

Акустична модель це не більше ніж одні зі складових великої системи. Що ж потрібно робити, якщо наш словник розпізнавання, який перед цим складався всього з двох слів, має тепер набагато більше елементів, наприклад десятки, сотні, тисячі або навіть мільйони? Більшість з цих слів будуть досить схожими на інші за вимовою, а можуть бути взагалі однаковими. В цей момент на допомогу приходить контекст. Роль акустики сильно падає якщо ми маємо контекст: проковтнуті слова, зашумлені сцени, або неоднозначну вимову можливо відновити за контекстом всього речення, за змістом. Й знову на допомогу приходять ймовірнісні моделі. Вони використовуються для урахування контексту знову. Кожному носію української мови стає очевидним що ймовірність сказаного «мама мила раму» набагато вище, ніж «мама мила синхрофазотрон» або «мама мила автомобіль». Тобто наявність контексту «мама мила ...» показує розподіл ймовірностей для наступного слова, що звісно відображає як морфологію так само і семантику. Такі мовні моделі носять назву н-грамних мовних моделей або n-gram language models. В любому випадку зрозуміло, що існують куди більш складні та потужні способи моделювання мови. [8]

При розгляді прикладів вище я навмисно зробив кілька спрощень і пропустив ряд важливих речей. Насамперед я стверджував, що головною будівельної одиницею мовлення являється фонема. Та насправді фонема є дуже великою одиницею. Для моделювання вимови одиночної фонеми, зазвичай використовується три окремих стани, а саме початок, середина і кінець. Фонеми є позиційно та контекстно-залежними. Формально одна й та ж сама фонема сильно відрізняється за звуком в залежності від того в якій частині слова вона розташована та з якими фонемами являється сусідами. Разом з тим, досить велика кількість комбінацій буде отримано в випадку простого перелічення всіх

можливих варіантів контекстно-залежних фонем, а більша частина з яких навіть неможливо зустріти в реальному житті. Щоб ця кількість була в розумних межах, зазвичай контекстно-залежні фонеми які близькі поєднуються ще на перших етапах тренування, та потім розглядаються вцілому.

Виконавши такі дії, ми, перш за все, робимо фонеми контекстно-залежними, а також, розділили кожну з них на три частини. Фонетичний алфавіт складається тепер саме з цих об'єктів . Також вони мають назву сенон. Кожний стан прихованої марківської моделі – це сенон. Модель використовує 48 фонем та приблизно 4000 сенонів.

Таким чином, акустична модель також приймає звук на вхід, а на вихід віддає розподіл ймовірностей за сенонами. Тепер можемо розглянути, що конкретно подається на вхід. Як вже було сказано звук нарізається ділянками по 25 мс на фрейми. Зазвичай, крок нарізки складає 10 мс, таким чином сусідні кадри дещо перетинаються своїми частинами. Звісно ж сирий звук що є амплітудою коливань за часом не є найбільш інформативною формою подання акустичного сигналу. Набагато краще мати спектр цього сигналу. Частіше за все логарифмований та відмасштабований спектр використовується на практиці, це такі спектри що відповідають закономірностям людського слухового сприйняття – Mel-перетворення. Далі отримані величини піддаються косинусному дискретному перетворенню (DCT), і в результаті чого отримуємо MFCC – Mel Frequency Cepstral Coefficients. (Слово Cepstral було отримано за допомогою перестановки літер у Spectral, таким чином це відображає наявність додаткового DCT). MFCC – це вектор який складаються зазвичай з 13 дійсних чисел. Числа можуть біти використані як вхід акустичної моделі в сирому вигляді», проте частіше всього піддаються великій кількості додаткових перетворень.

Тренування акустичної моделі являється досить складним та багатоетапним процесом. Алгоритми сімейства Expectation-Maximization використовуються для тренування, а саме такі, як алгоритм Баума-Велша. Суть алгоритмів такого роду в чергуванні двох кроків:

– крок Expectation – наявна модель використовується для підрахування маточікування функції правдоподібності;
– крок Maximization – параметри моделі змінюються для того, щоб максимізувати оцінку.

На ранніх етапах тренування використовуються звичайні акустичні моделі. На вхід даються звичайні й прості MFCC features, фонеми розглядаються поза контекстної залежності, для моделювання ймовірності емісії в HMM використовується суміш гауссіан з діагональними матрицями коваріацій (Diagonal GMMs – Gaussian Mixture Models). Стартовою точкою для тренування більш складної моделі, з більш складним входом, виходом або функцією розподілу ймовірності емісії являються результати кожної попередньої акустичної моделі. Існує дуже багато способів покращення акустичної моделі, однак найкращий ефект має перехід від GMM-моделі до DNN (Deep Neural Network), він підвищує якість розпізнавання практично в два рази. Нейронні мережі позбавлені багатьох обмежень, характерних для гауссових сумішей, мають кращу узагальнюючу здатність. Також, акустичні моделі які побудовані на нейронних мережах більш стійкі до шумів та володіють кращим швидкодією.

Для акустичного моделювання нейронна мережа тренується в декілька етапів. Для ініціалізації нейромережі використовують стек з обмежених машин Больцмана (Restricted Boltzmann Machines, RBM). RBM - це стохастична нейромережа, що тренується без участі вчителя. Хоча вивчені їй ваги не можна безпосередньо використовувати для розрізнення між класами акустичних подій, вони досить детально показують структуру мови. Можна ставитися до стохастичної нейромережа як до механізму вилучення ознак – отримана генеративна модель виходить прекрасною стартовою точкою для побудови дискриминативной моделі. Ряд технічних прийомів, поліпшуючих збіжність і запобігають перенавчання (overfitting) застосовується при тренуванні дискримінативна моделі з використанням класичного алгоритму зворотного поширення помилки. У підсумку на вході нейромережі отримуємо декілька фреймів MFCC-features (тільки центральний кадр підлягає класифікації, а інші в

свою чергу утворюють контекст), на виході отримуємо близько 4000 нейронів, що відповідають різним сенонам. Ця нейромережа використовується як акустична модель у production-системі. Як відбувається процес декодування відображено на рисунку 2.2.

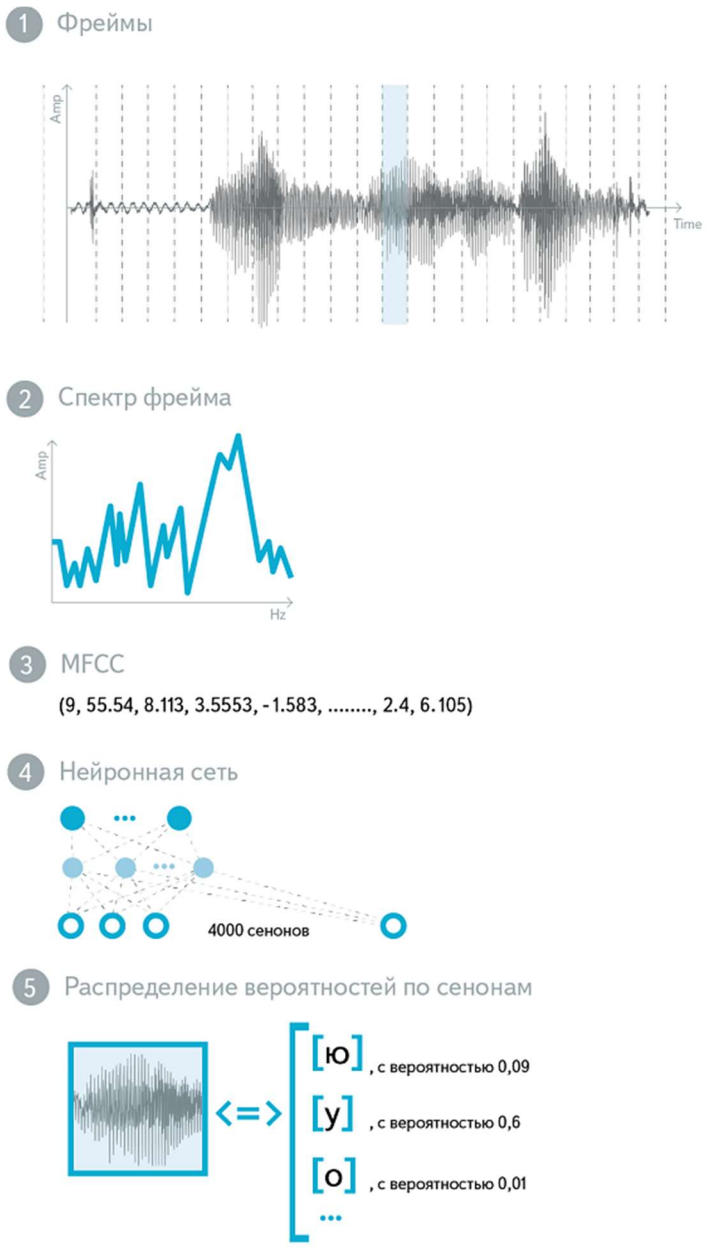


Рисунок 2.2 – Процес декодування

Також хотілося б розглянути детальніше процес декодування. Підхід, що був описаний напередодні, непридатний для задачі розпізнавання спонтанної мови з великим словником. Для цього необхідна структура даних, яка з'єднує в

одне ціле всі можливі пропозиції, які здатна розпізнати система. Саме такою структурою являється weighted finite state transducer (WFST) – насправді, це просто кінцевий автомат з вихідною стрічкою і вагами на ребрах. На вході цього автомата розташовані сенони, на виході ж слова. Процес декодування зводиться до вибору найкращого шляху в цьому автоматі та наданні вихідної послідовності слів, що відповідають цьому шляху. При цьому ціна проходу по кожній з дуг складається з двох компонент. Перша компонента відома відразу та вичисляється на етапі складання автомата. Вона має в собі вартість вимови, переходу в цей стан, оцінку правдоподібності з боку мовної моделі. Друга компонента вичисляється окремо для конкретного кадру – це акустична вага сенона, відповідного вхідного символу дуги що була розглянута. Декодування відбувається в режимі реального часу, тож досліджуються не всі можливі шляхи, а спеціальні евристики обмежують набір гіпотез найбільш ймовірними.

Звісно ж, найбільш цікавою з технічної точки зору частина являється побудова такого автомата. Ця задача вирішується в офлайні. Щоб перейти від простих марківських моделей для кожної контекстно-залежної фонеми до лінійних автоматів для кожного слова, необхідно використати словник вимов. Створення такого словника неможливо виконати вручну, і тут на допомогу приходять методи машинного навчання, а сама задача в науковому співтоваристві називається Grapheme-To-Phoneme, або G2P. В свою чергу, слова стикуються з іншими в мовну модель, яка також представлена у вигляді кінцевого автомата. Основною операцією тут є композиція WFST, але також важливі і різноманітні методи оптимізації WFST за розміром та ефективністю укладання в пам'яті.

Результатом процесу декодування являється список гіпотез, котрий може бути підданий послідуючій обробці. Наприклад, можливо використовувати більш потужну мовну модель для переанжировування найбільш вірогідних гіпотез. Результуючий список повертається користувачеві, відсортований за значенням confidence – ступенем нашої впевненості в тому, що розпізнавання пройшло вірно й успішно. Нерідко залишається всього лише одна гіпотеза, в цьому випадку додаток-клієнт в той же час переходить до виконання голосової команди.

На рисунку 2.3 зображено автомат списку гіпотез:

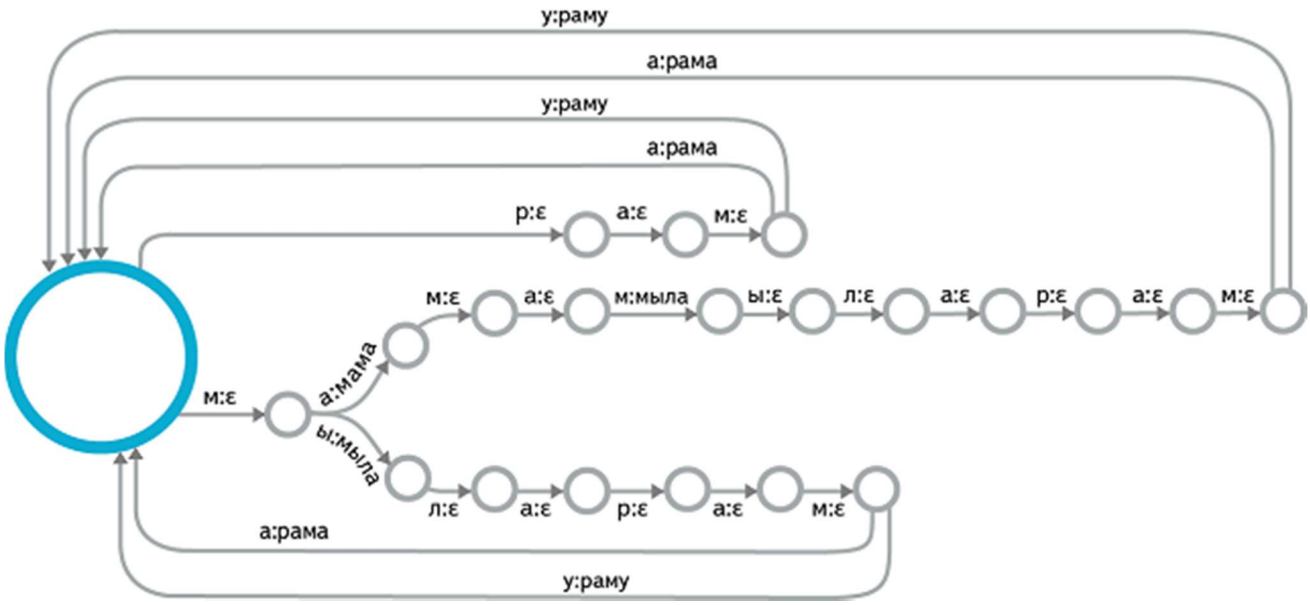


Рисунок 2.3 – Автомат списку гіпотез

Також хотілося б торкнутися питання про метрики якості систем розпізнавання мовлення. Найбільш популярна метрикою являється Word Error Rate (і зворотна їй Word Accuracy). По суті, вона показує частку невірно розпізнаних слів. Щоб розрахувати Word Error Rate для системи розпізнавання мови, використовують розмічені вручну корпусу голосових запитів, відповідних тематиці програми, що використовує розпізнавання мови.

2.2 Дослідження методів семантичного аналізу

Приклад (на основі заголовків статей):

Для прикладу було обрано декілька заголовків з різних статей новин. Вони були обрані не в випадковому порядку, річ в тому, що якщо б я взяв випадковий набір новин, то отримав би досить великий обсяг даних, що в сильно

ускладнило б подальший виклад. Таким чином, було вирішено обрати кілька заголовків.

Перш за все з цих заголовків були виключені, стоп-символи (вони перелічені на рисунку 2.4). Це слова, які ми можемо зустріти в кожному тексті, в повсякденному мовленні і які не несуть в собі абсолютно ніякого смислового навантаження, перш за все, це всі сполучники, частки, прийменники, а загалом допоміжні частини мовлення.

-	еще	него	сказать
а	ж	нее	со
без	же	ней	совсем
более	жизнь	нельзя	так
больше	за	нет	такой
будет	зачем	ни	там
будто	здесь	нибудь	тебя
бы	и	никогда	тем
был	из	ним	теперь
была	из-за	них	то
были	или	ничего	тогда
было	им	но	того
быть	иногда	ну	тоже
в	их	о	только
вам	к	об	том
вас	кажется	один	тот
вдруг	как	он	три
ведь	какая	она	тут
во	какой	они	ты
вот	когда	опять	у
впрочем	конечно	от	уж
все	которого	перед	уже
всегда	которые	по	хорошо
всего	кто	под	хоть
всех	куда	после	чего
всю	ли	потом	человек
вы	лучше	потому	чем
г	между	почти	через
где	меня	при	что
говорил	мне	про	чтоб
да	много	раз	чтобы
даже	может	разве	чуть
два	можно	с	эти
для	мой	сам	этого
до	моя	свое	этой
другой	мы	свою	этом
его	на	себе	этот
ее	над	себя	эту
ей	надо	сегодня	я
ему	наконец	сейчас	
если	нас	сказал	
есть	не	сказала	

Рисунок 2.4 – Таблиця стоп-слів

Крім зазначених слів має сенс також фільтрувати цифри, окремі букви та розділові знаки.

Наступним етапом була виконана операція стеммінгу. Вона являється не обов'язковою, за деякими джерелами відомо що, що гарні результати можливо отримати і без неї. Насправді так і є, якщо ваш набір текстів досить великий, то цей крок можна з легкістю не брати до уваги та пропустити. Також цей шаг можна пропускати якщо текст написаний англійською мовою, це дозволено через те, що в англійській мові ймовірність великої кількості варіацій слова досить низка, в порівнянні з російською або українською. У нашому ж випадку, ми не можемо дозволити собі пропускати цей крок, оскільки це призведе до істотного спотворення результатів. Для стеммінгу був обраний алгоритм Портера.

Наступним етапом були виключені слова які зустрічаються в єдиному екземплярі. Цей крок також являється необов’язковим, він не впливає на кінцевий результат, проте він дозволяє сильно спростити подальші математичні обчислення. В результаті у нас залишаються тільки індексовані слова, які виділені на рисунку 2.5 [14]

Британская **полиция** знает о местонахождении **основателя WikiLeaks**
В **суде США** начинается процесс **против** россиянина, рассылавшего спам
Церемонию вручения Нобелевской премии мира бойкотируют 19 **стран**
В **Великобритании арестован основатель** сайта **Wikileaks** Джулиан Ассандж
Украина игнорирует **церемонию вручения Нобелевской премии**
Шведский **суд** отказался рассматривать апелляцию **основателя Wikileaks**
НАТО и **США** разработали планы обороны **стран** Балтии **против** России
Полиция Великобритании нашла **основателя WikiLeaks**, но, не **арестовала**
В Стокгольме и Осло сегодня состоится **вручение Нобелевских премий**

Рисунок 2.5 – Виділені слова після індексації

Далі наступає етап латентно семантичного аналізу. На першому його кроці потрібно скласти частотну матрицю проіндексованих слів (дивитися рисунок 2.6). У даній матриці рядки відповідають проіндексованим словами, а стовпці відповідають документам. На перехрестях рядків та стовбців ми можемо спостерігати кількість зустрічань окремого слова в певному документі.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
wikileaks	1	0	0	1	0	1	0	1	0
арестова	0	0	0	1	0	0	0	1	0
великобритан	0	0	0	1	0	0	0	1	0
вручен	0	0	1	0	1	0	0	0	1
нобелевск	0	0	1	0	1	0	0	0	1
основател	1	0	0	1	0	1	0	1	0
полиц	1	0	0	0	0	0	0	1	0
прем	0	0	1	0	1	0	0	0	1
прот	0	1	0	0	0	0	1	0	0
стран	0	0	1	0	0	0	1	0	0
суд	0	1	0	0	0	1	0	0	0
сша	0	1	0	0	0	0	1	0	0
церемон	0	0	1	0	1	0	0	0	0

Рисунок 2.6 – Частотна матриця індексованих слів

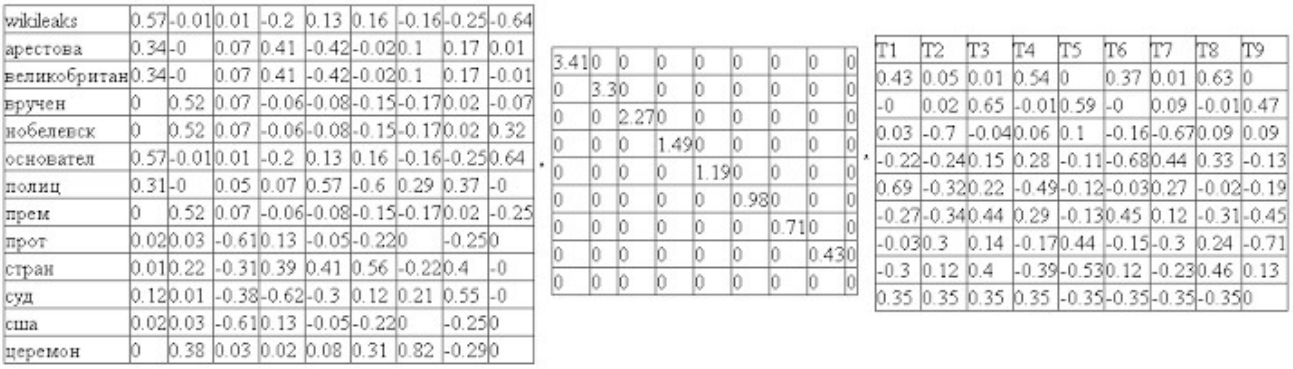
Вихідну матрицю M ми представляємо у вигляді формули 2:

$$M = U * W * V^t, \quad (2)$$

де, U і V^t – ортогональні матриці, W – діагональна матриця.

Це є наступним етапом що ми проводимо, а саме сингулярне розкладання отриманої матриці. Дане розкладання являється нічим іншим як математичною операцією що розкладає матрицю на три складових.

В порядку зменшення упорядковані діагональні елементи матриці W. Вони називаються сингулярними числами (рисунок 2.7).



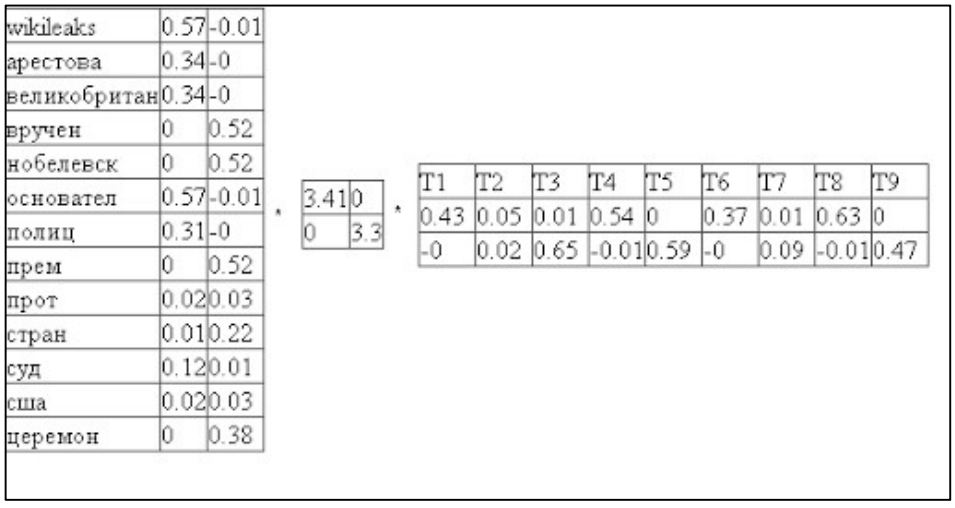
wikileaks	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	-0.64
арестова	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	0.01
великобритан	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	-0.01
вручен	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.07
нобелевск	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	0.32
основател	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	0.64
полиц	0.31	-0	0.05	0.07	0.57	-0.6	0.29	0.37	-0
прем	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.25
прот	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
стран	0.01	0.22	-0.31	0.39	0.41	0.56	-0.22	0.4	-0
суд	0.12	0.01	-0.38	-0.62	-0.3	0.12	0.21	0.55	-0
сша	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
церемон	0	0.38	0.03	0.02	0.08	0.31	0.82	-0.29	0

3.41	0	0	0	0	0	0	0	0
0	3.3	0	0	0	0	0	0	0
0	0	2.27	0	0	0	0	0	0
0	0	0	1.49	0	0	0	0	0
0	0	0	0	1.19	0	0	0	0
0	0	0	0	0	0.98	0	0	0
0	0	0	0	0	0	0.71	0	0
0	0	0	0	0	0	0	0.43	0
0	0	0	0	0	0	0	0	0

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47
0.03	-0.7	-0.04	0.06	0.1	-0.16	-0.67	0.09	0.09
-0.22	-0.24	0.15	0.28	-0.11	-0.68	0.44	0.33	-0.13
0.69	-0.32	0.22	-0.49	-0.12	-0.03	0.27	-0.02	-0.19
-0.27	-0.34	0.44	0.29	-0.13	0.45	0.12	-0.31	-0.45
-0.03	0.3	0.14	-0.17	0.44	-0.15	-0.3	0.24	-0.71
-0.3	0.12	0.4	-0.39	-0.53	0.12	-0.23	0.46	0.13
0.35	0.35	0.35	0.35	-0.35	-0.35	-0.35	-0.35	0

Рисунок 2.7. – Матриці після сингулярного розкладу

Сутність сингулярного розкладання в тому що воно виділяє ключові частини матриці, дозволяючи проігнорувати непотрібні шуми. Згідно простим правилам множення матриць, ми можемо бачити, що стовпці та рядки що відповідають меншим сингулярним значенням дають в результаті найменший внесок у підсумковий результат. Наприклад, ми можемо взяти й відкинути останні стовпці матриці U, а також останні рядки матриці V^t, залишивши тільки перші 2. Важливим моментом, є те що при цьому гарантується, оптимальність отриманого множення. Таке розкладання носить назву двовимірного сингулярного розкладання, та зображене на рисунку 2.8:



wikileaks	0.57	-0.01
арестова	0.34	-0
великобритан	0.34	-0
вручен	0	0.52
нобелевск	0	0.52
основател	0.57	-0.01
полиц	0.31	-0
прем	0	0.52
прот	0.02	0.03
стран	0.01	0.22
суд	0.12	0.01
сша	0.02	0.03
церемон	0	0.38

3.41	0
0	3.3

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47

Рисунок 2.8 – Матриця двовимірного сингулярного розкладання

Зазначивши на графіку точки відповідні текстам і словами, вийде (рисунок 2.9):

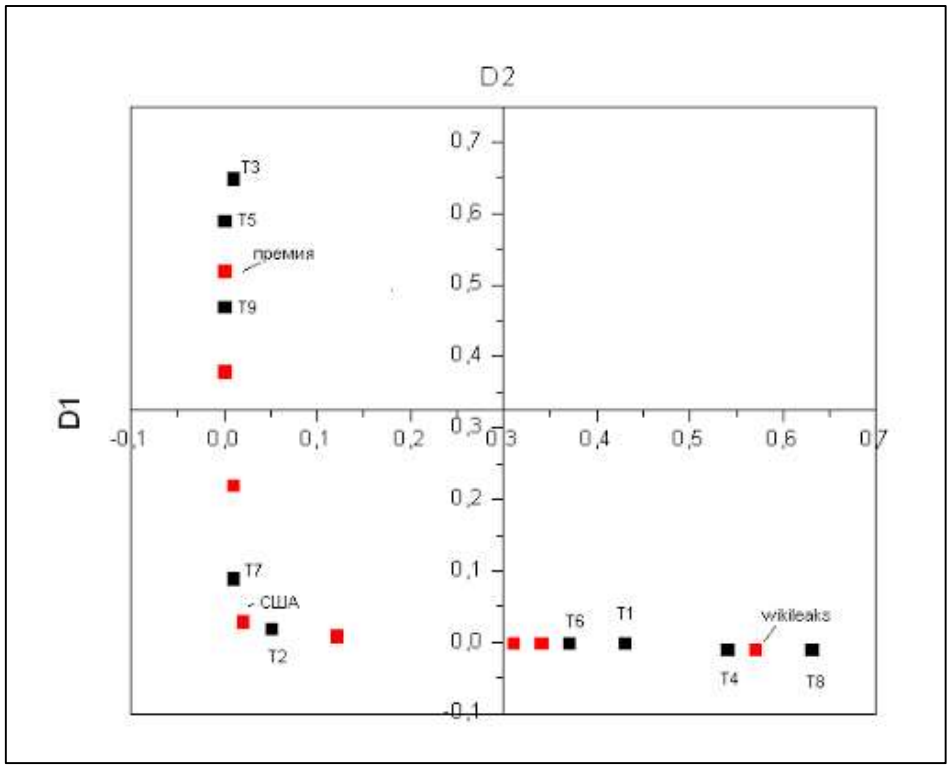


Рисунок 2.9 – Графік результатів

З даного графіка ми можемо побачити, що статті утворюють три незалежні групи, перша група статей розташована поруч зі словом «wikileaks», та й насправді, якщо поглянути на назви цих статей ми відразу можемо зрозуміти, що вони мають відношення до wikileaks. Інша група статей була утворена навколо слова «премія».

Кількість груп буде набагато більше на практиці, так як ми обрали досить схожу заголовки статей, й простір буде не двовимірним а багатовимірним, але в цілому весь процес та ідея залишається таким же як є. Таким чином ми можемо визначати розташування слів і статей в нашому просторі і використовувати цю інформацію для, наприклад, визначення тематики статті або відеозапису.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Перший етап в розробці програмного забезпечення став вибір мови програмування для подальшої роботи. Після проведеного аналізу було вирішено обрати мову C# платформи .NET, з ряду причин. Так наприклад C# підтримує просту інтеграцію зовнішнього плеєра та має досить просту реалізацію інтеграції з Google API. Так само за допомогою стандартних засобів, досить просто розробити інтерфейс майбутнього програмного забезпечення.

3.1 Інтеграція стандартного плеєра в проект

Перш за все потрібно інтегрувати медіа програвач Windows Media Player в наше ПЗ. Плеєр буде використаний для програвання відео та аудіофайлів. Було обрано саме WMP тому що насамперед це безкоштовний програвач який повністю задовольняє наші потреби. Також WMP виробляється корпорацією Майкрософт саме тому він вбудований у всіх операційних системах Windows. Звісно ж Майкрософт робить безкоштовні версії даного плеєру для інших досить популярних операційних систем, наприклад для Mac OS, та вони мають меншу функціональність, оновлення надходять досить рідко, також підтримують не всі популярні формати медіа файлів.

Для того що інтегрувати WMP було використано стандартний MediaPlayer Class платформи .NET Framework. Простором імен обраної бібліотеки являється System.Windows.Media. Ієрархією спадкування можемо поглянути на рисунку 3.1

MediaPlayer() – ініціалізує новий екземпляр класу mediaplayer.

Приклад інтегрованого плеєра можна подивитися в Додатку А та його реалізацію в Додатку Б.

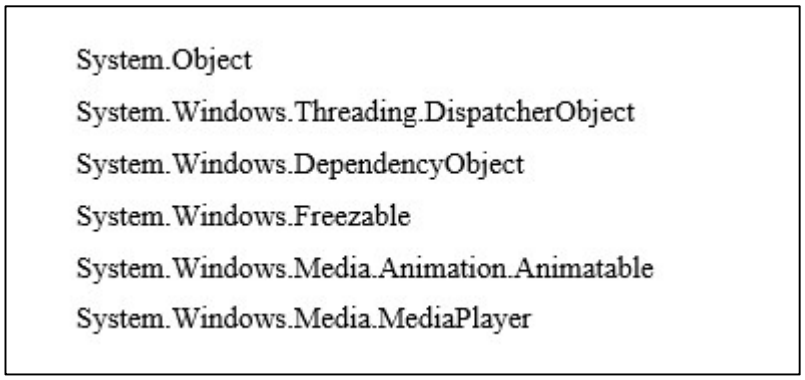


Рисунок 3.1 – Ієрархія спадкування MediaPlayer

3.2 Формати

Наступним етапом реалізації є вибір форматів відео файлів, для подальшої їхньої конвертації в аудіо. Зіткнувшись з даним питанням було вирішено приділити особливу увагу цій темі. Переглянувши всі можливі варіанти, було зроблено висновок, що простіше за все буде працювати з функції для конвертації відео формату avi в wav та вже потім конвертувати його в аудіоформат flac, таке рішення було зроблено через обмеження Google API яке здатне приймати тільки аудіо даного формату. Розробка та написання функцій почалася з вивчення відмінностей між різними відео та аудіо потоками, частотою дискретизації та бітрейту.

Формат відео файлу – це не що інше як структура, в якій відеозапис зберігається на носіях, наприклад це може бути пам'ять комп'ютера, флеш пам’ять, оптичний диск та інші. Формат можна визначити двома речами:

– файл-контейнер – безпосередньо в ньому зберігається запис;

– кодек – використовується для подальшої обробки звуку та зображення.

В процесі запису відео на цифровий пристрій (смартфон, фотоапарат, відеокамера або будь-який інший гаджет) звук та відео записуються в окремі потоки які називаються доріжками, в той самий час кодуючи кожен з них спеціальною програмою – кодеком. Наступним кроком обидва потоки

поміщаються в єдиний файл що і є тим самим файлом-контейнером, де вони забезпечуються описом. Дана інформація в подальшому допоможе медіапрогравачеві (в нашому випадку WMP) вірно синхронізувати звук з картинкою при подальшому відтворенні відеозапису.

Сам по собі кодек являється нічим іншим як програмою, яка стискає в момент запису та розпаковує назад в момент відтворення мультимедійної інформацію: відео, аудіо, анімацію та інші. Кожен вид інформації має власні кодеки, тобто обробляють звук одні кодеки, а ось кодують картинку вже зовсім інші.

Досить популярний контейнер AVI дозволяє помістити в собі декілька аудіодоріжок та субтитрів. Інший відомий контейнер MKV (Matroska) – здатний упакувати все, що заманеться, немає ніяких обмежень по кількості відео та аудіодоріжок, анімовані та тестові субтитри, всілякі шрифти для них та багато чого іншого.

Та все ж не всі кодеки можуть співпрацювати з любими форматами як аудіо так і відео. Так, контейнер MPEG маж підтримку тільки таких відео, що були стиснуті за допомого MPEG-1 чи MPEG-2. А от інший кодек – контейнер MP4 вже підтримує більшу кількість кодеків.

Існують формати які здатні видати ідеальне зображення і та чистий звук, тоді як інші здатні тільки на те щоб відобразити ледь прийнятну якість відео. Та все ж таки неможливо дати відповідь на питання: "Який з форматів відео краще?". Кожен формат маж своє призначення і в якомусь випадку він буде краще за інші, та в деяких він буде досить поганим варіантом. Також це сильно залежно від особливостей пристрою, на якому буде відтворюватись фільм або аудіозапис, кращим може виявитися той кодек який насправді самий простий або, найпоширеніший формат.

Як вже було сказано було прийняте рішення використовувати формат відео файлів AVI або Audio Video Interleave, з назви стає зрозуміло що сутність контейнеру полягає в чергуванні відео та аудіо. AVI був заснований ще в 1992

року компанією Microsoft для надання можливості операційній системі Windows обслуговування мультимедіа як частина технології Video for Windows.

Форматами файлів які відповідають за збереження звукових даних на комп'ютерах та інших цифрових пристроях називаються цифровими звуковими форматами. Також їх можуть називати звуковими файлами або аудіофайлами.

Взагалі основний принцип збереження аудіофайлів на комп'ютерах полягає в почерговій фіксації звукових коливань амплітуди. Вони відповідають положенню мембран в гучномовцях при відтворення аудіофайлу. Саме ці дані кодуються з певною частотою дискретизації та певним амплітудним розділенням. Також для полегшення передачі аудіофайлів та зменшення їх обсягів, вони можуть бути стиснуті з втратами або без втрат якості звучання.

Взагалі на даний час існує три основні групи аудіофайлів:

– формати які не були стиснуті, наприклад WAV, AIFF, AU або PCM, вони зазвичай мають найкращу якість звуку;
– формати які застосовують стиснення не втрачаючи при цьому якість – одним з них являється формат FLAC який використовує Google API Також є інші приклади: ATRAC Advanced Lossless, MPEG-4 ALS, WMA Lossless, Shorten, Apple Lossless, MPEG-4 DST;
– формати які застосовують стиснення та несуть зазнають втрат якості звуку. Найпопулярнішим з цих форматів являється MP3, а також lossy Windows Media Audio або WMA.

Варто враховувати, що такий формат як MIDI не відноситься до аудіофайлів. Також файли нотних редакторів, так як вони являють собою не більше ніж послідовність команд для музичного інструменту, й не мають жодної інформації безпосередньо про звук.

Формат аудіофайлу також слід відрізняти від аудіокодеку. Кодек здійснює кодування чи розкодування звукових даних, тоді як самі дані зберігаються у файлі відповідного звукового формату. Більшість форматів підтримують лише один тип кодування звукових даних, проте мультимедійні контейнери (напр. MKV або AVI) можуть підтримувати різні типи аудіо і відео даних.

Одним з найпоширеніших аудіофайлів які не зазнають стиснення являється WAV або waveform audio format. Цей формат був розроблений спільними зусиллями таки компаній як Microsoft та IBM. Основою для WAVE слугує форматі RIFF, поширюючи його на інформацію про те який кодек був застосований, якою була частота дискретизації та кількість каналів. Перш за все WAV так само як і RIFF був розроблений для комп'ютерів IBM Personal Computer, саме тому всі змінні формату записані в little endian. Для комп'ютерів PowerPC аналогом WAV являється AIFF.

Зазвичай для запису WAVE використовують нестиснений кодек PCM, хоча можна було б використати будь-який аудіокодек. Кодек PCM призводить до досить великих обсягів файлу як для аудіо файлі - близько 172 кБ на секунду. Обмеження обсягу до 4гб також є важливим недоліком цього формату. Це викликане використанням 32-бітної змінної. В наш час формат WAV був дещо зміщений з ринку стисненими форматами – наприклад MP3. Проте, завдячуючи своїй простоті, WAV і зараз активно використовується в таких процесах як редагування звуку а також на програвачах та цифрових диктофонах.

Після ретельного аналізу аудіо форматів а також обмежень зі сторони Google speech API було вирішено використовувати FLAC аудіо формат з частотою дискретизації 16000 Гц, бітрейтом 256 kbps та одноканальним кодуванням.

Після детального обзору формату FLAC або Free Lossless Audio Codec стає зрозумілим чому Google його використовує. Даний аудіокодек стискає аудіо але в той же час зберігає 100% оригінального звукового потоку. Якщо поглянути на інші звукові кодеки, наприклад MP3, WMA чи Ogg Vorbis, FLAC забезпечує стиснення без жодних втрат, і в результаті після розпакування звукові дані залишаються такими ж якими вони до початку стиснення. Однією з особливостей FLAC являється підтримка точності даних що досягається завдяки наявності у файлі відбитку MD5 оригінальних даних. [12]

Саме в такому форматі аудіозапис було відправлено сервер Google API, так як це самий оптимальний формат з тих, які Google API може прийняти для подальшого витягання аудіо.

Для простоти реалізації поставленої задачі, розпізнавання мови з відео було здійснено за допомогою витягання аудіо та подальшого його розпізнавання. Витяг аудіо відбувалося з допомогою розроблених мною функцію, в яких витягувався потік у форматі WAV з відео файлу, з подальшою її конвертацією в аудіо формат FLAC.

У Додатку Б можна побачити програмний код для конвертації відео файлу в аудіо.

Частотою дискретизації або sample rate називають кількість сигналів за одиницю часу при перетворенні в дискретний сигнал безперервного сигналу або його дискретизації. Герц являється одиницею виміру дискретизації. В випадку якщо квантування відбувається зі сталою періодичністю ми можемо застосувати поняття частоти дискретизації.

Оптимальною частотою дискретизації для нашої задачі виявилась 16000 Гц.

3.3 Інтеграція з Google Speech Recognition API

Google Cloud Speech API дозволяє розробникам конвертувати аудіо в текст, застосовуючи потужні нейромережеві моделі в зручному для використання API. API-інтерфейс розпізнає більш 80 мов і варіантів, для підтримки глобальної бази користувачів. Користувачі можуть записати текст диктуючи в мікрофон програми, включити командування і управління через голос, або записати аудіо файлів, серед багатьох інших варіантів використання.

Speech API розпізнає більш 80 мов і варіантів для підтримки глобальної бази користувачів. Ви також можете фільтрувати небажаний контент в текстових результатах.

Speech API можуть передавати текстові результати, повертаючи часткові результати розпізнавання, як вони стають доступними, а розпізнаний текст відразу з'являється в процесі говоріння. Крім того, Speech API може повертати розпізнаний текст з аудіо файлу.

Вам не потрібно передові обробки сигналу і шумозаглушення до відправки аудіо до Speech API. Служба успішно справляється з гучним звуком з різних середовищ.

Застосовуються передові глибокі вивчення нейромережевих алгоритмів для розпізнавання мовлення з неперевершеною точністю.

Для того щоб отримати доступ до Google API перш за все необхідно зареєструватися в Google Cloud Platform для отримання безкоштовного пробного ключа, що й було зроблено. Пробний ключ дозволяє робити не більше ніж 50 запитів за добу, та для наших потреб цього було достатньо для коректного тестування розробленого програмного забезпечення й складання звіту про точність розпізнавання у вигляді рисунку 3.2.

Рисунок 3.2 – Графік точності розпізнавання

С Google API ми отримуємо дані після розпізнавання в текстовому форматі JSON – текстовий формат обміну даними в мережі який зараз являється

найпопулярнішим, в ньому зберігаються пари «ключ-значення». За допомогою JSON Viewer Online ми можемо переглянути результати розпізнавання (рисунок 3.3)

За допомогою бібліотеки Newtonsoft.Json.Linq, було проведено подальший парсинг результатів, для їх обробки.

Приклад результатів можна побачити в Додатку А.

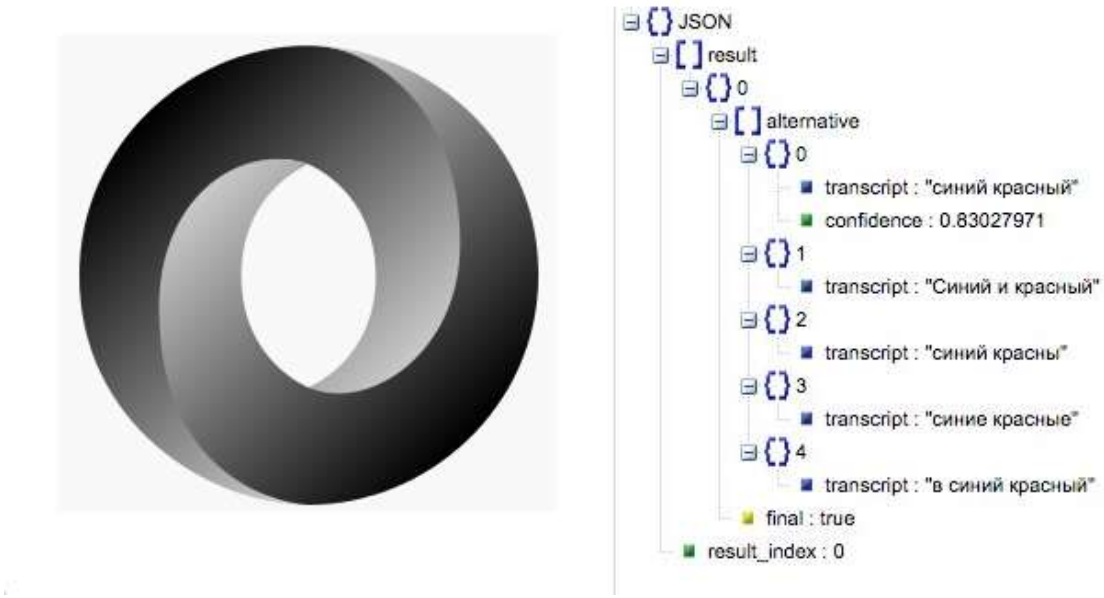


Рисунок 3.3 – JSON рядок результату розпізнавання

3.4 Пошук ключового слова

Розпізнаний текст зберігається, та відображається в якості Item до listBox, показник точності розпізнавання в свою чергу додається у richTextBox на головній формі. Далі проводиться пошук за ключовим словом, та в разі позитивного результату, виділяється певний Item у listBox, з подальшою можливістю відтворити відео, саме з того моменту, де звучала ця фраза або слова(рисунок 3.4).

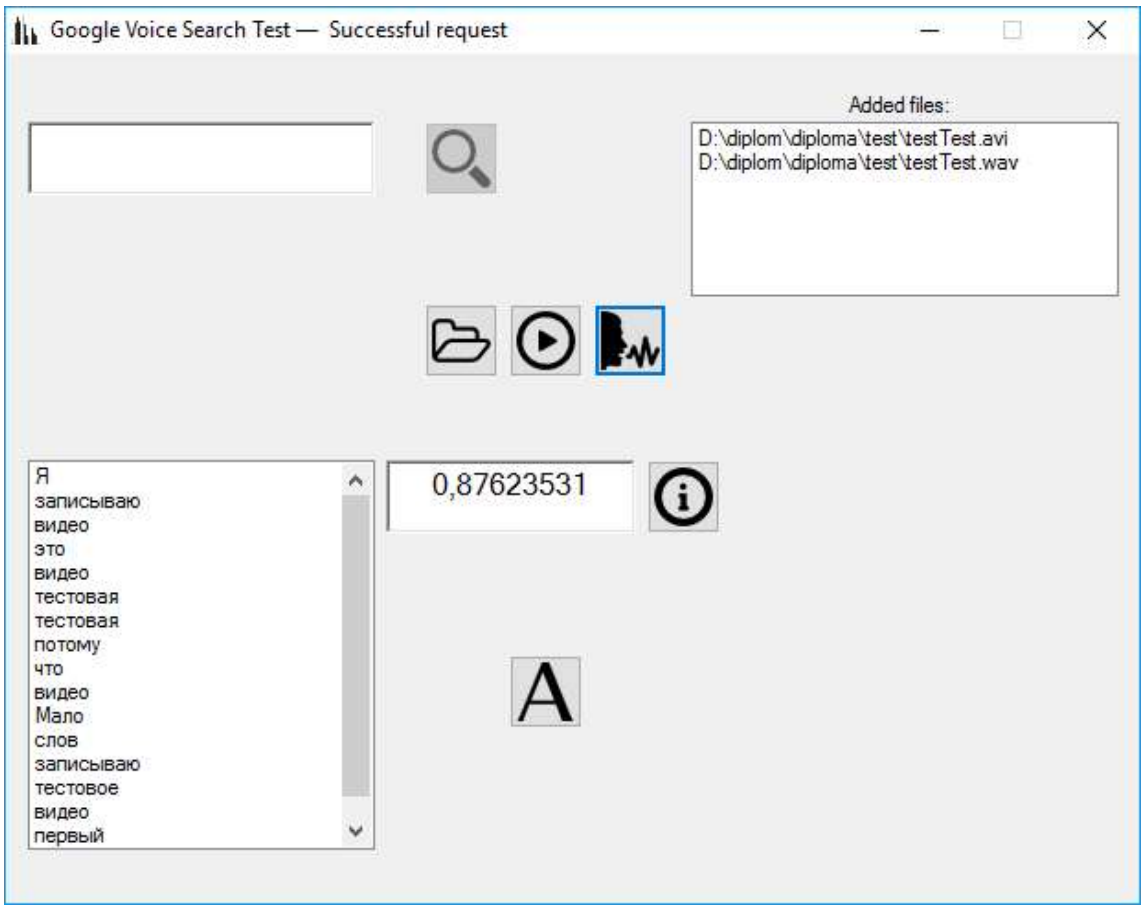


Рисунок 3.4 – Інтерфейс програми

3.5 Семантичний аналіз тексту

Наступним етапом реалізації являється семантичний аналіз розпізнаного тексту. Семантичний аналіз це етап автоматичного розуміння текстів, що основна ціль якого є виділення семантичних відносин, формування семантичного представлення текстів. Одним з можливих варіантів подання семантичного уявлення, складається з текстових фактів. Локальним семантичним аналізом являється аналіз в межах одного речення.

Зазвичай результати семантичного аналізу являються графом, семантичною мережею. Граф відображає відносини між двома вузлами – смисловими одиницями тексту. Глибина семантичного аналізу може бути різною, та найчастіше будується лише синтаксико-семантичне представлення всього тексту

або відокремленого речення. Найчастіше в глобальних системах семантичний аналіз здійснюється в той самий час коли виконується синтаксичний аналіз, це досягається за допомогою механізму розширених мереж переходів. В системі автоматичної обробки тексту поверхневого семантичного аналізу передує етап синтаксичного аналізу, за допомогою якого потім будуються вузли та відносини між ними.

Компанією Google було оголошено, що вона переходить від звичайного пошуку за ключовими словами до повністю розумного семантичного пошуку. Звісно ж світові гіганти мають масштабніші алгоритми пошуку, проте пошук в маленькій пісочниці виходить досить семантичним.

Приклад реалізації за прикладом з Розділу 2.1:

Напередодні було розглянуто основні етапи семантичного аналізу, тепер поглянімо детальніше що було реалізовано та в якій послідовності:

– видаляємо різні знаки пунктуації, а також інше сміття з тексту;
– приводимо все в нижній регістр та видаляємо всі прийменники;
– змінюємо слова до нормальної форми, наприклад якщо в тексті зустрічається одне слово в різних відмінках, ми повинні розпізнати його як єдине;
– якщо нам потрібно знайти просто схожі документи, то видаляємо слова які зустрічаються лише один раз – для аналізу схожості вони не мають ніякого сенсу і будучи віддаленими дозволять істотно заощадити пам'ять.

Далі розглянемо безпосередньо алгоритм аналізу, завдяки математичним бібліотекам тут все досить просто:

– ми складаємо матрицю нулів і одиниць, які відповідно представляють присутність або відсутність слів в документі;
– виконуємо сингулярне розкладання даної матриці, результатом чого отримуємо три інші матриці, з яких ми отримаємо координати документів та слів у просторі.

Останнім етапом нам залишається просте порівняння між собою координат документів та/або слів: ті, які знаходяться найближче один до одного і є потрібний результат, ті які подалі відповідно менш релевантні.

Також важливим етапом являється нормалізація ваги слів в матриці. Так, наприклад, слово «і» зустрічається досить часто, та воно має низьке значення, в той же час, скажімо, слово «США» зустрічається значно рідше та воно має більшу значимість. Стандартні звороти в

Останнім етапом та не останнім по важливості являється тестування створеного програмного забезпечення

Для тестування був підготовлений набір з 5 відео файлів, що були записані при різних обставинах та середовищі. У всіх випадках текст був розпізнаний повністю, та точність розпізнавання зменшувалася зі збільшенням кількості шумів. Пошук по розпізнаному тексту не потребує тестування, так як був використаний стандартний алгоритм пошуку. Обчислення точності визначення тематики відео відбувалося шляхом прослуховування відеозапису. Так як програма тестувалася на заздалегідь підготовлених відео, більш масштабне тестування не було необхідно.

Таким чином було розроблено за архітектурою товстого клієнту

Переваги мережевої архітектури клієнт-сервер «товстий клієнт»:

– товстий клієнт має широкий функціонал;
– режим багатокористувацької роботи;
– розвантаження мережевого трафіку;
– швидкодія залежить від конфігурації клієнту;
– зменшення витрат за рахунок зменшення супроводу та модернізації бізнес-логіки клієнта;

Та в свою чергу архітектури клієнт-сервер «товстий клієнт» має й перелік недоліків:

– великий розмір інсталяційного пакету для клієнту;
– працездатність клієнта залежить від платформи, під яку він розроблявся;
– підвищення рівня витрат на створення і супровід сервера.

ВИСНОВКИ

Під час виконання роботи були проаналізовані та досліджені методи розпізнавання мовлення, а також методи семантичного аналізу, такі як:

– приховані марківські моделі;

– n-грамні моделі;

– алгоритм Вітербі;

– латентно-семантичний аналіз;

– алгоритм Стеммер Портера.

Методи які стали найбільш актуальними для поставленого завдання були використані для реалізації програмного забезпечення. На протязі роботи було спроектовано та реалізовано програму розпізнавання мовлення з відео файлу та його семантичного аналізу. Якщо поглянути на світові тенденції то це є досить актуальним поняттям, й має досить велику актуальність на наш час. Результати дослідження можуть бути використані в пошукових системах, відеохостингах, сервісах з потокової передачі музики.

Окрім цього для подальшого використання та впровадження даної програми необхідно розширити функції пошуку та розпізнавання. Одним з варіантів являється проведення модернізації пошуку до таких систем як наприклад пошук по картинці в Google. Таким самим чином можливо реалізувати пошук за відривком відео, або за тематикою. Наразі мною не було знайдено жодного аналогу такої системи.

Досить схожа схема пошуку була знайдена на відео хостингу YouTube, проте пошук виконується лише за назвою відео та його тегах, дана система дозволяє отримати в результаті пошук саме за частинами фраз, словами з відео або аудіо файлу.

ПЕРЕЛІК ПОСИЛАНЬ НА ЛІТЕРАТУРНІ ДЖЕРЕЛА

1. ДСН 3.3.8.062-114. И. Л. Мазуренко. Компьютерные системы распознавания речи – 2009 – 920с.
2. Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: 2014. pp 4–25
3. Поиск оптимальной системы аудио распознавания речи // Виктор Осетров, 2014. [Електронний ресурс] - https://habrahabr.ru/post/231629/ (Дата звернення 03.01.2017)
4. Т. Хасті, Р. Тібшірані, Дж. Фрідман. Елементи статистичного навчання. Збір даних, висновок та прогнозування. 2-е видання. – Спрингер, 2013. – 335с
5. Стеммер Портера для русского языка // 2010 [Електронний ресурс] - http://www.algorithmist.ru/2010/12/porter-stemmer-russian.html (Дата звернення 10.01.2017)
6. Четвериков Г.Г., Дударь З.В., Вечирская І.Д., Дискретні структури: навч. посібник для студентів, які навчаються за напрямами "Комп'ютерні науки" і "Прикладна математика". – Харьків: ХНУРЄ, 2014. – 319 с.
7. Існує три основні групи аудіофайлів // 2011 [Електронний ресурс] - http://mylektsii.ru/2-40765.html (Дата звернення 25.01.2017)
8. Девіс К.Х., Біддульф Р. та Балашек С. Автоматичне розпізнавання мовлення розмовних цифр, 1952. – С. 637- 642.
9. Дослідження ефективності застосування марковських прихованих моделей для побудови голосових компонент інтерфейсу користувача з програмними додатками // 2014 [Електронний ресурс] - http://knowledge.allbest.ru/programming/3c0a65635a2ac79a5d53b88421206d27_1.html (Дата звернення 25.04.2017)
10. WAV // Wikipedia [Електронний ресурс] - https://uk.wikipedia.org/wiki/WAV (Дата звернення 15.04.2017)

11. Используем Google Voice Search в своем приложении .NET // 2011 [Електронний ресурс] - https://m.geektimes.ru/post/117234/ (Дата звернення 25.02.2017)
12. Gobinda G. Chowdhury. “Natural Language Processing”. In: Annual Review of Information Science and Technology, 2003. – C. 51-89.
13. FLAC // Wikipedia [Електронний ресурс] - https://uk.wikipedia.org/wiki/FLAC (Дата звернення 20.03.2017)
14. Латентно-семантический анализ // 2010 [Електронний ресурс] - https://m.habrahabr.ru/post/110078/ (Дата звернення 20.02.2017)