Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Технології голосової взаємодії з веб-орієнтованим віртуальним середовищем для людей з особливими потребами

Виконав: студент 6 курсу групи СІм-61
спеціальності
123 «Комп'ютерна інженерія»
(шифр і назва спеціальності (напряму підготовки)

Кивацький І.М.
(підпис) (прізвище та ініціали)

Керівник Лупенко С.А.
(підпис) (прізвище та ініціали)

Нормоконтроль Луцик Н.С.
(підпис) (прізвище та ініціали))

Завідувач кафедри Осухівська Г.М.
(підпис) (прізвище та ініціали)

Рецензент Литвиненко Я.В.
(підпис) (прізвище та ініціали)

м. Тернопіль – 2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Осухіська Г.М.
(підпис) (прізвище та ініціали)
« 01 » 10 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю 123 Комп'ютерна інженерія

студенту Кивацький Іван Миколайович
(прізвище, ім'я, по батькові)

1. Тема роботи Технології голосової взаємодії з веб-орієнтованим віртуальним середовищем для людей з особливими потребами

Керівник роботи *Лупенко Сергій Анатолійович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «__28__» _вересня_ 2020 року №__4/7-687_

2. Термін подання студентом роботи 21.12.2020

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналітична частина. 2 Теоретична частина. 3. Практична реалізація. 4 Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема, мета, задачі, об'єкт, предмет дослідження. 2. Актуальність. 3. Метод голосової Взаємодії. 4 Use-case діаграма. 5. Додавання нової команди 6. "Розумна" команда7. Блок схеми розпізнавання команд 8. Блок схема співставлення команд. 9. Проблема сумісності webspeech api . 10 Висновки .

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., доцент	01.11.20	07.12.20
Безпека в НС	Стадник І. Я., професор	01.11.20	09.12.20

7. Дата видачі завдання 01 10 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	28.09.20	Виконано
2	Аналіз літературних джерел	29.09.20-15.10.20	Виконано
3	Обґрунтування актуальності дослідження	15.10-21.10.20	Виконано
4	Аналіз предмету дослідження та предметної області	21.10-30.10.20	Виконано
5	Проведення дослідження методів та засобів аналітичного опрацювання даних	30.10-05.11.20	Виконано
6	Оформлення розділу «Аналітична частина»	05.11-12.11.20	Виконано
7	Оформлення розділу «Теоретична частина»	12.11-22.11.20	Виконано
8	Оформлення розділу «Практична частина»	22.11-10.12.20	Виконано
9	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	26.11-12.12.20	Виконано
10	Нормоконтроль	11.12-14.12.20	Виконано
11	Попередній захист роботи	15.12.20	Виконано
12	Захист кваліфікаційної роботи	22.12.20	

Студент _____ Кивацький М.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лупенко С.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Технології голосової взаємодії з веб-орієнтованим віртуальним середовищем для людей з особливими потребами // Кваліфікаційна робота за освітнім рівнем «магістр»// Кивацький Іван Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних систем та мереж, група СІм-61 // Тернопіль, 2020 // с. – 78, рис. – 29, аркушів А1. – 10, бібліогр. – 23.

Ключові слова: голосова взаємодія, web accessibility , webspeech api, typescript.

Кваліфікаційну роботу магістра присвячено покращенню доступності веб-орієнтованих середовищ для людей з особливими потребами, а саме реалізації методу голосової взаємодії з веб-орієнтованими середовищами для людей з особливими потребами. Проведено дослідження актуальності проблеми та розглянуто існуючі рішення. Розглянуто математичне забезпечення яке використовується для розпізнавання мовлення та фонетичних алгоритмів які дозволяють покращити результати розпізнавання мови. У даній кваліфікаційній роботі спроектовано архітектуру та програмно реалізовано утиліту для реалізації голосової взаємодії з веб середовищами. Програмну утиліти реалізовано у вигляді паттерну "Фасад" для використання WebSpeech API, котрий дозволяє спростити використання складного API для розробників за рахунок створення більш простого API як надбудови над основним.

У роботі наведені результати порівняння ефективності методів розпізнавання мовлення та фонетичних алгоритмів, описано проблеми які існують в методі голосової взаємодії з веб середовищами . Утиліта реалізовано на мові Typescript.

ANNOTATION

Technologies of voice interaction with web-oriented virtual environment for the disabled people. // Master thesis // Kyvatskyi Ivan// Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Systems and Nets, group CIm - 61 // Ternopil, 2020 // p. – 78, fig. – 29 , Sheets A1 - 10 , Ref. - 23.

Keywords: voice interaction, web accessibility , webspeech api, typescript.

The thesis dedicated to improving the accessibility of web-based environments for people with special needs, namely the implementation of the method of voice interaction with web-based environments for people with special needs. The research of urgency of a problem is carried out and the existing decisions are considered. Mathematical software used for speech recognition and phonetic algorithms that improve speech recognition results are considered. In this qualification work, the architecture and software implementation utility for voice interaction with web environments are designed. The software utility is implemented in the form of a “Facade” pattern for using the WebSpeech API, which simplifies the use of a complex API for developers by creating a simpler API as an add-on to the main one.

The paper presents the results of comparing the effectiveness of speech recognition methods and phonetic algorithms, describes the problems that exist in the method of voice interaction with web environments. The utility is implemented in Typescript.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ПММ – Прихована марковська модель

РНМ – Рекурентні нейронні мережі

ДКЧП - Довготривала короткочасна пам'ять

WCAG – Web Content Accessibility Guidelines

W3C - World Wide Web Consortium

ПНП – Пошук найкоротшого префіксу

API (Application Programming Interface) - набір чітко визначених методів для взаємодії різних компоненті.

JSGF(JSpeech Grammar Forma) - формат граматики Java Speech або формат граматики JSpeech;

ВСТУП

Актуальність теми. Згідно більшості досліджень кожна п'ята людина має певні фізіологічні відхилення, котрі вимагають особливих потреб при отриманні інформації , тому актуальність проблеми є дуже гострою. Особливі потреба поділяють на зорові(поганий зір, дальтонізм, сліпота), слухові(слабка чутливість звуку, глухота), психологічні та проблеми з моторикою(уповільнена реакція, складність використання стандартних пристрої вводу).

Кожна з цих категорій обмежень потребує певних методів та засобів адаптацій у дизайні та в роботі з системою. У більшості випадків ці адаптації покращують зручність не тільки для групи людей з конкретним відхиленням але і користь більшості користувачів. Використання корисних ілюстрацій, правильно організованого вмісту та чіткою навігацією спрощуються роботу з системою, наприклад субтитри є необхідністю для користувачів у яких є проблеми зі слухом, але вони будуть корисними і для тих користувачів які переглядають відео без звуку.

Багато хто вважає, що трата коштів, навіть нехай незначних, на адаптацію ресурсів для користувачів з особливими потребами не виправдана, але це неправильна думка, адже інвалід це такий же потенційний споживач, і він також як і всі бере участь в фінансових операціях. Ігнорування потенційних ринків через небажання витрачати невеликої кількості часу і коштів на покращення доступності системи для людей з особливими потребами при його створенні є яскравим прикладом неефективного та недалекоглядного менеджменту.

Оскільки технологія голосової взаємодії з веб-орієнтованими середовищами є мало дослідженою, важкою у використанні через можливість виникнення великої кількості помилок та поганої сумісності з сучасними фреймворками потрібно розробити утиліту(бібліотеку) яка буде розроблена згідно паттерну Фасад, тобто буде надавати простіше API для користування.

Мета дипломної роботи є удосконалення технології доступності веб-сайту для людей з особливими потребами.

Для досягнення мети наукового дослідження потрібно виконати наступні завдання:

- опрацювати літературні джерела та переглянути існуючі рішення;
- сформулювати основні проблеми з якими стикалися інші дослідники;
- спроектувати архітектуру та опрацювати можливі помилки сумісності;
- реалізувати методи покращення доступності веб-сайту;
- покрити програмний код юніт тестами для уникнення можливих помилок;
- провести ручне тестування функцій утиліти.

Об’єктом дослідження є процес розробки та вдосконалення технологій доступності віртуальних веб середовищ для людей з особливими потребами, та процес голосової взаємодії користувачів з системою для покращення доступності веб сайту.

Предметом дослідження є технології WebSpeech API для реалізації можливості користування веб-сайтом за допомогою голосового управління.

Методи дослідження. Методом дослідження було обрано симбіоз методу систематичного аналіз та методу конкретизації, спираючись на те що проблема доступності Інтернету для людей з особливими потребами є проблема яка обумовлена великою кількістю факторів які мають досить тісні зв'язки між собою, тому вимушено їх вирішувати методом систематичного аналізу. Метод конкретизації допоможе сконцентрувати вирішення конкретної проблеми і не розпорошувати її на проблеми на вирішення яких ми не можемо впливати чи вирішити. Емпіричний метод не підходять для дослідження такої проблеми через те що сама проблема виникає через те що у всіх людей різні методи відчуття, сприймання і уявлення і спиратись на власні відчуття та методи сприйняття інформації буде неправильно. Метод абстрагування призначений для інших типів

наукових досліджень, де виконується прогнозування чи планування, а не вирішення конкретної проблеми.

Наукова новизна одержаних результатів. Вперше спроектовано та програмно реалізовано утиліту, котра дозволяє спростити реалізацію методу голосової взаємодії.

Практична цінність результатів дослідження. Використання методу голосової взаємодії з веб сайтом дозволяє вивести доступність Інтернет середовища на новий рівень, це має в собі як плюс для користувачів так і плюс для компаній. Користувач отримує можливість зручного використання сайту, в свою чергу компанії отримують велику кількість потенційних клієнтів, які не могли використовувати їхній продукт через особливі потреби в користуванні.

Публікації. Основні результати, одержані у дипломній роботі магістра, опубліковані та апробовані на IX міжнародній науково - технічній конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя та на VIII науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року) як тези конференцій.

1. Лупенко С.А., Кивацький І.М. Проблема доступності Інтернету для людей з особливими потребами. Матеріали IX міжнародної науково - технічної конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: ТНТУ. 2020.
2. Лупенко С.А., Кивацький І.М. Технології голосової взаємодії з веб-орієнтованим віртуальним середовищем Матеріали VIII науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року).Тернопіль: ТНТУ. 2020.

РОЗДІЛ 1
АНАЛІЗ ДОСЛІДЖЕНЬ У СФЕРІ ДОСТУПНОСТІ ІНТЕРНЕТУ ДЛЯ ЛЮДЕЙ З ОСОБЛИВИМИ ПОТРЕБАМИ

1.1 Важливість Інтернету для людей з особливими потребами.

Хоча оцінки різняться, більшість досліджень виявляють, що приблизно п’ята частина (20%) населення має певну інвалідність(рис. 1.1). Не всі ці люди мають інвалідність, що ускладнює їм доступ до Інтернету, але це все ж значна частина населення.

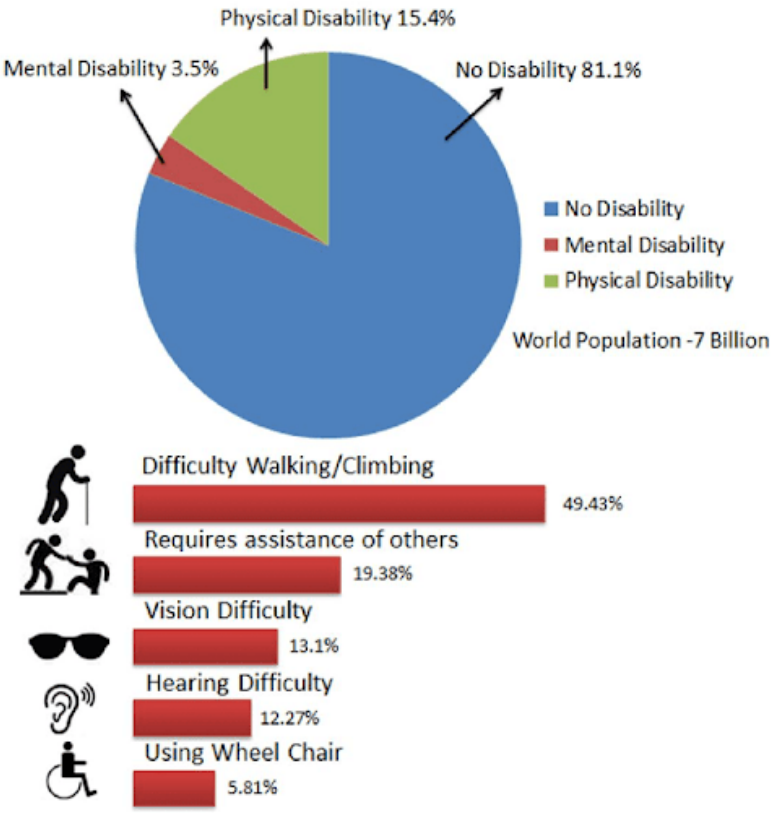


Рис. 1.1 Статистичні дані про кількість людей з фізичними та психологічними відхиленнями

Підприємствам було б нерозумно навмисно виключати 20, 10 або навіть 5 відсотків потенційних клієнтів зі своїх веб-сайтів. Для шкіл, університетів та державних структур це було б не лише нерозумно, але й у багатьох випадках також порушувало б закон.. Незважаючи на це більшість Інтернет простору взагалі не пристосовані для людей в котрих проблеми з зором(дальтонізм, сліпота), при тому що існують технології для покращення доступності сайтів.

Поняття доступності прийнято зводити до сліпоти, але насправді перелік ушкоджень котрі потребують покращення доступності є ширший:

- ушкодження слуху — повна чи часткова втрата слуху;
- ушкодження зору — погіршення якості зору чи дальтонізм вимагають можливості конфігурувати інтерфейс гнучко для власного комфорту, наприклад додавши контрасту, змінити шрифт, чи збільшивши його розмір, повна сліпота вимагає семантичної верстки для адекватної роботи скрінрідерів;
- проблеми з моторикою можуть не дозволяти користуватися стандартними методами вводу даних;
- когнітивні порушення (епілепсія, ускладнена концентрація, уповільнене сприйняття інформації) потребують якісного контенту, можливість спростити контент.[5]

Рис. 1.2 Проблема доступності Інтернету є ширшою ніж прийнято вважати

Кожна з основних категорій обмежених можливостей вимагає певних типів адаптацій у дизайні веб-контенту. Здебільшого ці адаптації приносять користь майже всім, а не лише людям з обмеженими можливостями(рис. 2.1)

Майже всі користуються корисними ілюстраціями, правильно організованим вмістом та чіткою навігацією. Так само, хоча підписи є необхідністю для глухих користувачів, вони можуть бути корисними для інших, включаючи тих, хто переглядає відео без аудіо.

При створенні по-справжньому якісного сайту, питання його доступності для всіх категорій користувачів, незалежно від їх фізичних можливостей і обмежень, повинен бути одним із першочергових завдань. Концепція доступності, яка також іноді називається accessibility, розглядає всіх людей рівноправними членами суспільства за будь-яких умовах.

Як правило, згадка про доступність для людей з особливими потребами у більшості розробників викликає роздратування, так як вважається, що даний аспект створення продукту піднімається лише в контексті якогось особливого ставлення до інвалідів та сприймається ними в ареолі соціального утриманства.

До того ж якщо ресурс створюється з нуля, то початкове закладання в нього параметрів доступності практично ніяк не збільшить кошторис проекту, тому що доступність досягається не використанням якихось додаткових дорогих надбудов, а, як правила, лише дотриманням ряду правил в рамках звичайних Інтернет-технологій.

Інтернет - одна з найкращих речей, що коли-небудь траплялися з людьми з обмеженими можливостями. Ви можливо, не думали про це таким чином, але все, що вам потрібно зробити, - це подумати про отримання інформації такими людьми до появи інформації, щоб зрозуміти, чому це так важливо.

Наприклад, перед Інтернетом, як сліпі люди читали газети? Вони в основному ні. Аудіокасети чи роздруківки шрифтом Брайля коштували дорого.У кращому випадку вони могли попросити члена сім'ї чи друга прочитати їм газету. Цей метод працює, але робить сліпих людей залежними від інших.[3]

Зараз більшість газет публікують свій вміст в Інтернеті у форматі, який може бути зчитаний за допомогою додаткового програмного забезпечення для людьми з проблемами зору. Ці програми читають електронний текст вголос, щоб сліпі люди могли користуватися комп’ютерами та отримувати доступ до будь-якого текстового вмісту через комп'ютер(рис. 1.3).

Рис. 1.3 Програма view-reader

Раптом незрячим людям не потрібно покладатися на інших людей, щоб прочитати їм газету. Їм не доведеться чекати дорогих аудіокасет чи дорогих об’ємних роздруківок шрифтом Брайля. Вони просто відкривають веб-браузер і слухають, як їх читач екранів читає їм газету, і вони роблять це самостійно, коли хочуть, і як тільки вміст публікується.

Також люди з руховими порушеннями, які не можуть забрати газету чи перегортати її сторінки, можуть отримати доступ до інтернет-газет через свій комп’ютер, використовуючи певні допоміжні технології, які адаптують комп'ютерний інтерфейс до власних вад. [3]

Адаптації є більш досконалими, як при використанні спеціальних клавіатур(рис. 1.4) або програмного забезпечення для відстеження очей, що дозволяє людям користуватися комп’ютером, не що інше, як рухи очей.

Рис. 1.4 Клавіатура Брайля

Люди з проблемами слуху можливостями можуть читати газети самостійно, але вони також можуть читати онлайн-стенограми або підписи Інтернет-мультимедійного контенту. Багато людей з когнітивними вадами також можуть отримати велику користь від структури та гнучкості веб-контенту.[6]

1.2 Існуючі дослідження

Проблему доступності Інтернету досліджує і продовжує досліджувати консорціум W3C котрий розробив рекомендації WCAG.[16]

Web Content Accessibility Guidelines, (WCAG) – керівництво по Інтернет доступності(рис. 1.5), це набір рекомендацій для того, щоб зробити сайт більш доступним як для тих, у кого є інвалідність, так і для тих, у кого її немає.[17]

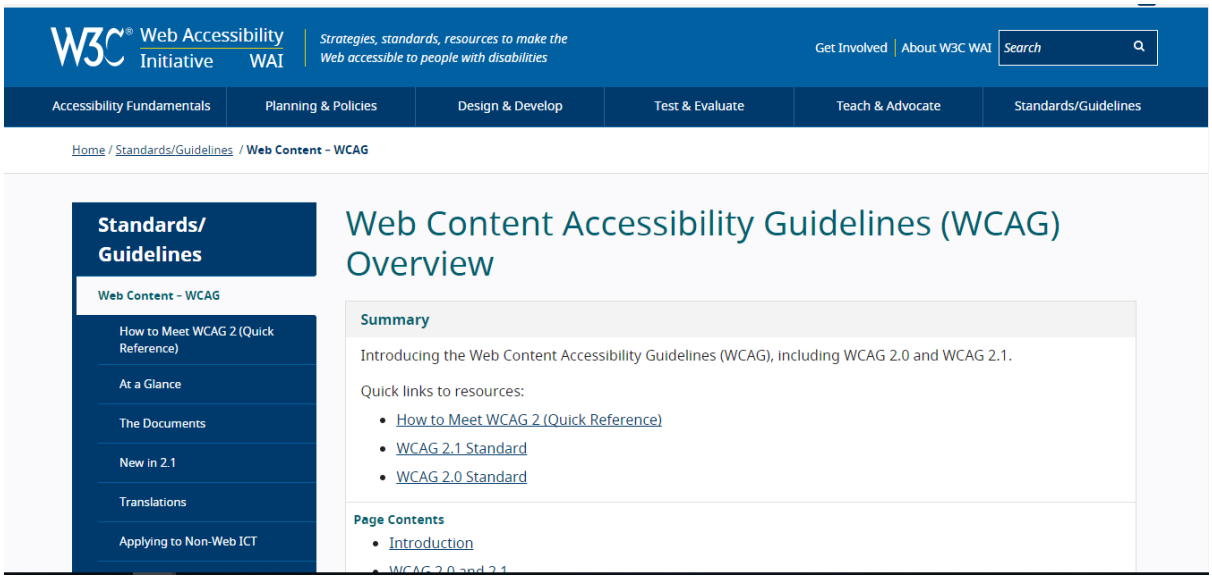


Рис. 1.5 Керівництво WCAG

Головними чотирма принципами WCAG є

- помітність;
- працездатність;
- зрозумілість;
- надійність;

12 Основних рекомендацій WCAG 2.0:

- надавайте текстові альтернативи для будь якого не текстового контенту для того щоб його можна було переводити в інші форми представлення котрі дозволяють більшій кількості користувачів отримати її корисне навантаження, е може бути: звукова представлення інформації, шрифт Брайля, мова, символи;
- надайте альтернативи та субтитри для відео, аудіо матеріалів, також потрібно додавати опис слайдів якщо використовуємо слайдшоу;
- створіть контент, який можна передати різними способами (наприклад, в більш простій розмітці), що не втрачає при цьому сенсу і структури, це покращить роботу
- зробіть так, щоб користувачам було простіше бачити і чути, і відокремте передній план від фону;

- зробіть все функції доступними з клавіатури;
- дайте користувачам достатньо часу, щоб прочитати ваш контент і скористатися ним;
- уникайте способів розміщення контенту, про які відомо, що вони викликають нервові реакції;
- надайте користувачам допомогу в навігації і пошуку контенту.
- зробіть текстовий контент читабельним і зрозумілим;
- зробіть сторінки свого сайту передбачуваними;
- допоможіть користувачам уникати помилок і виправляти їх;
- збільшуйте сумісність сайту з існуючими і майбутніми агентами(веб-браузерами) користувача, в тому числі технічними засобами реабілітації.[17]

Керівництво містить техніки і критерії оцінки для кожної з цих рекомендацій.

Також існують “Керівництво по доступності агентів користувача” (UAAG) і Керівництво по доступності інструментів авторизації (Authoring Tool Accessibility Guidelines, ATAG). UAAG допомагає зробити більш доступними агентів користувача (браузери, браузерні додатки, програвачі і так далі), а ATAG - інструменти авторизації (авторизація на сайті, в блогах і так далі).

WCAG 2.0 - основне керівництво, з яким слід ознайомитися більшості розробників. У США є закони, що вимагають робити сайти як можна більш доступними. У Канаді, Японії, Іспанії, Великобританії, Швеції та на Філіппінах теж прийняті відповідні нормативні документи.[16]

На мою думку в стандартні WCAG не надається уваги методу покращення доступності веб сайту за допомогою реалізації голосої взаємодії з сайтом, тобто користувач голосовими командами керує веб сайтом, і в свою чергу веб сайт голосовими відповідями повідомляє про результати виконання чи причини не виконання.

Такий метод роботи веб-сайту дозволить не тільки значно покращити зручність користування веб-сайтом для людей з особливими потребами але і для користувачів без фізичних вад.

Для реалізації методу голосової взаємодії з сайтом потрібно реалізувати розпізнавання мовлення щоб отримувати команди від користувача, при цьому потрібно взяти до уваги те що у користувачів можуть бути специфічні вимови чи вади мовлення, наприклад заїкання. Такі зовнішні чинники можуть погіршити якість розпізнавання тексту, тому потрібно використовувати методи та алгоритми які б максимально точно розпізнавали мовлення у звуковому сигналі не зважаючи на зовнішні чинники.

Доступ до звукового сигналу користувача у веб-браузері забезпечується через WebSpeech API інтерфейс, який в свою чергу забезпечує можливість розпізнавати текст з вхідного аудіо потоку (зазвичай через пристрій розпізнавання телефону за замовчуванням) і відповідати відповідно.[18]

Також Web Speech API дозволяє веб-сайту відповідати користувачу не тільки у вигляді стандартних методів виводу інформації, а і за допомогою звукових відповідей на дії користувача, цю функцію забезпечує SpeechSynthesis інтерфейс котрий дозволяє синтезувати звукову інтерпретацію текстових повідомлень(рис 1.6).

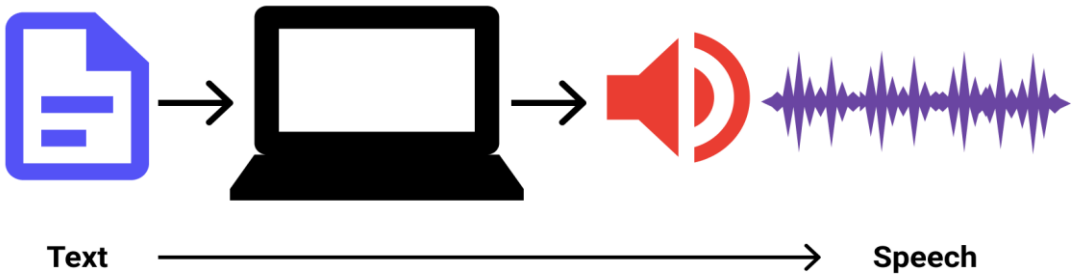


Рис. 1.6 SpeechSynthesis

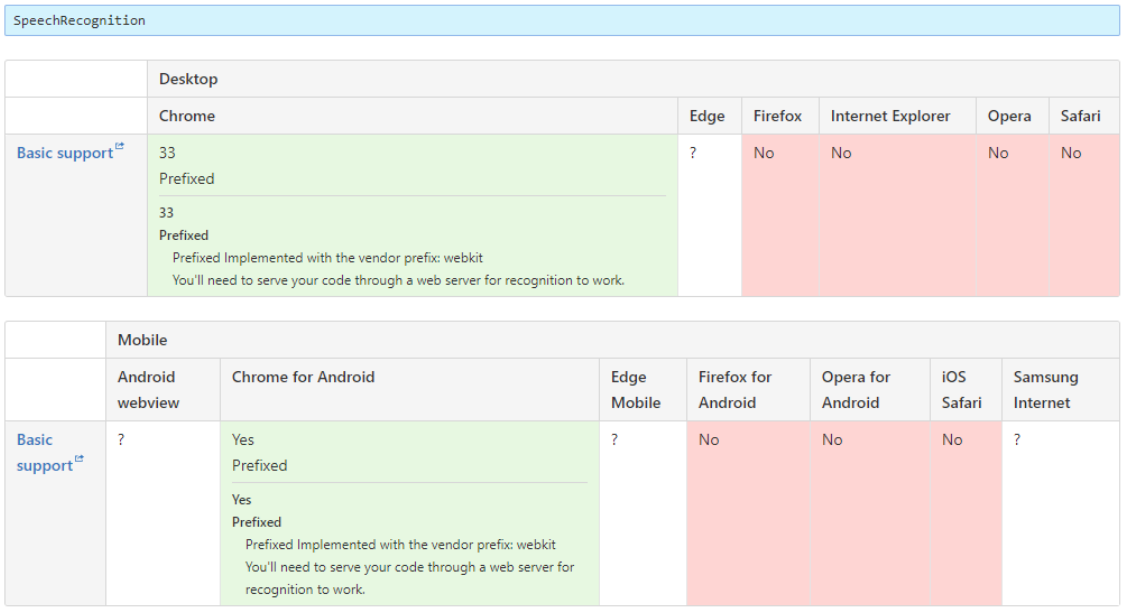
Скориставшись конструктором інтерфейсу ви можете створити новий SpeechRecognition об'єкт, котрий буде отримувати звуковий сигнал з обраного мікрофона та опрацьовувати його, у нього є ряд подій для виявлення початку мовлення, закінчення мовлення, початок його розпізнавання та завершення розпізнавання з отриманим результатом. через мікрофон пристрою(рис 1.7). SpeechGrammar інтерфейс надає контейнер для певного набору граматики, яке ваше застосування повинне використовувати. [18]

Рис. 1.7 Інтерфейс SpeechRecognition

Граматика визначається за допомогою JSpeech Grammar Format (JSGF.) , розроблений формат JSGF компанією Sun Microsystems, це текстове відображення граматик для використання в розпізнаванні мови для таких технологій, як XHTML + Голос. JSGF приймає стиль та конвенції мови програмування Java на додаток до використання традиційних граматичних позначок.[18]

Коли слово чи фразу успішно розпізнано вона повертається як результат (або список результатів) як текстовий рядок, і в результаті можуть бути ініційовані подальші дії. Загалом, система розпізнавання мови за замовчуванням, наявна на пристрої, буде використовуватися для розпізнавання мови - більшість сучасні ОС мають систему розпізнавання мови для видачі голосових команд.

Недоліком методу голосової взаємодії з веб-сайтом є проблема сумісності з застарілими браузерами(рис. 1.8 і 1.9). Це зумовлено тим що API інтернет-браузерів не є стандартизовано, і такі браузери як Chrome та Firefox підтримують більш сучасні технології, які дозволяють зробити веб-сайт більш доступний.[19]

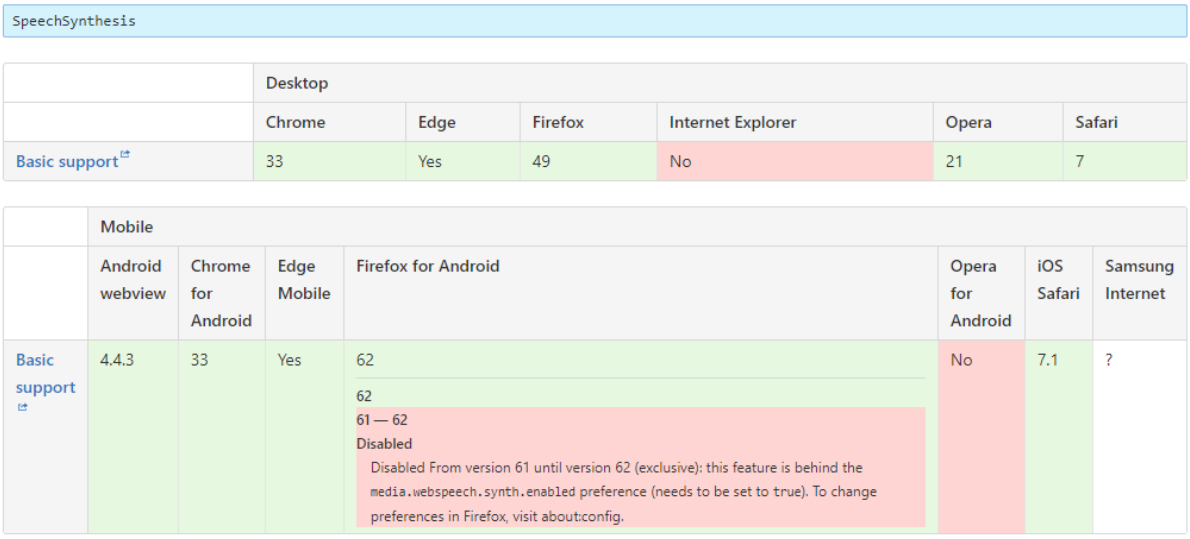


SpeechRecognition

	Desktop					
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic support	33 Prefixed 33 Prefixed Prefixed Implemented with the vendor prefix: webkit You'll need to serve your code through a web server for recognition to work.	?	No	No	No	No

	Mobile						
	Android webview	Chrome for Android	Edge Mobile	Firefox for Android	Opera for Android	iOS Safari	Samsung Internet
Basic support	?	Yes Prefixed Yes Prefixed Prefixed Implemented with the vendor prefix: webkit You'll need to serve your code through a web server for recognition to work.	?	No	No	No	?

Рис. 1.8 Підтримка технології SpeechRecognition в браузерах



SpeechSynthesis

	Desktop					
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic support	33	Yes	49	No	21	7

	Mobile						
	Android webview	Chrome for Android	Edge Mobile	Firefox for Android	Opera for Android	iOS Safari	Samsung Internet
Basic support	4.4.3	33	Yes	62 62 61 — 62 Disabled Disabled From version 61 until version 62 (exclusive): this feature is behind the media.webspeech.synth.enabled preference (needs to be set to true). To change preferences in Firefox, visit about:config.	No	7.1	?

Рис. 1.9 Підтримка технології SpeechSynthesis в браузерах

1.3 Висновки до розділу

Проаналізовано проблему доступності веб-орієнтованих середовищ для людей з особливими потребами у результаті якого виявлено, що проектування та розробка методу голосової взаємодії є актуальною задачею і потребує додаткового дослідження та реалізації.

Проведено аналіз сучасних підходів та існуючих рішень щодо покращення доступності веб-орієнтованих середовищ для людей з особливими потребами. На основі аналізу існуючих рішень встановлено, що існуючі методи покращення доступності дають недостатній результат.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ ДОСТУПНОСТІ ІНТЕРНЕТУ ДЛЯ ЛЮДЕЙ З ОСОБЛИВИМИ ПОТРЕБАМИ

2.1 Розпізнавання мовлення

Розпізнавання мовлення процес перетворення звукового запису в текстове відображення. Системи розпізнавання мовлення класифікуються за кількома характеристиками.

За типом структурних одиниць поділяються на такі які використовують:

- фрази;
- слова;
- фонеми;
- алофони(фонема яка змовлена конкретним фонетичним оточенням).

За алгоритмом який використовується поділяються на:

- нейронні мережі;
- прихована Марковська модель(Hidden Markov Model);
- динамічне програмування(Dynamic Time Warping);
- методи дискримінантного аналізу.

Також за розміром бувають системи з обмеженим набором слів та з словниками великого об’єму.

Основні характеристики які описують системи розпізнавання мовлення та відтворення мови:

- розбірливість мови є відносною величиною елементів мовлення які були розпізнані правильно(звуки, фрази, склади, слова) визначається як відсоток від кількості вхідних елементів;
- якість мовлення характеристика котра дає оцінку якості мовлення яке потрапило котре потрапляє через системи передачі;

− нормальний темп мовлення - вимова з певною сталою швидкістю, при котрій середня тривалість контрольної фрази равн рівна 2.4с

− пришвидшений темп мовлення - вимова зі швидкістю, при якій середня тривалість контрольної фрази є більшою чи рівною 1.5с

− цілісність інформації характеристика яка вказує в якій мірі зберігається корисне навантаження мовлення.

2.2 Прихована Марковська модель

Прихована марковська модель(Hidden Markov model) є статистичною марковською моделлю, у котрій систему яку моделюють прийнято розглядати як конкретний марковський процес з деякими прихованими станами.

Можна представити прихована марковська модель як одну з найпростіших динамічних Баєсових мережу, котра є баєсовою мережею що співвідносить змінні між собою через певні суміжні проміжки часу, зображено на рисунку 2.1.

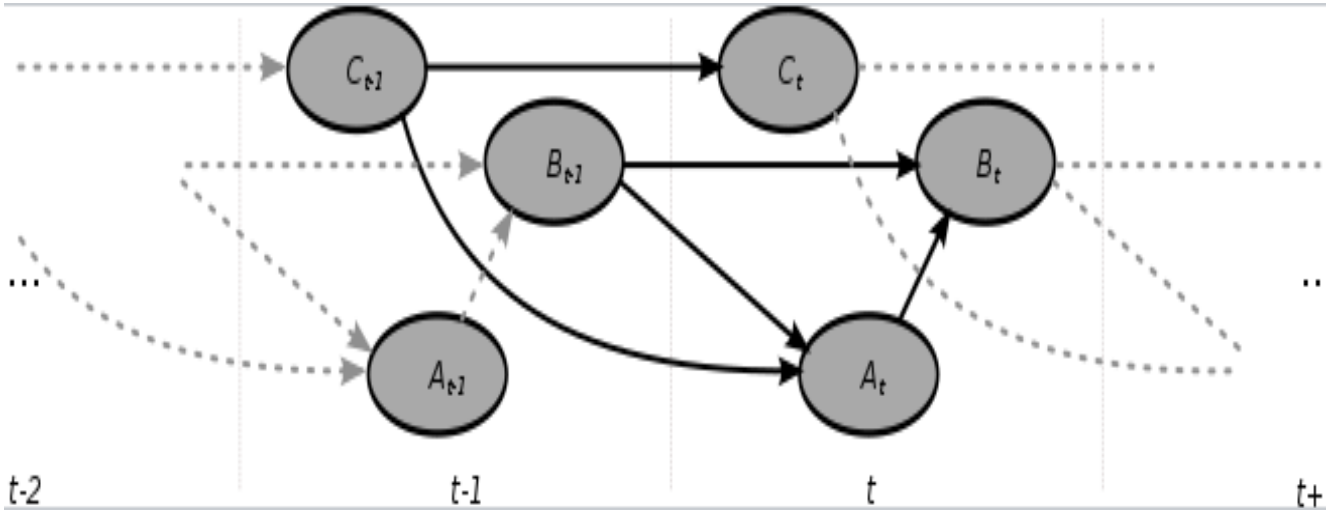


Рис. 2.1 Динамічна баєсова мережа, складена з 3 змінних.

У стандартному типі прихованої моделі Маркова, що розглядається тут, простір станів прихованих змінних є дискретним, тоді як самі спостереження можуть бути або дискретними (як правило, генеруються з категоріального

розподілу), або безперервними (як правило, з гаусового розподілу). Параметри прихованої моделі Маркова бувають двох типів - ймовірності переходу та ймовірності викидів (також відомі як вихідні ймовірності) Усі стани містить ймовірнісний розподіл вихідних значень котрі можуть бути.[9]

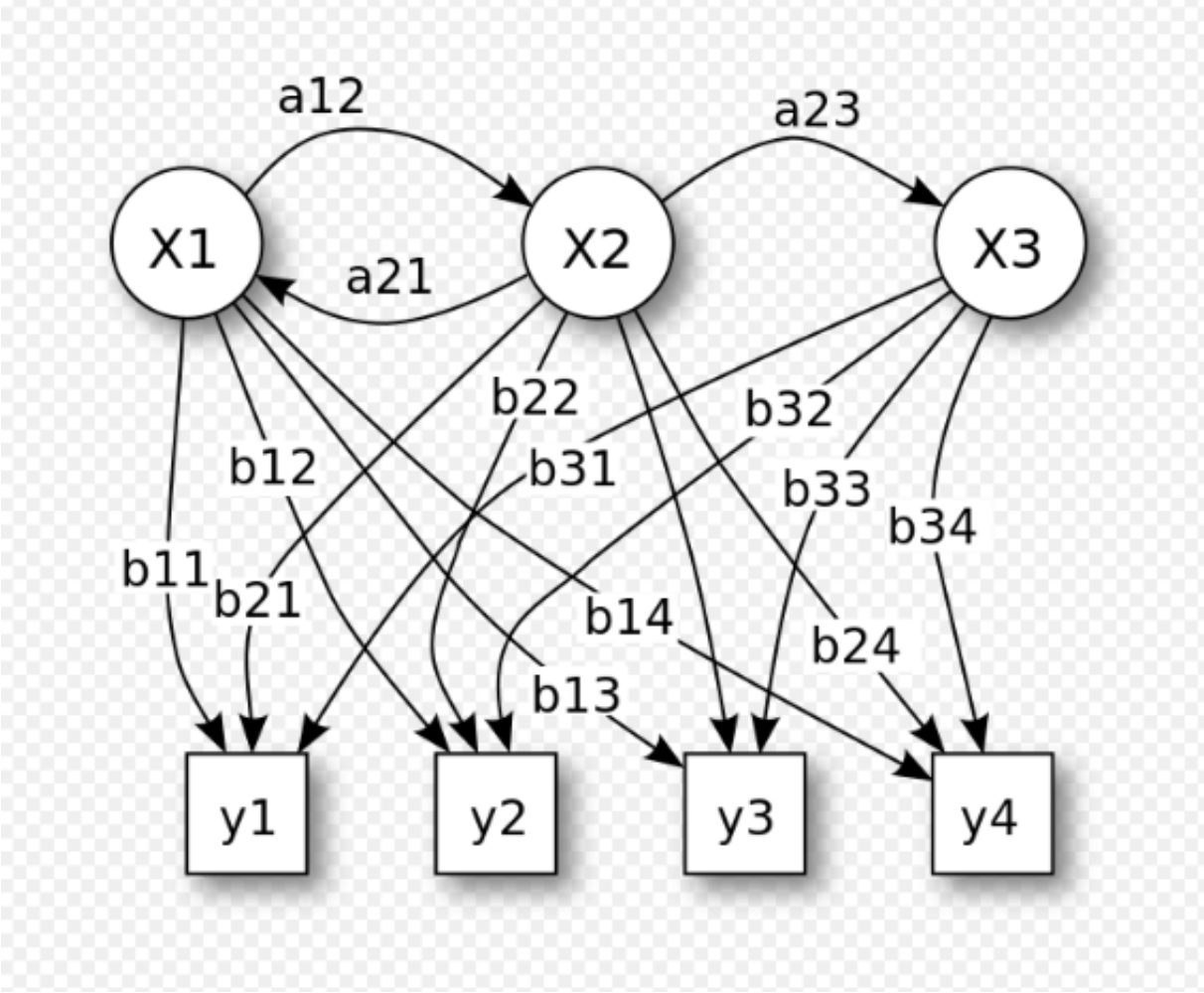


Рис. 2.2 Приклад ймовірнісних параметрів прихованої марковської моделі

На рисунку 2.2 зображений приклад ймовірнісні параметри прихованої марковської моделі, де

- x –стани;
- y – можливі спостереження;
- a –ймовірності переходів станів;
- b – ймовірності виходів.

Приховані марковські моделі можуть розглядатися як узагальнення певної сумішевої моделі, де латентні змінні, котрі контролюють, яка складова суміші буде обиратися відповідно кожному спостереженню, пов'язані марковським процесом, а не є незалежними одна від одної.

Часто приховані марковські моделі узагальнюють(pairwise Markov models) та триплетних марковських моделей (triplet Markov models), і це дозволяє розглядати більш складні складніші структури даних, та моделювати нестаціонарні дані.

У своїй дискретній формі прихований марковський процес можна представити як певне узагальнення задачі про урни із поверненням (де кожен елемент з урни перед наступним кроком повертається до своєї урни).[9]

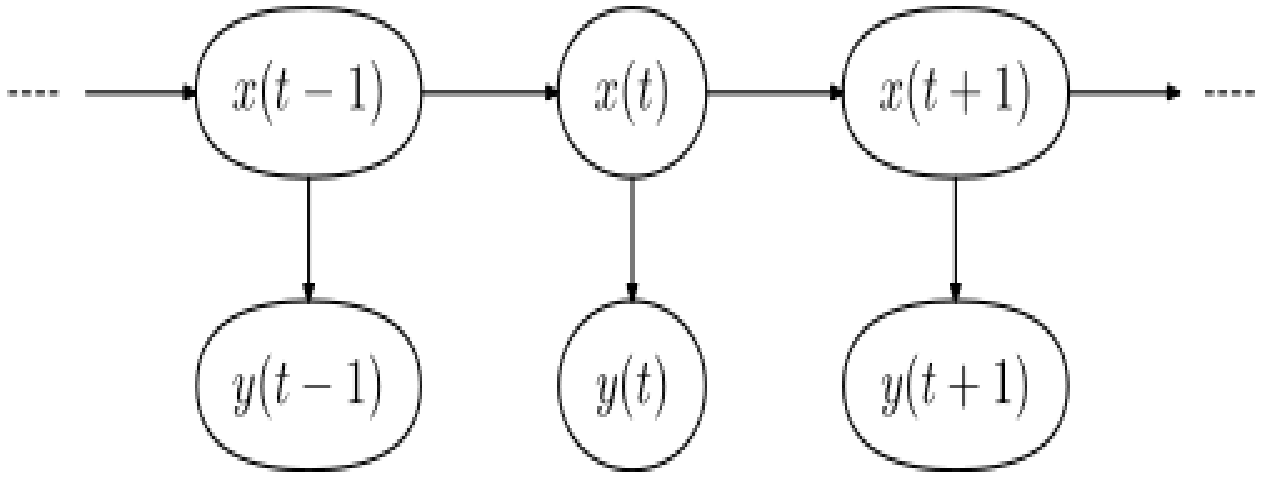


Рис. 2.3 Загальна архітектура приладу ПММ

Рисунок 2.3 показує загальну архітектуру екземпляра ПММ. Кожнен з овалів являє собою випадкову величину, яка може прийняти будь-яке з ряду значень. Випадкова величина x (t) є прихованим станом у момент часу t (з моделлю з наведеної діаграми)

Випадкова змінна y(t) є спостереженням у момент часу t (де y(t) ∈ { y1, y2, y3, y4 }.

Діаграму зображену на рисунку 2.3 прийнято називати гратковою діаграмо, стрілки на ній позначають ймовірнісні залежності.

Це називається марковською властивістю. Так само, значення спостережуваної змінної y(t) залежить тільки від самого значення прихованої змінної x(t), саме у той же момент часу t.

Параметри ПММ бувають двох типів ймовірності виходів та ймовірності переходів. Ймовірності переходу контролюють спосіб вибору прихованого стану в момент часу t, враховуючи прихований стан у момент часу

Прийнято вважати, що прихований простір станів складається одного з N можливих значень, котрі є змодельовані як категоріальний розподіл.

Це означає, що для кожного з N можливих станів, в яких може перебувати прихована змінна в момент часу t, існує ймовірність переходу з цього стану в кожний з N можливих станів прихована змінна під час t + 1 для загальної кількості ймовірностей переходу N^2.

Отже, матриця N x N є матрицею Маркова. Оскільки будь-яку одну ймовірність переходу можна визначити, як тільки відомі інші, існує загальна кількість параметрів переходу N (N-1).

Крім того для кожного з N потенційно можливих станів існує набір ймовірностей виходів, котрий повинен керувати розподілом спостережуваної змінної у конкретний момент часу для встановленого стану прихованої змінної в момент часу. Розмір цього напряму є залежним від природи спостережуваної змінної. [10]

З іншого боку, у випадку коли спостережувана змінна є M-вимірним вектором з відповідний розподілом до обраного багатовимірного нормального розподілу, то буде M параметрів, котрі виконують контроль над середніми, та M(M+1)/2 параметрів, котрі повинні контролювати коваріаційну матрицю, загальним числом параметрів виходу.

Коли умова задовольняється , якщо значення M є не малим, є зручніше обмежити природу коваріацій між конкрентими елементами вектора

спостережень, для цього наприклад можемо припустити, що елементи не є незалежними овід інших, або, більш широко , залежать від тільки від фіксованого числа елементів які знаходяться поруч.

2.3 Рекурентні нейронні мережі

Останніми роками широкого використання набувають методи на основі рекурентних нейронних мереж (RNN).

Рекурентними нейронними мережами(recurrent neural networks) називають клас штучних нейронних мереж. В цьому клас з'єднання між вузлами утворюють граф котрий є орієнтований у часі. Орієнтований граф утворює внутрішній стан мережі, це дозволяє мережі проявити динамічну поведінку в часі.

На відміну від нейромереж прямого розповсюдження, РНМ часто для обробки довільних послідовностей входів використовують свою внутрішню пам'ять.Таке їхнє використання дозволяє застосовувати їх до таких задач, як розпізнавання сегментованого неперервного рукописного тексту та розпізнавання мовлення.[7]

Основною архітектурою РНМ є повнорекурентна мережа, мережа нейроподібних вузлів, кожен з орієнтованим з'єднанням до кожного іншого вузла. Кожен з вузлів має змінну в часі дійснозначну активацію.

Одні з вузлів називаються вхідними вузлами, деякі — вихідними, а решту прихованими.

Для контрольованого навчання в умовах дискретного часу послідовності реальних значень вхідних векторів надходять на вхідні вузли, по одному вектору. На будь-якому даному кроці часу кожна невхідна одиниця обчислює свою поточну активацію (результат) як нелінійну функцію зваженої суми активацій усіх підключених до неї одиниць.

Цільові активації, які надає супервізор, можуть надаватися для деяких вихідних одиниць на певних етапах часу. Наприклад, якщо вхідна послідовність є

мовним сигналом, що відповідає вимовленій цифрі, кінцевим цільовим виходом в кінці послідовності може бути мітка, що класифікує цифру.

В навчальних налаштуваннях підкріплення жоден учитель не подає цільові сигнали. Натомість для оцінки продуктивності РНМ іноді використовують функцію винагороди або функція допасованості , яка впливає на її вхідний потік через вихідні блоки, підключені до виконавчих механізмів, що впливають на навколишнє середовище. Це може бути використано для гри, в якій прогрес вимірюється кількістю виграних очок.[7]

Кожна послідовність видає помилку як суму відхилень усіх цільових сигналів від відповідних активацій, обчислених мережею. Для навчального набору з численних послідовностей загальна помилка - це сума помилок усіх окремих послідовностей.

Одним з інших видів РНМ є рекурсивні нейронні мережі, вони створюються рекурсивним застосуванням того самого набору ваг до графоподібної структури яка є диференційовною шляхом обходу цієї структури в топологічній послідовності.

Такі мережі прийнято тренувати зворотним режимом авто-диференціювання Їх було введено для навчання розподілених представлень структури, таких як терміни логіки. Оскільки структура відповідає лінійному ланцюжкові вони є окремим випадком рекурсивних нейронних мереж є самі РНМ, чия структура відповідає лінійному ланцюжкові. [9]

Рекурсивні нейронні мережі застосовують для обробки природної мови. На рисунку 2.4 зображено приклад простої рекурсивної нейронної мережі.

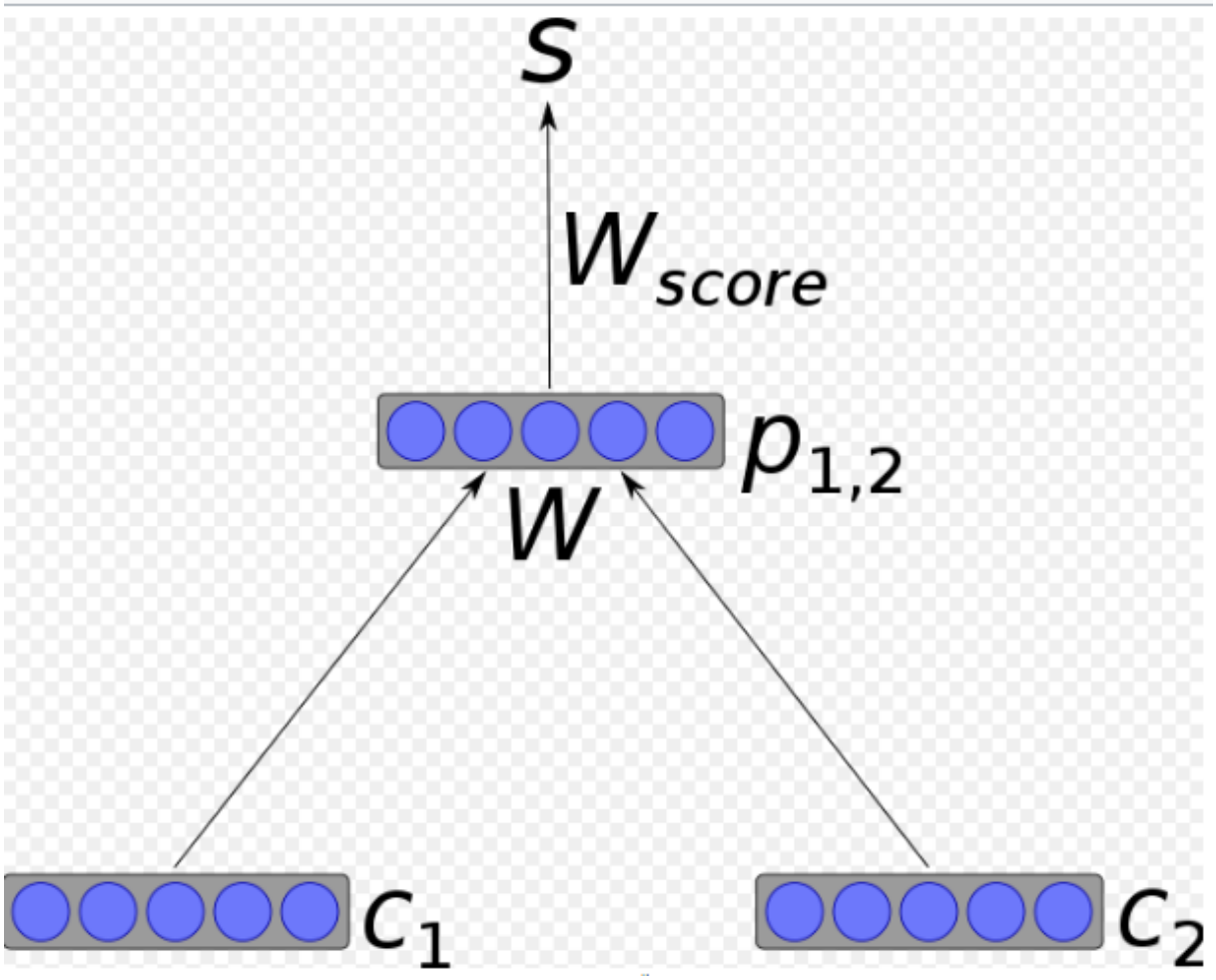


Рис. 2.4 Приклад простої рекурсивної нейронної мережі

2.4 Довга короткочасна пам'ять

Для задач розпізнавання мовлення найчастіше застосовують архітектуру довгої короткочасної памяті.(long short-term memory) при використанні рекурентних нейронних мереж. Дана архітектура була запропонована наприкінці 90-х.

Як і більшість РНМ мережа довгої короткочасної пам'яті є універсальною тому що за умови достатньої кількості вузлів у мережі вона здатна обчислювати усе те саме що може обчислювати звичайний комп'ютер, за умови, що вона має відповідну матрицю вагових коефіцієнтів, котра буде розглядатися як програма мережі. [2]

Мережі ДКЧП добре підходять для класифікації, обробки та прогнозування на основі даних часових рядів, оскільки між важливими подіями часового ряду можуть бути затримки невідомої тривалості. Вони були розроблені для вирішення проблеми зникаючого градієнта, з якою можна зіткнутися при навчанні традиційних РНМ.

Відносна нечутливість до довжини зазору є перевагою ДКЧП перед РНМ, прихованими марковськими моделями та іншими методами навчання послідовності у багатьох додатках.

У наш час основні технологічні компанії, включно з Google, Apple, Microsoft, використовують мережі ДКЧП для продуктів які реалізують розпізнавання мовлення.

Існує кілька архітектур блоків ДКЧП. Загальна архітектура складається з комірки (частина пам'яті блоку ДКЧП) і трьох "регуляторів", як правило, називаються воротами, потоку інформації всередині блоку ДКЧП: вхідного шлюзу, вихідного шлюзу та забутого шлюзу. Деякі варіанти блоку ДКЧП не мають одного або декількох із цих воріт або, можливо, мають інші ворота. Наприклад, закриті рекурентні одиниці не мають вихідного шлюзу.

Інтуїтивно комірка відповідає за відстеження залежностей між елементами у вхідній послідовності. Вхідний затвір контролює ступінь надходження нового значення в комірку, ворота забуття контролює ступінь, до якого значення залишається в комірці, а вихідний шлюз контролює ступінь, в якому значення в комірці використовується для обчислення вихідних даних активація блоку ДКЧП. Функція активації воріт ДКЧП часто є логістичною сигмовидною функцією. Блоки ДКЧП прийнято тренувати методом зворотного поширення в часі. [2]

ДКЧП поділяються на традиційні, вічкові та згорткові, традиційна ДКЧП із забувальними вузлами $c_0=0$ і $h_0=0$, о позначає поелементний добуток(добуток Адамара):

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \qquad (2.1)$$

$$i_t = \sigma_g(W_f x_t + U_i h_{t-1} + b_i) \quad (2.2)$$

$$o_t = \sigma_g(W_f x_t + U_o h_{t-1} + b_o) \quad (2.3)$$

$$c_t = f_t \circ C_{t-1} + i_t \circ \sigma_c(W_f x_t + U_c h_{t-1} + b_c) \quad (2.4)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (2.5)$$

де x_t – вхідний вектор;

h_t – вихідний вектор;

c_t – вектор комірки;

W,U, b – матриці та вектор параметрів;

f_t, - вектор забувального вентиля;

i_t - вектор вхідного вентиля, є вагою отримання інформації.;

o_t - Вектор вихідного вентиля, описує кандидатність на вихід;

σ_g, σ_c, σ_h – функції активації.

Вічкова ДКЧП із забувальними вентилями. h_{t-1} y_t застосовується, натомість у більшості місць застосовується c_{t-1}.

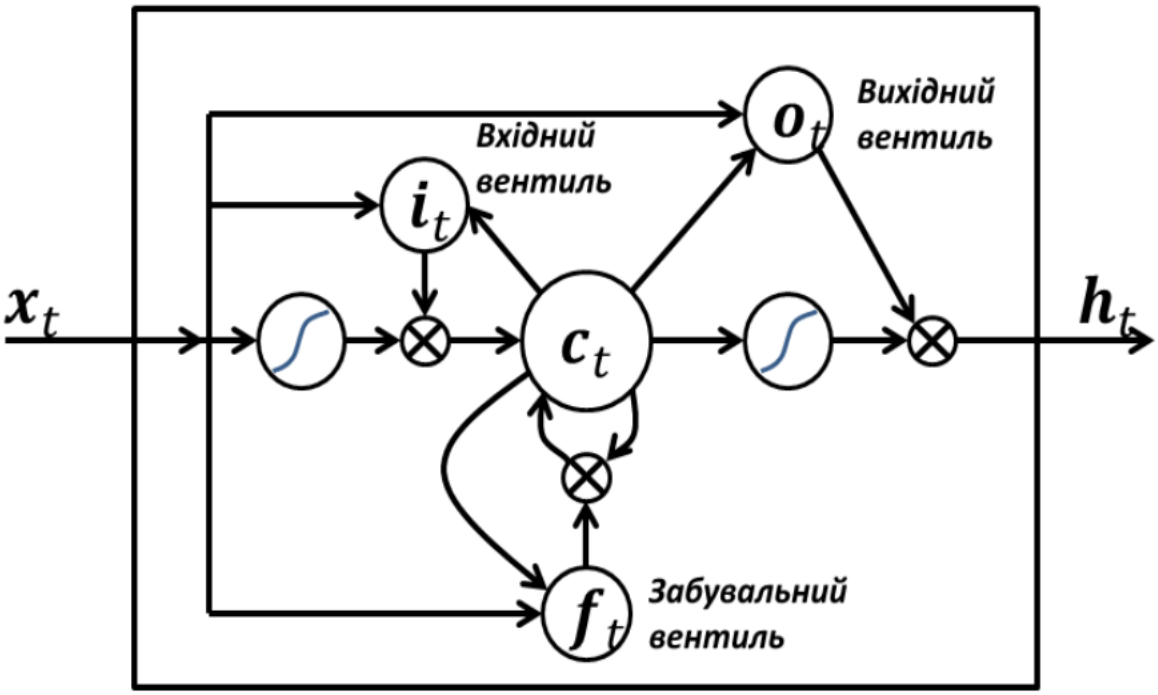


Рис. 2.5 Блок вічкової ДКЧП

На рисунку 2.4 зображено блок вічкової ДКЧП з забувальним, вхідним та вихідним вентилями. Вихідні стрілки з вузла c_t насправді позначають виходові стрілки з C_{t-1}, крім єдиної стрілки зліва направо.

Згорткова ДКЧП, замість звичайного добутку використовує оператор згортки.

РНМ з використанням одиниць ДКЧП можна тренувати під контролем, на наборі навчальних послідовностей, використовуючи алгоритм оптимізації, такий як градієнтний спуск, поєднаний із зворотним розповсюдженням у часі для обчислення градієнтів, необхідних під час процесу оптимізації, для зміни кожної ваги мережі ДКЧП пропорційно похідній похибки (на вихідному рівні мережі ДКЧП) щодо відповідної ваги.

Проблема використання градієнтного спуску для стандартних РНМ полягає в тому, що градієнти помилок швидко зникають в геометричній прогресії із величиною часового лагу між важливими подіями.

2.5 Розпізнавання мовлення за допомогою РНМ

Для розпізнавання мовлення використовують модифіковані РНМ, тому що стандартна структура не підходить для виконання задач розпізнавання мови, тому що коли відбуваються розрахунки відхилень та ваг малі помилки зникають надто швидко, а великі надто швидко зростають, це зумовлено великою кількістю множень.

Використовуються ДКЧП комірки. Комірка ДКЧП описується наступною формулою:

$$c_t = f_t c_{t-1} + i_t \tanh(W_{hc} h_{t-1} + W_{xc} x_t + b_c) \qquad (2.6)$$

де x_t – вхідний вектор;

c_{t-1} – вектор комірки;

W, b – матриці та вектор параметрів;

f_t, - вектор забувального вентиля;

i_t - вектор вхідного вентиля, є вагою отримання інформації.

Згідно формули 2.6 коли $f_t = 0$ комірка забуває попереднє значення і записує нове. Коли $i_t = 1$ значення встановлюється спираючись на вхідні дані та дані попереднього прихованого шару. [11]

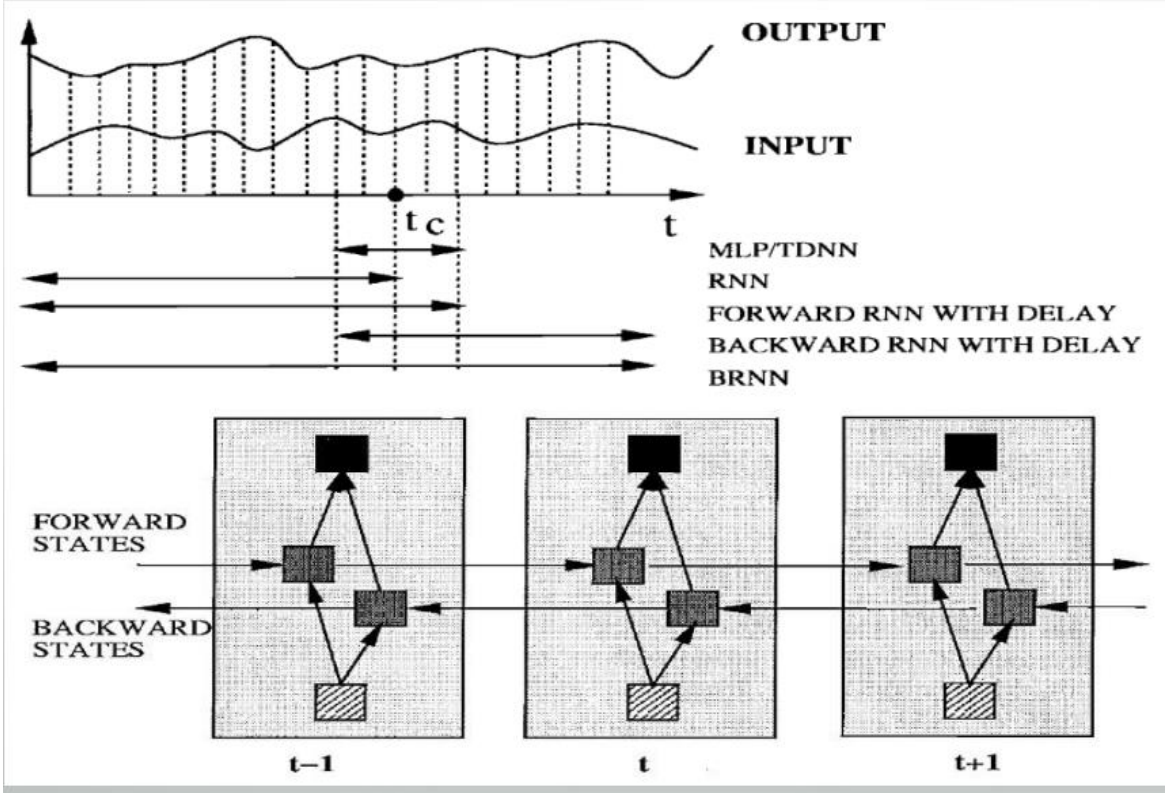


Рис. 2.6 Діаграма РНМ для розпізнавання мовлення

Наступним кроком відбувається навчання акустичної моделі, для цього використовують CTC (Connectionist Temporal Classification).

Основною ідеєю CTC є те, що замість того, щоб генерувати стрічку як виведення з нейронної мережі, замість цього потрібно генерувати розподіл імовірності на кожному кроці в часі (від t = 1 до t = T).

Після цього є можливість розшифрувати розподіл ймовірностей використовуючи принцип максимальної правдоподібності і знайти потрібну стрічку стрічку.

Наступним кроком, потрібно виконати навчання мережі, створивши цільову функцію, котра виконує пошук максимальної ймовірності декодування для заданої послідовності x, для досягнення ℓ мітки.[11]

Після отримання розподілу ймовірності потрібно декодувати отриманні розподіли за допомогою алгоритму пошуку найкращого префіксу(ПНП)(рис 2.7).

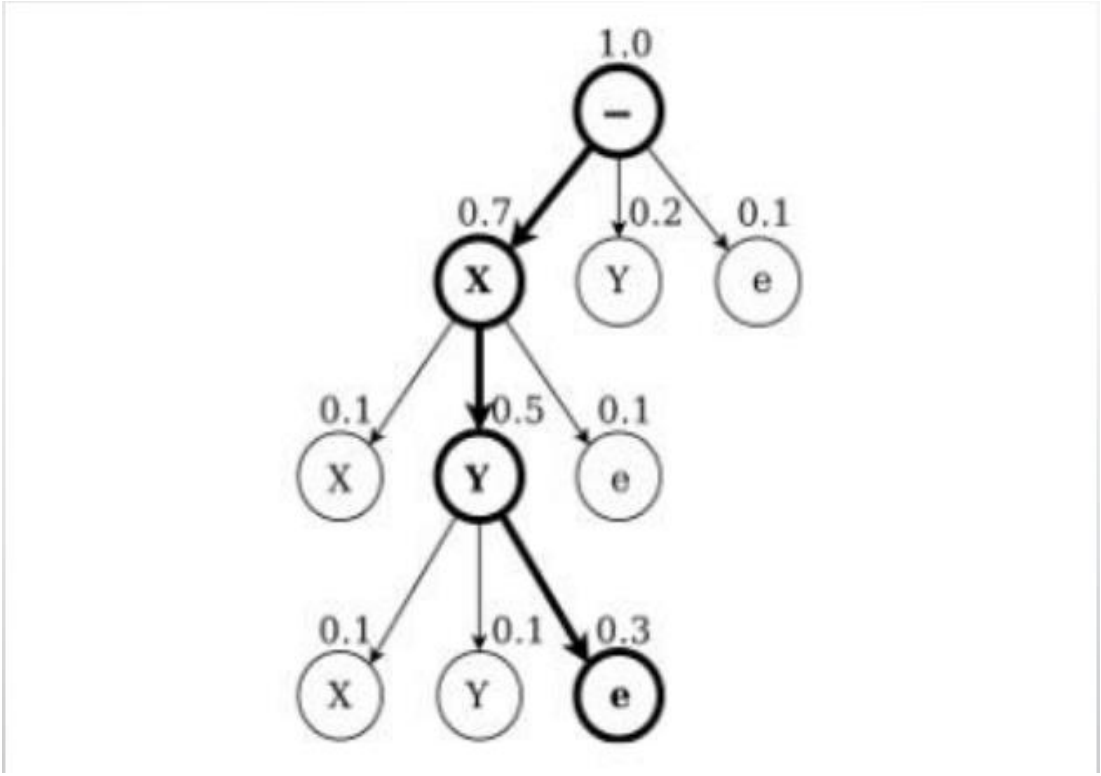


Рис. 2.7 Алгоритм пошуку найкращого префіксу

Параметри моделі

- 100 блоків ДКЧП в прихованих шарах зворотного і прямого поширення;
- активаційна функція , сигмоїд для воріт комірок ДКЧП або гіперболічний тангенс для вхідних і вихідних комірок;
- приховані шари пов’язані самі з собою, та відповідно з виходами і входами;
- вхідний шар містить 26 нейронів, вихідний шар з 62 юнітів (61 фонема та 1 вихід для пробілу);
- загальна кількість ваг рівна 114 або 662;
- для навчання використовувався метод зворотного поширення помилки і поточного градієнту (тобто ваги змінюються після кожного циклу тренування);
- похибка навчання 10^{-4};
- значення активаційних функцій мережі скидаються на 0 для кожного етапу тренування;
- при використанні методу ПНП необхідна ймовірність пробілу рівна 99,99%;
- ваги ініціалізуються згідно рівномірного розподілу з межами (-0,1 ; 0,1);
- під час навчання додається Гаусівський шум із стандартним відхиленням 0,6 для покращення узагальненості навчання.[11]

2.6 Soundex алгоритм

Soundex алгоритм це фонетичний алгоритм котрий використовується для індексації слів за вимовою в англійській мові. Він виконує однакове представлення різних омофонів, що значно пришвидшує їх пошук, незважаючи на неточності(рис. 2.8).

Алгоритм переважно опускає голосні звуки і використовує тільки приголосні звуки, голосні опускаються, окрім першої букви.

Soundex є найвідомішим серед усіх фонетичних алгоритмів (він використовується популярними СКБД MySQL PostgreSQL, та Oracle), та часто використовується як синонім до «фонетичного алгоритму».

Удосконалення Soundex є основою для багатьох сучасних фонетичних алгоритмів. Soundex код складається з букви й трьох числових розрядів: першу літеру імені й цифри кодування наступних приголосних.

Подібні приголосні мають одні й ті ж значення. Голосні звуки можуть вплинути на кодування, але кодується тільки перша літера.

Для пошуку вірного значення потрібно виконати наступні кроки:

1. Перша літера залишається незмінною;
2. Кожна приголосна кодується відповідним кодом:

- b, F, P, V => 1;
- c, G, J, K, Q, S, X, Z => 2;
- d, T => 3;
- l => 4;
- m, N => 5;
- r => 6;
- h, W ігноруються в процесі кодування.

3. Дві сусідні літери у випадку однакового числа кодуються як одне, також літери з однаковим числом, розділених h або w кодуються як одне число;
4. Продовжуєте допоки не буде однієї букви і трьох цифр.

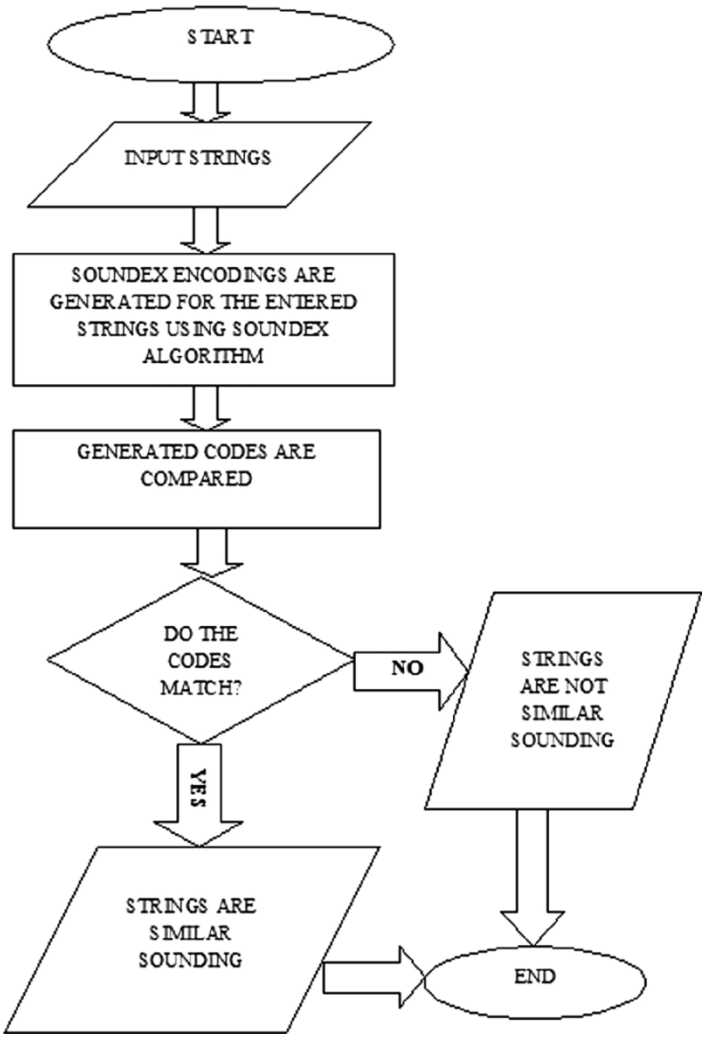


Рис. 2.8 Блок схема Soundex алгоритму

2.7 NYSIIS алгоритм

Фонетичний код, відомий як NYSIIS, є фонетичним алгоритмом. він відрізняє підвищення точності 2,7% в порівнянні з традиційним алгоритмом Soundex

Алгоритм полягає у наступному:

1. Перетворюємо перші літери слова: MAC → MCC, K. M. → N, K → C, pH, PF → FF, SCH → SSS;
2. Перетворюємо останні букви слова : Є. Є. → Y, IE → Y, DT, RT, RD, NT, ND → D;

3. Перший символ в ключі є першим символом слова;
4. Перекладаємо інші літери за правилами:

- EV → AF інакше A, E, I, O, U → A;
- Q → G, Z → S, M → N;
- KN → N інакше K → C;
- SCH → SSS, PH → FF;
- H → Якщо попередній або наступний не є голосною, попередні;
- W → Якщо попередні є голосною, A;
- додати потік до ключа, якщо потік не такий, як останній символ ключа;
- коли останній символ рівний літері S то ми видаляємо його;
- якщо попередні символи AY, замінити на Y;
- видалити останній символ;
- додаємо ключ до слова, починаючи з 3 символу (перші символи відкидаємо);
- у випадку коли довжина є більшою за 6 символів, тоді ми обрізаємо слово по перші шість символів. (вони потрібні тільки для справжніх NYSIIS, деякі версії використовують цілий рядок).[9]

2.8 Висновки до розділу

Досліджено методи розпізнавання мовлення за допомогою рекурентних нейронних мереж та прихованих марковських моделей. Детально розглянуто метод розпізнавання мовлення за допомогою довгої короткочасної пам’яті та пошуку найкращого префіксу.

Проведено аналіз фонетичних алгоритмів та принципів їх роботи, для покращення чіткості розпізнавання слів в наборі мовлення.

РОЗДІЛ 3
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ УТІЛІТИ

3.1 Typescript

Для реалізації програмної утиліти було обрано мову Typescript.

TypeScript - мова програмування, яка позиціонується як альтернативна мова для розробки клієнтських аплікацій в веб застосунка..

Фактично Typescript є надбудовою над Javascript, він додає строгу типізацію, більш звичне ООП. Але Typescript не виконується в браузерах, браузери чи серверні утиліти(Node.js) просто не вміють інтерпретувати його синтаксис, тому розробка відбувається на Typescript, а в браузері виконується Javascript отриманий від псевдо-компіляції Typescript в Javascript.[20]

Переваги над JavaScript:

- статична типізація;
- класичне ООП, як в інших традиційних об'єктно-орієнтованих мовах);
- краща масштабованість.

Також в Typescript є ряд недоліків;

- погана підтримка Browser API;
- вимагає додаткових описів властивостей об’єктів;
- виконується не Typescript а Javascript.

Один з основних принципів Typescript це те що весь існуючий код котрий написаний JavaScript є сумісним з TypeScript, це значить що в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені бібліотеки чи утиліти.

Що важливіше є можливість поступового переходу з Javascript на Typescript, тобто можливо пофайлово оновлювати існуючу кодову базу.

3.2 Програмна реалізація

Для використання бібліотеки її потрібно попередньо встановити за допомогою одного з пакетних менеджерів(рис. 3.1)

```
shalenyj@waterproof-pc ~ npm install help4you
```

Рис. 3.1 Приклад встановлення за допомогою пакетного менеджера npm

Після підключення бібліотеки в проект її треба ініціалізувати. Блок схема алгоритму ініціалізації зображена на рисунку 3.3 Приклад ініціалізації зображений на рисунку 3.2. Після успішної ініціалізації утиліти в консолі з’явиться повідомлення про результат(рис. 3.4).

```
await Help4You.initialize({
  lang: "en-GB",
  isContinuous: true,
  isDebugActive: true,
  listen: true,
  useSoundex: true,
  speed: 1.5,
  volume: 0.8
});
```

Рис. 3.2 Приклад ініціалізації

Рис. 3.3 Блок схема ініціалізації

Під час виклику методу initialize ми можемо передати набір конфігурацій, як для розпізнавання голосових команд так і для відтворення звукових відповідей:

- lang вказує на якій мові користувач буде вводити звукові команди, і відповідно на цій ж мові бібліотека буде давати звукові відповіді;
- isContinuous - вказує чи після завершення введення звукової команди ми повинні продовжувати реагувати на наступні команди;
- speed - цей атрибут визначає швидкість мовлення під час голосових відповідей. Це значення є відносним швидкості за замовчуванням для кожного доступного голосу, де 1 - це швидкість за замовчуванням, що підтримується механізмом синтезу мовлення або певним голосом (який повинен відповідати нормальній швидкості мовлення). 2 в два рази швидше, а 0,5 вдвічі повільніше. Важливо зазначити що конкретні голоси можуть додатково обмежувати мінімальну та максимальну швидкості, наприклад, конкретний голос може

насправді говорити не швидше ніж у 3 рази відносно стандартної швидкості, навіть якщо ви вказали значення більше 3. Значення за замовчуванням - 1;

- volume визначає гучність висловлювання. Він може змінюватися від 0 до 1 включно, за замовчуванням 1;
- voice атрибут голос синтезу мовлення, котрий користувач бажає використовувати. Коли створюється об'єкт SpeechSynthesisUtterance, цей атрибут повинен бути ініціалізований з значенням null. Якщо під час виклику методу speak () для цього атрибута встановлено один із об’єктів SpeechSynthesisVoice, повернутих getVoices (), тоді користувацький агент повинен використовувати цей голос. Якщо цей атрибут не встановлений або має значення null під час виклику методу speak (), тоді агент користувача повинен використовувати голос за замовчуванням бравзера користувача. Голос який обрав користувач за замовчуванням повинен підтримувати поточну мову (див. lang);[18]
- pitch атрибут визначає тон мовлення для висловлювання. Він коливається від 0 до 2 включно, де 0 є найнижчим та 2 найвищим. 1 відповідає висоті за замовчуванням механізму синтезу мовлення або конкретному голосу. Системи синтезу мовлення або голоси можуть додатково обмежувати мінімальний та максимальний темпи;
- mode атрибут визначає в якому режимі буде працювати розпізнавання мовлення, доступні значення: “normal”, “remote”. Значення remote означає що розпізнавання буде виконувати додаток самостійно за допомогою функції remoteProcessorHandler.

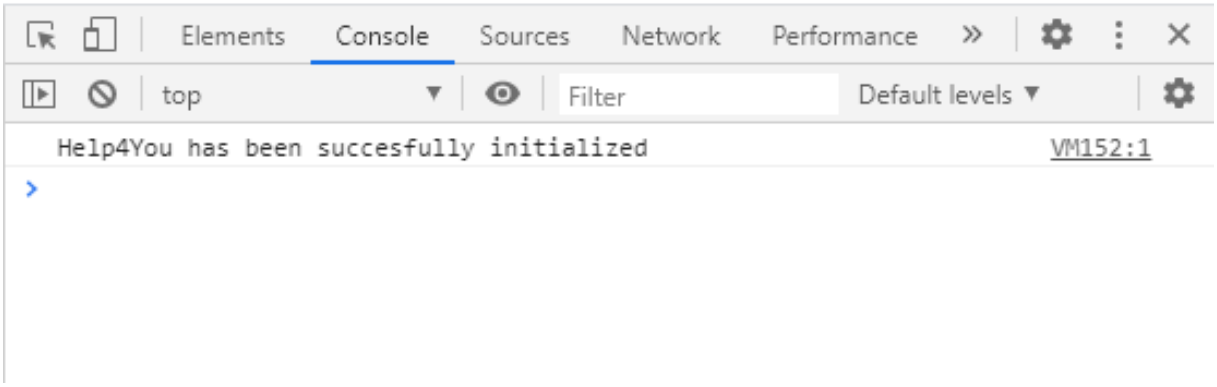


Рис. 3.4 Повідомлення про успішну ініціалізацію

Для зупинки бібліотеки потрібно викликати метод fatality, він призупинить розпізнавання команд і відповідно відтворення мовлення.

Бібліотека використовує власні інтерфейси які є необхідними при створенні бібліотеки за допомогою Typescript. Це дозволить покращити типізацію і тим самим спросить програмний код.

Інтерфейс Command відповідає за представлення команди та її специфіку(рис 3.5)

```
interface Command {
    indexes: Array<any>;
    action: Function;
    description ?: string;
    isSmart ?: Boolean;
}
```

Рис. 3.5 Інтерфейс Command

Властивості інтерфейса означають:

- поле indexes відповідає за масив стрічок які описують які словосполучення будуть викликати функцію яка описана в полі action;

- поле action містить в собі функцію яка буде викликана коли користувач введе потрібну команду;
- поле description відповідає за опис команди, її призначення;
- прапорець isSmart позначає команду як “розумну”, що означає що крім статично описаного тексту команди вона ще може опрацьовувати слова які були провлені але командою були позначеними як надлишкові.

Для додавання нової команди потрібно викликати метод addCommands. Приклад додавання нової команди наведено на рисунку 3.6 , блок схема алгоритму додавання нових команд зображена на рисунку 3.7.

```
const commandHello = {
  indexes:["вітаю","привіт"], // Ці слова будуть опрацьовуватись як виклик функції описаної в полі action
  action(){ // Функція яка буде викликана
      myApp.handleWelcome()
  },
  description: 'Демонстраційна команда'
};

Help4You.addCommands(commandHello); // Додаємо команду до реєстру команд
```

Рис. 3.6 Приклад додавання нової команди

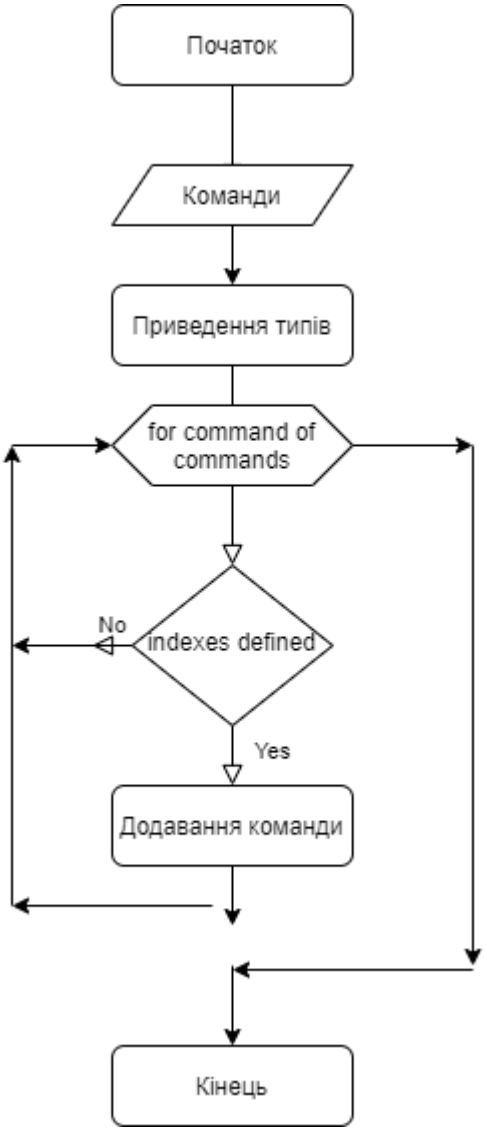


Рис. 3.7 – Блок-схема алгоритму додавання нових команд

Використання “розумних команд” дозволяє команду більш гнучкою, та залежною від контексту. У прикладі на рисунку 3.8 замість того щоб спочатку виконати команду пошуку усіх контактів, після цього відтворити звуковими повідомленнями усіх контактів користувача ми це робимо в рамках одного голосового введення, що покращує зручність для користувача і зменшує навантаження на систему ніж при десятках проміжних операціях.

```
const smartCommand = {
  indexes: ["Чи є у мене друг *","Чи маю я знайомого *","Чи є  * моїм контактом"],
  isSmart: true,
  action(i,wildcard){
    const contacts = ["Іван","Микола","Петро","Віка","Олег"];

    if(contacts.includes(wildcard.trim())){
      Help4You.say(`Так. ${wildcard} є Вашим контактом`)
    }else{
      Help4You.say(`Ні. ${wildcard} не є Вашим контактом`)
    }
  },
  description:"Перевірка чи є у списку контактів імя",
};
```

Рис. 3.8 Приклад Розумної команди

Також є скорочений варіант для опрацювання повідомлень від користувача(рис. 3.9), приклад виконання наведено на рисунку 3.10.

```
Help4You.on(["Hi mate"], input => {
  alert("Вітаємо Вас на сайті");
})
```

Рис. 3.9 Скорочений варіант додавання команди

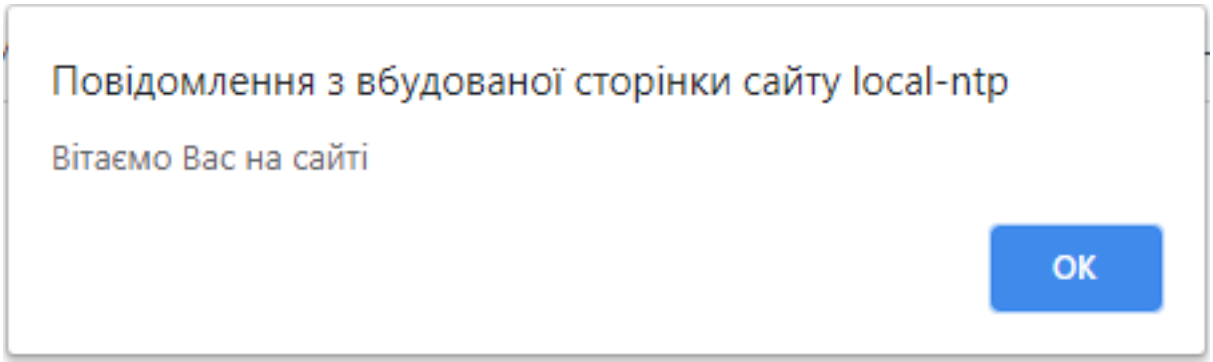


Рис. 3.10 Результат виконання команди

Enums GlobalEvents описує глобальні події які використовує бібліотека, наприклад подія яка повідомляє про те що розпочалося розпізнавання тексту.

Користувач може додати обробники до кожної події, і коли подія відбудеться буде викликана функція передана користувачем.

Доступні події:

- ERROR - відбулась помилка під час розпізнавання мовлення, чи під час відтворення мовлення, аргументом є об’єкт помилки;
- SYNTHESIS_START - подія означає що розпочався синтез звукового повідомлення для його відтворення, аргументом функції є текст який буде відтворений;
- SYNTHESIS_END - подія означає що закінчився синтез звукового повідомлення і воно було відтворене, аргументом є текст який був відтворений;
- RECOGNIZED - завершилось розпізнавання тексту, аргументом є текст який було розпізнано;
- RECOGNITION_START - повідомляє про те що розпочалось розпізнавання тексту;
- COMMAND_MATCHED - подія яка повідомляє про успішне знаходження команди яка відповідає голосовому вводу користувача, аргументом є об’єкт команди яка співпала;
- NOT_COMMAND_MATCHED - повідомляє що утиліта не змогла знайти команду яка б відповідала введеному користувачем тексту, аргументом текст який було розпізнано.

В додатку Б представлено блок схему алгоритму розпізнавання команд від користувача. SpeechRecognition дає змогу додати callbacks на наступні події:

- audiostart відбувається, коли користувацький агент(браузер) починає отримувати звукові сигнали з пристрою вводу;
- soundstart спрацьовує, коли виявлено якийсь звук, можливо мовлення. Цю подію варто опрацьовувати з низькою частотою;
- speechstart спрацьовує коли розпочалася передача сигналу мовлення, який буде використовуватися для розпізнавання мови;

- speechend відбувається коли сигнал, який буде використаний для розпізнавання мови, закінчиться;
- soundend спрацьовує, коли завершився звуковий сигнал, можливо мовлення;
- audioend відбувається, коли користувацький агент(браузер) завершив передавати звукові сигнали з пристрою вводу;
- result спрацьовує, коли розпізнавання мови успішно завершилося і є результат;
- nomatch здійснюється коли розпізнавання мови не змогло розпізнати мову в звуковому сигналі;
- start спрацьовує коли розпочалось прослуховування на мовлення;
- end спрацьовує коли закінчилось прослуховування на мовлення;
- error відбувається у випадку виникнення помилки під час розпізнавання мовлення.[18]

В додатку В представлено блок схему алгоритму пошуку команди яку намагався ввести користувач згідно тексту який було розпізнано.

Команди в яких прапорець isSmart рівний true мають вищий пріоритет ніж інші, тому пошук спочатку відбувається серед них.(рис 3.11).

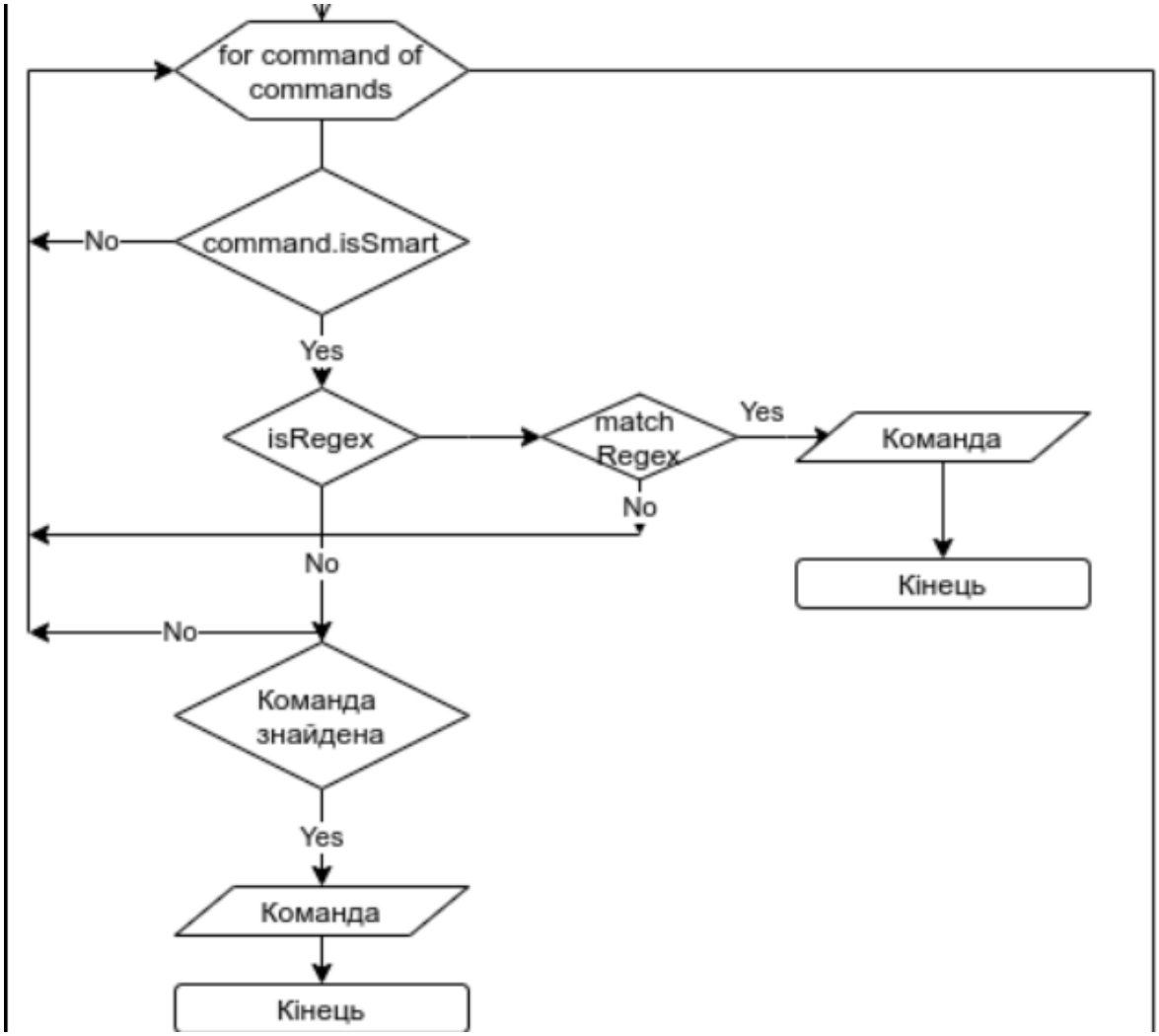


Рис. 3.11 - Пошук “розумної команди”

У випадку коли в масиві indexes обєкта команди вказано регулярні вирази, перевіряємо за допомогою регулярного виразу чи текст який було отримано після розпізнавання відноситься до цієї команди. Коли передаємо значення типу String співпадіння перевіряється за допомогою методів об'єкту String.

Будь яке співпадіння викликає вихід з функції з поверненням знайденої команди.

Якщо співпадіння не було знайдено алгоритм розпочинає пошук по інших командах(рис. 3.12)

Рис. 3.12 Пошук співпадіння серед інших команд

Якщо після цього не вдалось знайти команду наступним кроком ми використовуємо Soundex алгоритм одночасно і для indexes поля обєкта команди з отриманим текстом. Реалізації Soundex алгоритму зображена на рисунку 3.13.

```
soundex(s: string) {
    const a = s.toLowerCase().split('');
    const f = a.shift();
    let r = '';
    const codes = {a:"",e:"",i:"",o:"",u:"",b:1,f:1,p:1,v:1,c:2,g:2,j:2,k:2,q:2,s:2,x:2,z:2,d:3,t:3,l:4,m:5,n:5,r:6};

    r = f + a
        .map((v,) => codes[v])
        .filter((v, i, a) => ((i === 0) ? v !== codes[f] : v !== a[i - 1]))
        .join('');

    return (r + '000').slice(0, 4).toUpperCase();
}
```

Рис. 3.13 Програмна реалізація Soundex алгоритму

3.3 Порівняння методів розпізнавання мовлення

Переваги використання прихованих марковських моделей для розпізнавання мовлення:

- проблема розпізнавання вирішується аналітичним методом;
- розпізнавання слів котрі не несуть корисного навантаження, тобто слів які є виключно набором букв, без змісту;
- є простішими в процесі навчання та в методах реалізації.

До недоліками такої моделі можна віднести відносно низьку точність та не найкращі показники роботи в умовах шуму.[11]

Система	Міра помилки, %
Контекстно незалежна ПММ	38.85
Контестно залежна ПММ	35.21
BLSTM/HMM	33.84
BLSTM/HMM із зваженими помилками	31.57
HTK (метод пошуку кращого шляху)	31.47
HTK (метод пошуку кращого префіксу)	30.51

Рис. 3.14 Порівняння існуючих методів

Переваги використання рекурентний нейронних мереж для розпізнавання мовлення:

- вища швидкість роботи;
- точність розпізнавання більше;
- краще працюють в умовах підвищеного шуму;
- добре працює в умовах неточності та незавершеності промовлених слів.

Недоліками є

- вимагає великих обчислювальних потужностей;

- необхідна велика кількість прикладів для навчання;
- багато часу для навчання.

На рисунках 3.14 і 3.15 зображено порівняння існуючих методів розпізнавання мовлення

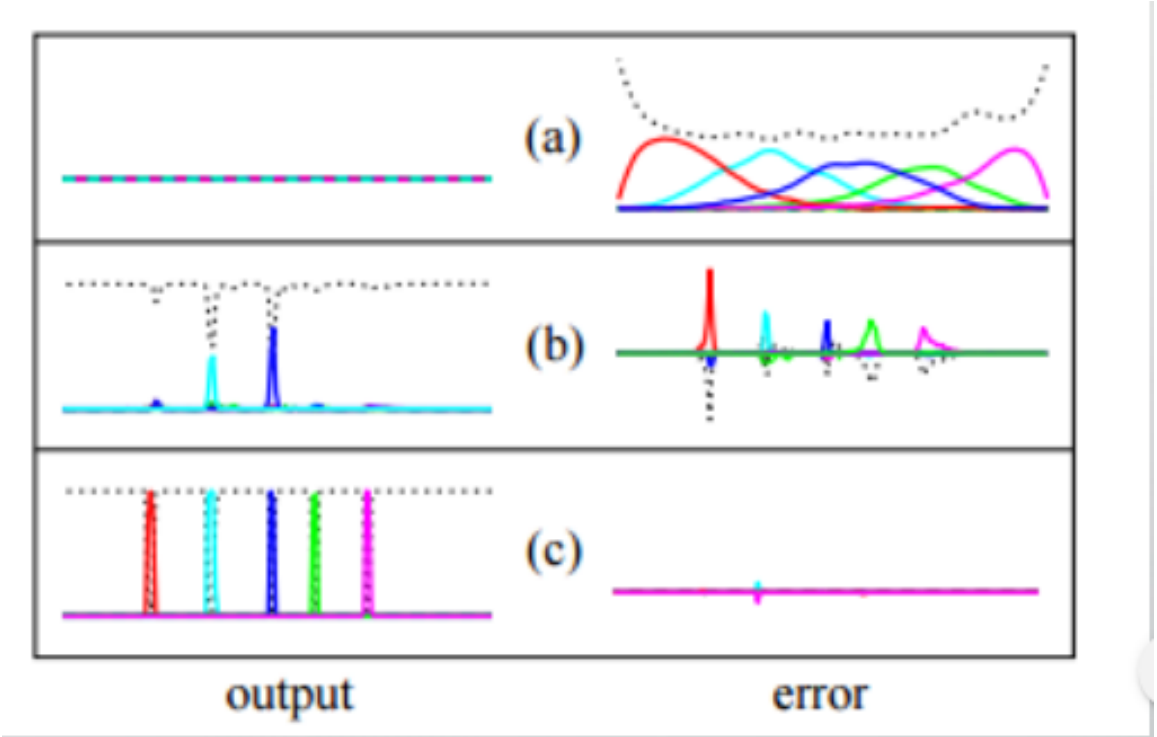


Рис. 3.15 Порівняння існуючих методів

3.4 Порівняння фонетичних алгоритмів

У пунктах 2.6 - 2.7 другого розділу описані найбільш поширені та продуктивні алгоритми Soundex та NYSIIS(подвійний Soundex) алгоритми.

На рисунку 3.16 наведено порівняльну діаграму фонетичних алгоритмів за точністю визначення.

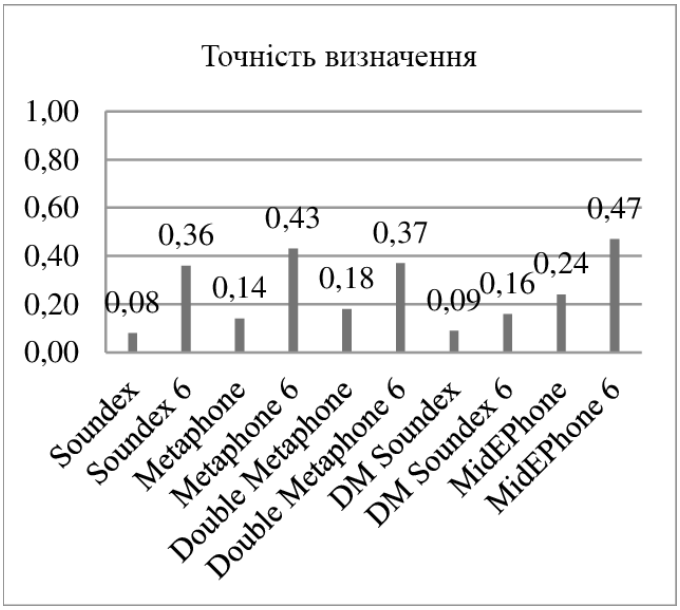


Рис. 3.16 Порівняння фонетичних алгоритмів за точністю

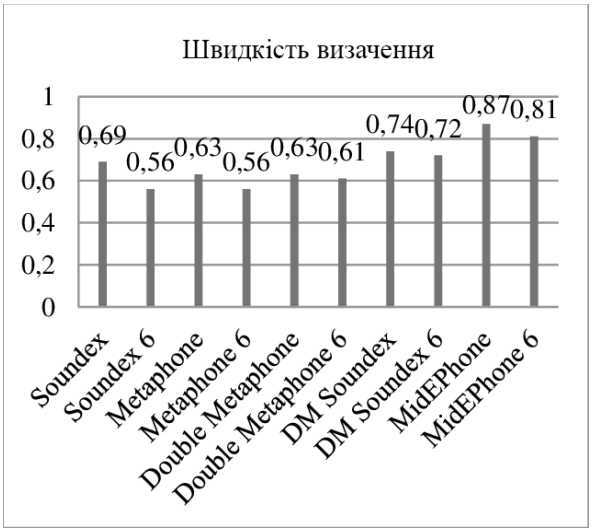


Рис. 3.17 Порівняння фонетичних алгоритмів за швидкість визначення

Згідно порівняльних діаграм можна зробити висновок що використовуючи швидший алгоритм ми програємо в якості і навпаки, якщо виграємо в точності то програємо в швидкості.

3.5 Висновки до розділу

Спроектовано утиліту(бібліотеку) згідно паттерну Фасад, тобто буде надавати простіше API для користування, реалізовано утиліти яка дозволяє реалізувати голосову взаємодію з веб-орієнтованими середовищами, що дає змогу покращити доступність для людей з особливими потребами.

Проаналізовано алгоритми роботи утиліти, описано функціонал, та наведено приклади використання також проведено порівняння методів розпізнавання мовлення, та ефективність фонетичних алгоритмів.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Метою кваліфікаційної роботи магістра є реалізація методу голосової взаємодії з веб-орієнтованими середовищами для людей з особливими потребами. Зважаючи на те, що проведення робіт з проектування та програмної реалізації утиліти передбачає використання комп’ютерної техніки, а саме ПК та периферійних пристроїв, то є обов’язковим дотримання всіх вимог техніки безпеки і охорони праці.

Для ефективної і безпечної роботи колективу працівників з розробки ПЗ комп’ютерних систем, в тому числі і фахівців з дослідження методів та інструментальних засобів, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці [22].

Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями».

Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог;

– переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків

Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

– Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.[22]

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014.

Організація робочого місця фахівця із дослідження повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Лінія електромережі для живлення комп’ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

У випадку коли в приміщенні одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Порядок роз’єднання при відключенні має бути зворотним.

Не допускається підключати комп’ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В.

Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

У результаті аналізу вимог щодо охорони праці користувачів комп’ютерів, визначено особливості організації робочих місць, вимог з природного та штучного освітлення, електробезпеки, для ефективної і безпечної роботи фахівців.

4.2 Інженерний захист персоналу об'єкту та населення. Правила застосування

Інженерний захист – це комплекс організаційних і інженерно-технічних заходів, що проводяться завчасно а також в оперативному порядку і спрямованих на запобігання або максимальне зниження втрат при викикненні надзвичайних ситуацій шляхом забезпечення укриття і життєдіяльності персоналу об'єкту та населення в захисних спорудах, запобігання, усунення або зниження до допустимого рівня негативного впливу вражаючих факторів, стихійних лих, аварій, природних і техногенних катастроф. Заходи інженерного захисту регламентуються низкою нормативних документів, основним з яких є Кодекс цивільного захисту України .[22]

Оцінка інженерного захисту працівників об’єкта полягає у визначенні показників, які характеризують здатність інженерних споруд забезпечити надійний захист людей, що можливо при виконанні наступних умов:

−загальна місткість захисних споруд дозволяє вкрити найбільшу працюючу зміну;

−захисні властивості захисних споруд відповідають потрібним (забезпечують захист людей від надлишкового тиску ударної хвилі та іонізуючих випромінювань);

−системи життєзабезпечення захисних споруд забезпечують життєдіяльність персоналу протягом встановленого терміну безперервного перебування їх в захисних спорудах;

−розміщення захисних споруд відносно місць роботи дозволяє людям укритися за сигналом ЦО у встановлені строки.

На основі оцінки інженерного захисту визначаються заходи, спрямовані на підвищення надійності заходу персоналу об’єкта від вражаючих факторів, а відповідно, і на підвищення стійкості функціонування об’єкта в умовах виникнення НС.[23]

Оцінка захисних споруд за місткістю, місткість захисних споруд об'єкта визначається у відповідності з нормами об'ємно-планувальних рішень. За кількістю місць оцінюється можливість укриття працівників у найбільшій працюючій зміні.

Послідовність оцінки:

1. Виявляється наявність основних і допоміжних приміщень і відповідність їх розмірів нормам об'ємно-планувальних рішень.

2. Розраховується кількість місць (М) з урахуванням норм на одну людину: S_1=0,5 м^2/люд. за наявності в захисній споруді двоярусних нар, S_1=0,4 м^2/люд. за наявності трьохярусних нар (для приміщень висотою 2,9 м і більше):[23]

$$M=S_п/S_1 \qquad (4.1)$$

де $S_п$ – площа приміщень для укриття, м^{2,}

S_1 – площа приміщень на 1 людину, м^2.

3. Перевіряється відповідність об'єму приміщень в зоні герметизації встановленій нормі на одну людину (не менше 1,5 м^3/люд.).

Для цього розраховується об'єм всіх приміщень в зоні герметизації $V_о$ (крім приміщень дизельних електростанцій, тамбурів і розширювальних камер):

$$V_о = S_о \cdot h \qquad (4.2)$$

де h – висота приміщень, м;

$S_о$ – загальна площа всіх приміщень в зоні герметизації, м^2.

Далі визначається об'єм приміщення, що припадає на одну людину:

$$V_1=V_0/M \qquad (4.3)$$

Якщо $V_1$1,5 м^3/люд., то розрахункова місткість М приймається за фактичну місткість захисної споруди.

4. Визначається показник, який характеризує захисні споруди за місткістю (можливість укриття працюючого персоналу), - коефіцієнт місткості:

$$К_м = М/Н \qquad (4.4)$$

де М – загальна кількість місць в захисній споруді;

Н – чисельність найбільшої працюючої зміни, ч.

За результатами розрахунків робиться висновок про можливість укриття працівників об’єкта. Перевіряється наявність у сховищі необхідної кількості нар і відповідність їх місткості сховища.

Оцінка систем життєзабезпечення захисних споруд включає в себе оцінку

системи повітропостачання здійснюється в наступній послідовності:

1. Визначаються тип, склад і параметри системи і визначається кількість повітря, що постачається системою за годину в двох режимах: в режимі I – чистої вентиляції і в режимі II – фільтровентиляції.

Кількість зовнішнього повітря, що подається у сховище, приймається:

– за режимом I – 8,10, 11 і 13 м^3/год на одну людину відповідно до температури зовнішнього повітря до 20^{o}С (I кліматична зона); 20-25^{o}С (II зона), 25-30^{o}С (III зона) і більше 30^{o}С (IV зона);

– за режимом II – 2 м^3/год на одну людину, 5 м^3/год на одного працівника на пункті правління і 10 м^3/год на одного працюючого на електроручному вентиляторі (ЕРВ).

У сховищах, що розташовані у III і IV кліматичних зонах, для режиму II необхідно передбачити охолоджуючі пристрої або збільшення кількості повітря, що подається, до 10 м^3/год на людину.[23]

Як джерело для охолодження передбачається вода, що зберігається у заглиблених резервуарах чи отримана з водозабірних свердловин.

Інженерний захист є обов'язковою умовою для попередження чи максимального зниження втрат та матеріальних збитків при виникненні надзвичайних ситуацій. Раціонально сплановані підготовлені та реалізовані заходи інженерного захисту забезпечують зниження можливих людських втрат та матеріальних збитків, створюють умови для успішного проведення аварійно-рятувальних та інших невідкладних робіт.

4.3. Висновки до розділу

В цьому розділі проаналізовано важливі питання охорони праці та безпеки в надзвичайних ситуаціях, висвітлено питання інженерного захисту персоналу об’єкта та населення.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено аналіз актуальності проблеми доступності веб-орієнтованих середовищ для людей з особливими потребами та досліджено існуючі рішення.

Існуючі рішення ігнорують метод голосової взаємодії з середовищем через складність реалізації оскільки цей напрямок є малодослідженим, та є складним у використанні через можливість виникнення великої кількості помилок які потрібно опрацьовувати.

Предметом дослідження було обрано дослідження можливого використання технології WebSpeech API для реалізації керування веб-сайтом за допомогою голосового управління.

У другому розділі розглянуто математичне забезпечення систем розпізнавання мовлення за допомогою прихованих марковських моделей та рекурентних нейронних мереж. Детально описано алгоритм розпізнавання мови за допомогою ДКЧП архітектури та використання алгоритму пошуку найкращого префіксу. Також досліджено фонетичні алгоритми, переваги та недоліки їх використання для покращення результатів розпізнавання мовлення.

В третьому розділі було спроектовано та програмно реалізовано програмну утиліту паттерну “Фасад” для спрощення реалізації методу голосової взаємодії, детально розглянуто алгоритми роботи WebSpeech API та програмної утиліти.

Проведено порівняння фонетичних методів та методів розпізнавання мовлення.

Недоліком методу голосової взаємодії є проблема сумісності з застарілими браузерами, оскільки вони не підтримують WebSpeech API і це робить неможливим отримання голосових команд від користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лупенко С.А. Циклічні функції та їх класифікація в задачах моделювання циклічних сигналів та коливних систем. Вимірювальна та обчислювальна техніка в технологічних процесах. Випуск 1, 2005. 177–185с.

2. Sepp Hochreiter, Jürgen Schmidhuber Long short-term. Packt Publishing, 1997. 1735–1780с.

3. Luke McGrath: How to Meet the Web Content Accessibility Guidelines. Lulu, 2016. 420с.

4. Лупенко С.А. Приймак Умовно періодичні випадкові процеси із змінним періодом. Вісник ТДТУ Том 10 Випуск 2, 2005. 143–152с.

5. Sarah Horton A Web for Everyone: Designing Accessible User Experiences Paperback, 2014. 280с.

6. Richard Rutter Web Accessibility: Web Standards and Regulatory Compliance . Apress, 2008. 190с.

7. Goller C., Küchler A. Learning task-dependent distributed representations by backpropagation through structure. Neural Networks, 1996., IEEE 374-394с;

8. Лупенко С.А. Циклічне функціональне відношення як основа математичного формалізму теорії моделювання та аналізу циклічних сигналів. Вісник ТДТУ Том 12 Випуск 3, 2007. 183–195с.

9. Lawrence Rabiner A tutorial on Hidden Markov Models and selected applications in speech recognition. UCSB, 1989.257–286с.

10. Satish L, Gururaj BI Use of hidden Markov models for partial discharge pattern classification. Deli, 2003. 98-130с.

11. Xiangang Li, Xihong Wu Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. Seul. 2015 298-302с.

12. Лупенко С.А Теоретичні основи моделювання та опрацювання циклічних сигналів в інформаційних системах. Магнолія-2006, 344с.

13. Лупенко С.А. Циклічне функціональне відношення як основа математичного формалізму теорії моделювання та аналізу циклічних сигналів. Вісник ТДТУ Том 12 Випуск 3, 2007. 183–195с.

14. Лупенко С.А. Циклічні функції та їх класифікація в задачах моделювання циклічних сигналів та коливних систем. Вимірювальна та обчислювальна техніка в технологічних процесах. Випуск 1, 2005. 177–185с.

15. Лупенко С.А. Оператор перетворення шкали в задачах моделювання та аналізу циклічних сигналів. Вісник ТДТУ Том 12 Випуск 4, 2007. 141–152с.

16. W3C. URL: https://www.w3.org (дата звертання: 07.12.2020).

17. WCAG. URL: https://www.w3.org/WAI/standards-guidelines/wcag/ org (дата звертання: 07.12.2020).

18. WebSpeech API Docs. URL: https://wicg.github.io/speech-api (дата звертання: 07.12.2020).

19. Can I use. URL: https://caniuse.com/ (дата звертання: 07.12.2020).

20. Typescipt for Javascript Programmers. https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html (дата звертання: 07.12.2020).

21. Кивацький І.М. технології голосової взаємодії з веб-орієнтованим віртуальним середовищем. Інформаційні моделі, системи та технології: Праці VIII наук.-техн. конф. (Тернопіль, 09-10 грудня 2020 р.) Тернопіль, 2020. – С. 106.

22. Зеркалов Д.В. Охорона праці в галузі: Загальні вимоги. Навчальний посібник. К.: Основа. 2011. 551 с.

23. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навч. посібник. – 2011. – 215 с.

ДОДАТОК А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Мариборский университет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедиміната (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих ученних та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів.
Актуальні задачі сучасних технологій – Тернопіль 25-26 листопада 2020.

УДК 004.031.6
С.А. Лупенко, док. техн. наук, І.М. Кіяцький
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПРОБЛЕМА ДОСТУПНОСТІ ІНТЕРНЕТУ ДЛЯ ЛЮДЕЙ З ОСОБЛИВИМИ ПОТРЕБАМИ

S.A. Lupenko Dr, I.M. Kyvatskyi

THE PROBLEM OF INTERNET ACCESSIBILITY FOR PEOPLE WITH SPECIAL NEEDS

Веб-доступність передбачає проектування і розробку веб-сайтів, додатків і технологічних рішень з урахуванням можливості їх використання людьми з особливими потребами. Іншими словами, такі користувачі можуть самостійно отримувати, шукати інформацію і спілкуватися за допомогою Інтернету.

Доступність Інтернету залежить від декількох взаємодіючих між собою компонентів, зокрема веб-технологій, веб-браузерів, інструментів розробки та веб-сайтів.

Багато компонент веб-доступності досить прості для розуміння і використання. Деякі рішення, в той же час, є комплексними і вимагають додаткових знань для адаптації.

Використання рекомендацій WCAG покращують доступність веб-сайту, але не в такій мірі, яку потребують люди з особливими потребами.

Використання технології Web Speech API дає змогу реалізації голосової взаємодії з сайтом, тобто користувач замість використання клавіатури може взаємодіяти з сайтом за допомогою голосових команд, в свою чергу сайт окрім візуального виведення інформації додає голосовий вивід інформації як альтернативний для людей з відхиленнями зору.

Найбільш ефективна і дієва практика передбачає забезпечення веб-доступності з самого початку проектування веб-сайту, оскільки це забезпечить менші витрати часу у порівнянні з додаванням доступності на вже існуючий веб-сайт.

Використання технології голосової взаємодії з веб сайтом дає змогу покращити суттєво не тільки доступність сайту для людей з особливими потребами, але і зробить користування сайтом зручнішим не тільки для людей з певними вадами але і для будь якого користувача.

Література

1. Heydon Pickering. Inclusive Design Patterns — London : 2017. — 543 p.
2. Laura Kalbag. Accessibility For Everyone – Paris : A Book Apart 2017 – 231 p.
3. Hello Geri. Color Accessibility Workflows – Chicago : A Book Apart 2017 – 132 p.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

МАТЕРІАЛИ

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004.031.6
С.А. Лупенко, док. техн. наук, І.М. Кивацький
(Тернопільський національний технічний університет імені Івана Пулюя)

ТЕХНОЛОГІЇ ГОЛОСОВОЇ ВЗАЄМОДІЇ З ВЕБ-ОРІЄНТОВАНИМ ВІРТУАЛЬНИМ СЕРЕДОВИЩЕМ

UDC 004.031.6
S.A. Lupenko, Dr, I.M. Kyvatskyi

VOICE INTERACTION TECHNOLOGIES WITH A WEB-ORIENTED VIRTUAL ENVIRONMENT

Голосова взаємодія з веб-орієнтованими середовищами призначена для введення команд в середовище за допомогою голосу, і середовище дає змогу отримувати інформацію не тільки в текстовому представленні, але і звуковому. Можливість реалізації голосової взаємодії в веб-орієнтованих віртуальних середовищах (бравзерах) дає технологія WebSpeecApi.

Доступ до розпізнавання мовлення забезпечується через SpeechRecognition інтерфейс, котрий в свою чергу забезпечує можливість розпізнавати текст з вхідного аудіо потоку і надавати відповіді. Скориставшись конструктором інтерфейсу є змога створити новий SpeechRecognition-об'єкт, у якого є ряд подій для виявлення початку мовлення через мікрофон пристрою. SpeechGrammar-інтерфейс надає контейнер для певного набору граматик, котрі будуть використовувати відповідний програмний додаток. Граматика визначається за допомогою JSpeech Grammar Format (JSGF).

Використання синтезу мови здійснюється за допомогою SpeechSynthesis інтерфейсу, компонент text-to-speech дозволяє веб-додаткам прочитати свій текстовий контент. У SpeechSynthesisVoice-об'єктах є різні типи голосу, і різним частинам тексту можна призначати SpeechSynthesisUtterance-об'єкти.

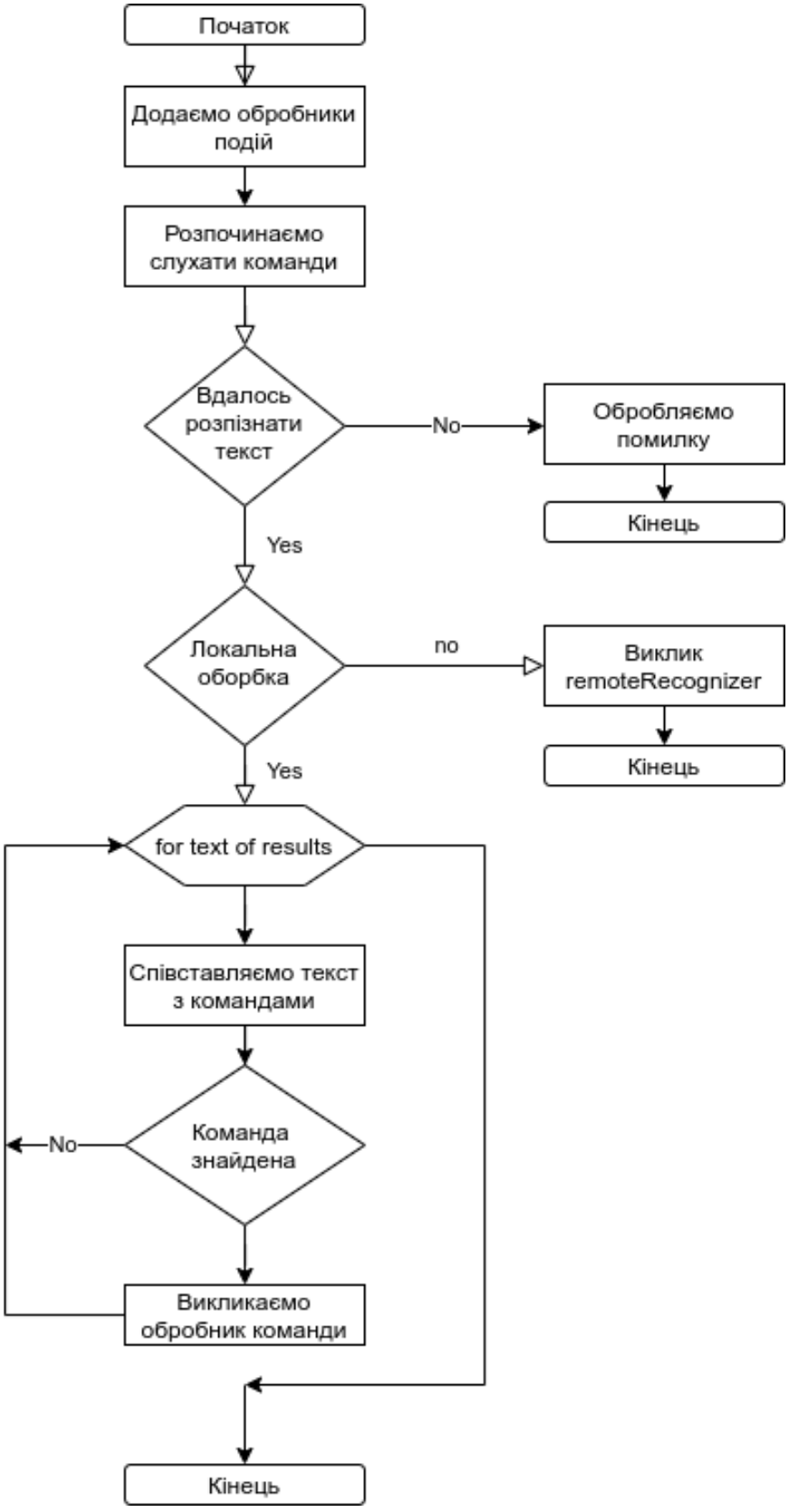
Недоліком методу з веб-орієнтованими середовищами є проблема сумісності з старими браузерами. Це зумовлено тим, що API Інтернет-бравзерів не є стандартизовано, і такі браузери як Chrome та Firefox підтримують більш сучасні технології, які довзоляють зробити веб-сайт більш доступним.

Література.

1. Heydon Pickering. Inclusive Design Patterns – London : 2017. – 543 p.
2. Laura Kalbag. Accessibility For Everyone – Paris : A Book Apart 2017 – 231 p.
3. Hello Geri. Color Accessibility Workflows – Chicago : A Book Apart 2017 – 132 p.

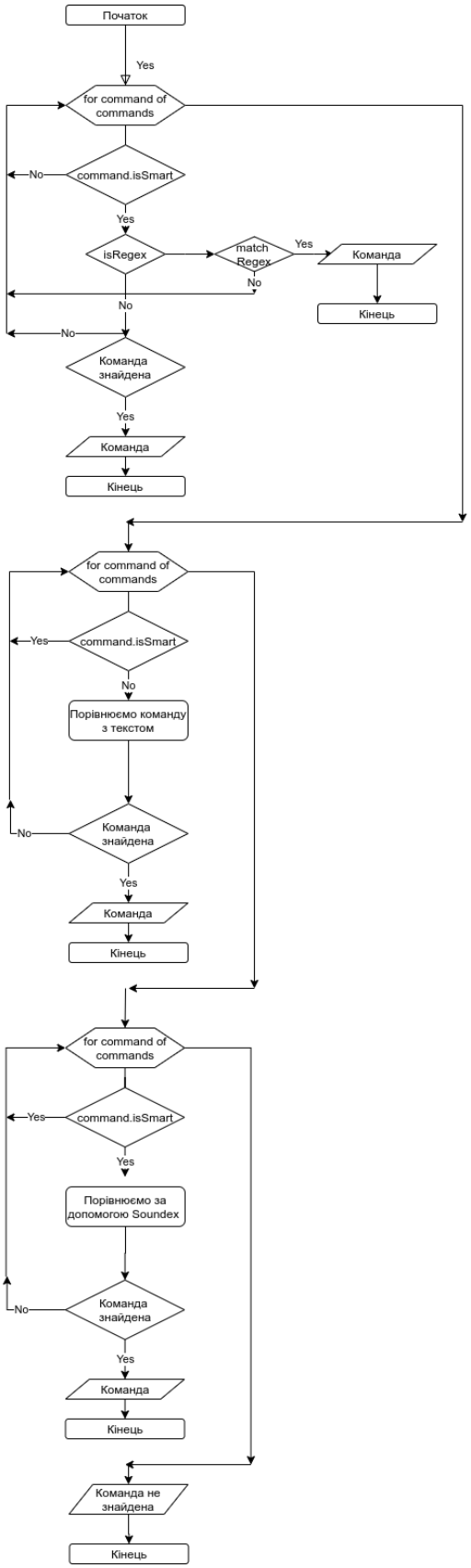
ДОДАТОК Б

Блок схема розпізнавання команд



ДОДАТОК В

Блок схема співставлення команд



ДОДАТОК Г

Лістинг коду

```
interface Command {
  indexes: Array<any>;
  action: Function;
  description ?: string;
  isSmart ?: Boolean;
}

interface Flags {
  restartRecognition ?: Boolean
}


interface SayCallbacksObject {
  lang?: string;
  onStart?: Function;
  onEnd?: Function;
}

interface Settings {
  lang: string;
  isRecognizing?: boolean;
  isContinuous?: boolean;
  /*
    A float representing the rate value. It can range between 0.1 (lowest) and 10 (highest),
    with 1 being the default pitch for the current platform or voice, which should correspond to a normal speaking rate.
    Other values act as a percentage relative to this, so for example 2 is twice as fast, 0.5 is half as fast, etc
  */
  speed?: number;
  // A float that represents the volume value, between 0 (lowest) and 1 (highest)
  volume?: number;
  listen: boolean;
  mode?: string;
  isDebugActive?: boolean;
  helpers?: {
    fatalityPromiseCallback?: any;
    redirectRecognizedTextOutput?: any;
```

```
remoteProcessorHandler?: any;};
executionKeyword?: string;
obeyKeyword?: string;
isSpeaking?: boolean;
obeying?: boolean;
useSoundex?: boolean;
name?: string;
}

interface PromptOptions {
question: string;
options: any;
beforePrompt?: Function;
onStartPrompt?: Function;
onEndPrompt?: Function;
onMatch?: Function;
isSmart?: Boolean;
}
interface MatchedCommand {
index: number;
instruction: Command;
wildcard?: any;
}
interface Device {
isMobile: Boolean;
}

declare global {
  interface Window {
```

```
    webkitSpeechRecognition: object;
  }
}
enum EVENTS {
  ERROR = "ERROR",
    SYNTHESIS_START = "SYNTHESIS_START",
    SYNTHESIS_END = "SYNTHESIS_END",
    RECOGNIZED = "RECOGNIZED",
    RECOGNITION_START = "RECOGNITION_START",
    RECOGNITION_END = "RECOGNITION_END",
    MATCHED = "MATCHED",
    NOT_MATCHED = "NOT_MATCHED"
}
export default class Help4You {
  // Stores an object with available voices for
WebkitSpeechSynthesis
  // Stores the webkitSpeechRecognition instance used by Help4You.
  private recognitionInstance: any;
  // An array that stores all commands for instance
  private commands: Array < Command > = [];
  /*
    Due to issue with garbage collector the
SpeechSynthesisUtterance object
    onEnd event doesn't get triggered sometimes. So we need to
keep the reference of the
    object inside this array to work with them.
  */
  private talksHolder: Array < any > = [];
  private flags: Flags = {
    restartRecognition: false
  };
  // default settings
  private settings: Settings = {
    lang: 'en-GB',
    isRecognizing: false,
    isContinuous: false,
    speed: 1,
    volume: 1,
    listen: false,
    mode: "normal",
    isDebugActive: false,
    helpers: {
      redirectRecognizedTextOutput: null,
      remoteProcessorHandler: null,
      fatalityPromiseCallback: null
    },
    executionKeyword: null,
    obeyKeyword: null,
    isSpeaking: false,
    obeying: true,
    useSoundex: false,
```

```
    name: null
  };
  private device: Device = {
    isMobile: false,
  };
  // Triggered at the declaration of
  constructor() {
    // the execution of speechSynthesis.getVoices will return at
the first time an empty array.
    if (window.speechSynthesis) {
      speechSynthesis.getVoices();
    } else {
      console.error("Help4You.js can't speak without the Speech
Synthesis API.");
    }
    if (window.webkitSpeechRecognition) {
      this.recognitionInstance = new( < any >
window).webkitSpeechRecognition();
    } else {
      console.error("Help4You.js can't recognize voice without the
Speech Recognition API.");
    }
    if (navigator.userAgent.match(/Android/i) ||
navigator.userAgent.match(/webOS/i) ||
navigator.userAgent.match(/iPhone/i) ||
navigator.userAgent.match(/iPad/i) ||
navigator.userAgent.match(/iPod/i) ||
navigator.userAgent.match(/BlackBerry/i) ||
navigator.userAgent.match(/Windows Phone/i)) {
      this.device.isMobile = true;
    }
  }
  //add new commands
  addCommands(commands: Command | Array < Command > ): void {
    const commandsAsArrays = Array.isArray(commands) ? commands :
[commands];
    for (const command of commandsAsArrays) {
      if (Array.isArray(command.indexes)) {
        this.commands.push(command)
      } else {
        console.error("The given command doesn't provide any index
to execute.");
      }
    }
  };
  clearTalksHolder(): void {
    this.talksHolder = [];
  };
  // display external logs, turn off for prod
  debug(message: string, type: string = '') {
    if (this.settings.isDebugActive === true) {
```

```
      console.log(`${type}: ${message}`)
    }
  }
  // reset commands
  emptyCommands() {
    this.commands = [];
  }
  //Returns an object with data of the matched element
  execute(voz): MatchedCommand {
    let commandFromRecognize = voz.toLowerCase();
    if (!commandFromRecognize) {
      console.warn("Internal error: Execution of empty command");
      return;
    }
    // If initialized with a name, verify that the string begins
with it
    if (this.settings.name) {
      if (commandFromRecognize.indexOf(this.settings.name) != 0) {
        this.debug(`Help4You requires with a name
"${this.settings.name}" but the name wasn't spoken.`, "warn");
        return;
      }
      // Remove name from voice command
      commandFromRecognize =
commandFromRecognize.slice(this.settings.name.length);
    }
    this.debug(">> " + commandFromRecognize);
    for (const instruction of this.commands) {
      const options = instruction.indexes;
      let result = -1;
      let wildy = "";
      for (const [index, option] of options) {
        if (!instruction.isSmart) {
          continue; //Jump if is not smart command
        }
        // Process RegExp
        if (option instanceof RegExp) {
          // If RegExp matches
          if (option.test(commandFromRecognize)) {
            this.debug(">> REGEX " + option.toString() + " MATCHED
AGAINST " + commandFromRecognize + " WITH INDEX " + index + " IN
COMMAND ", "info");
            result = index;
            break;
          }
          // just wildcards
        } else {
          if (option.indexOf("*") !== -1) {
            const group = option.split("*");
            if (group.length > 2) {
```

```
              console.warn("Help4You found a smart command with "
+ (group.length - 1) + " wildcards. Help4You only support 1
wildcard for each command. Sorry");
              continue;
            }
            const [before, after] = group;
            if ((after === "") || (after === " ")) {
              if ((commandFromRecognize.indexOf(before) !== -1) ||
((commandFromRecognize).indexOf(before.toLowerCase()) !== -1)) {
                wildy = commandFromRecognize.replace(before, '');
                wildy =
(wildy.toLowerCase()).replace(before.toLowerCase(), '');
                result = index
                break
              }
            } else {
              if ((commandFromRecognize.indexOf(before) !== -1) ||
((commandFromRecognize).indexOf(before.toLowerCase()) !== -1)) {
                wildy = commandFromRecognize.replace(before,
'').replace(after, '');
                wildy =
(wildy.toLowerCase()).replace(before.toLowerCase(),
'').replace(after.toLowerCase(), '');
                wildy =
(wildy.toLowerCase()).replace(after.toLowerCase(), '');
                result = index
                break
              }
            }
          } else {
            console.warn("Founded command marked as SMART but have
no wildcard in the indexes, remove the SMART for prevent extensive
memory consuming or add the wildcard *");
          }
        }
      }
      if (result >= 0) {
        this.triggerEvent(EVENTS.MATCHED);
        const response: MatchedCommand = {
          index: result,
          instruction: instruction,
          wildcard: {
            item: wildy,
            full: commandFromRecognize
          }
        };
        return response;
      }
    }
    for (const instruction of this.commands) {
      let options = instruction.indexes;
```

```
      let result = -1;
      /**
       * Execution of match with identical commands
       */
      for (const [index, option] of options) {
        if (instruction.isSmart) {
          continue; //Jump wildcard commands
        }
        if (commandFromRecognize === option.toLowerCase()) {
          this.debug(">> MATCHED FULL EXACT OPTION " + option + "
AGAINST " + commandFromRecognize + " WITH INDEX " + index + " IN
COMMAND ", "info");
          result = index
          break;
        }
      }
      if (result >= 0) {
        this.triggerEvent(EVENTS.MATCHED);
        const response: MatchedCommand = {
          index: result,
          instruction
        };
        return response;
      }
    }
    for (const instruction of this.commands) {
      let options = instruction.indexes;
      let result = -1;
      /**
       * Execution of match with identical commands
       */
      for (const [index, option] of options) {
        if (instruction.isSmart) {
          continue; //Jump wildcard commands
        }
        if ((commandFromRecognize.indexOf(option) >= 0)) {
          this.debug(">> MATCHED INDEX EXACT OPTION " + option + "
AGAINST " + commandFromRecognize + " WITH INDEX " + index + " IN
COMMAND ", "info");
          result = index;
          break;
        } else if
(((commandFromRecognize).indexOf(option.toLowerCase()) >= 0)) {
          this.debug(">> MATCHED INDEX OPTION CHANGING ALL TO
LOWERCASE " + option + " AGAINST " + commandFromRecognize + " WITH
INDEX " + index + " IN COMMAND ", "info");
          result = index;
          break;
        }
      }
      if (result >= 0) {
```

```
        this.triggerEvent(EVENTS.MATCHED);
        const response: MatchedCommand = {
          index: result,
          instruction: instruction
        };
        return response;
      }
    }
    if (this.settings.useSoundex) {
      for (const instruction of this.commands) {
        let options = instruction.indexes;
        let result = -1;
        for (const [index, option] of options) {
          if (instruction.isSmart) {
            continue; //Jump wildcard commands
          }
          if (this.soundex(commandFromRecognize) ==
this.soundex(option)) {
            this.debug(
              `>> Matched Soundex command '${option}' AGAINST
'${commandFromRecognize}' with index ${index}`, "info"
            );
            result = index
            this.triggerEvent(EVENTS.MATCHED);
            const response: MatchedCommand = {
              index: result,
              instruction
            };
            return response;
          }
        }
      }
    }
    this.debug(`Event reached : ${EVENTS.NOT_MATCHED}`);
    this.triggerEvent(EVENTS.NOT_MATCHED);
    return;
  }
  //force end
  fatality() {
    return new Promise((resolve, reject) => {
      this.settings.helpers.fatalityPromiseCallback = resolve;
      this.flags.restartRecognition = false;
      try {
        // If config is continuous mode, deactivate anyway.
        this.recognitionInstance.stop();
      } catch (e) {
        reject(e);
      }
    });
  }
  getAvailableCommands(): Array < Command > {
```

```
    return this.commands;
  }
  // Return voices available in your browser
  getVoices() {
    return window.speechSynthesis.getVoices();
  }
  // Verify if the browser supports speechSynthesis.
  isSpeechSupported(): boolean {
    return Boolean(window.speechSynthesis);
  }
  // Verify if the browser supports webkitSpeechRecognition.
  isRecognizingSupported(): boolean {
    return Boolean(window.webkitSpeechRecognition);
  }
  //Stops the actual and pending messages that Help4You have to
say.
  shutUp() {
    if (window.speechSynthesis) {
      do {
        window.speechSynthesis.cancel();
      } while (window.speechSynthesis.pending === true);
    }
    this.settings.isSpeaking = false;
    this.clearTalksHolder();
  }
  // active settings
  getProperties(): Settings {
    return this.settings;
  }
  // current language
  getLanguage(): string {
    return this.settings.lang;
  }
  hey(resolve: Function, reject: Function) {
    let isAllowed;
    if (this.device.isMobile) {
      this.recognitionInstance.isContinuous = false;
      this.recognitionInstance.interimResults = false;
      this.recognitionInstance.maxAlternatives = 1;
    } else {
      this.recognitionInstance.isContinuous = true;
      this.recognitionInstance.interimResults = true;
    }
    this.recognitionInstance.lang = this.settings.lang;
    this.recognitionInstance.onstart = () => {
      this.debug("Event reached : " + EVENTS.RECOGNITION_START);
      this.triggerEvent(EVENTS.RECOGNITION_START);
      this.settings.isRecognizing = true;
      isAllowed = true;
      resolve();
    };
```

```
    this.recognitionInstance.onerror = (event) => {
      // Reject promise on initialization
      reject(event.error);
      // dispath error (when)
      this.triggerEvent(EVENTS.ERROR, {
        code: event.error
      });
      if (event.error == 'audio-capture') {
        isAllowed = false;
      }
      if (event.error == 'not-allowed') {
        isAllowed = false;
        this.triggerEvent(EVENTS.ERROR, {
          code: "info-denied",
          message: "Help4You needs the permision of the
microphone, is denied"
        });
      }
    };
    this.recognitionInstance.onend = function () {
      if (this.flags.restartRecognition === true) {
        if (isAllowed === true) {
          this.recognitionInstance.start();
          this.debug("Continuous mode enabled, restarting",
"info");
        } else {
          console.error("Verify the microphone ");
        }
        this.triggerEvent(EVENTS.RECOGNITION_END, {
          code: "continuous_mode_enabled",
          message: "OnEnd event reached with continuous mode"
        });
      } else {
        if (this.settings.helpers.fatalityPromiseCallback) {
          this.settings.helpers.fatalityPromiseCallback();
          this.triggerEvent(EVENTS.RECOGNITION_END, {
            code: "continuous_mode_disabled",
            message: "OnEnd event reached without continuous mode"
          });
        }
      }
      this.settings.isRecognizing = false;
    };
    let onResultProcessor;
    if (this.settings.mode == "normal") {
      onResultProcessor = (event) => {
        if (!this.commands.length) {
          this.debug("No commands to process in normal mode.");
          return;
        }
        let cantidadResultados = event.results.length;
```

```
        this.triggerEvent(EVENTS.RECOGNIZED);
        for (let i = event.resultIndex; i < cantidadResultados;
++i) {
          let identificated = event.results[i][0].transcript;
          if (event.results[i].isFinal) {
            let comando: MatchedCommand =
this.execute(identificated.trim());
            if (typeof
(this.settings.helpers.redirectRecognizedTextOutput) ===
"function") {

this.settings.helpers.redirectRecognizedTextOutput(identificated,
true);
            }
            if ((comando) && (this.settings.isRecognizing ==
true)) {
              this.debug("<< Executing Matching Recognition in
normal mode >>", "info");
              this.recognitionInstance.stop();
              this.settings.isRecognizing = false;
              // Execute the command if smart
              if (comando.wildcard) {
                comando.instruction.action(comando.index,
comando.wildcard.item, comando.wildcard.full);
                // Execute a normal command
              } else {
                comando.instruction.action(comando.index);
              }
              break;
            }
          } else {
            // Redirect output when necesary
            if (typeof
(this.settings.helpers.redirectRecognizedTextOutput) ===
"function") {

this.settings.helpers.redirectRecognizedTextOutput(identificated,
false);
            }
            if (typeof (this.settings.executionKeyword) ===
"string") {
              if
(identificated.indexOf(this.settings.executionKeyword) != -1) {
                let comando =
this.execute(identificated.replace(this.settings.executionKeyword,
'').trim());
                if ((comando) && (this.settings.isRecognizing ==
true)) {
                  this.debug("<< Executing command ordered by
ExecutionKeyword >>", 'info');
                  this.recognitionInstance.stop();
```

```
                  this.settings.isRecognizing = false;
                  if (comando.wildcard) {
                    comando.instruction.action(comando.index,
comando.wildcard.item, comando.wildcard.full);
                  } else {
                    comando.instruction.action(comando.index);
                  }
                  break;
                }
              }
            }
            this.debug("Normal mode : " + identificated);
          }
        }
      }
    }
    if (this.settings.mode == "remote") {
      onResultProcessor = (event) => {
        let cantidadResultados = event.results.length;
        this.triggerEvent(EVENTS.RECOGNIZED);
        if (typeof (this.settings.helpers.remoteProcessorHandler)
!== "function") {
          return this.debug("The remoteProcessorService is
undefined.", "warn");
        }
        for (let i = event.resultIndex; i < cantidadResultados;
++i) {
          let identificated = event.results[i][0].transcript;
          this.settings.helpers.remoteProcessorHandler({
            text: identificated,
            isFinal: event.results[i].isFinal
          });
        }
      }
    }
    this.recognitionInstance.onresult = (event) => {
      onResultProcessor(event);
    };
    if (this.settings.isRecognizing) {
      this.recognitionInstance.stop();
      this.debug("Event reached : " + EVENTS.RECOGNITION_END);
      this.triggerEvent(EVENTS.RECOGNITION_END);
    } else {
      try {
        this.recognitionInstance.start();
      } catch (e) {
        this.triggerEvent(EVENTS.ERROR, {
          code: "recognition_overlap",
          message: "A webkitSpeechRecognition instance has been
started while there's already running. Is recommendable to restart
the Browser"
```

```
        });
      }
    }
  }
  initialize(config: Settings): Promise < boolean > {
    Object.assign(this.settings, config)
    if (typeof config !== "object") {
      return Promise.reject("Setting should be an object");
    }
    if (config.hasOwnProperty("continuous")) {
      this.flags.restartRecognition = config.isContinuous;
    }
    if (this.settings.listen === true) {
      return new Promise((resolve, reject) => {
        this.hey(resolve, reject);
      });
    }
    return Promise.resolve(true);
  }
  triggerEvent(name: string, param ? : any) {
    const event = new CustomEvent(name, {
      'detail': param
    });
    document.dispatchEvent(event);
    return event;
  }
  when(event: string, action: Function) {
    document.addEventListener(event, (e) => {
      action(e["detail"]);
    });
  }
  remoteProcessorService(action: Function) {
    this.settings.helpers.remoteProcessorHandler = action;
  }
  //Verify if there's a voice available for a language using its
language code identifier.
  checkLanguage(languageCode: string) {
    return Boolean(this.getVoice(languageCode));
  }
  isSpeaking() {
    return this.settings.isSpeaking;
  }
  isRecognizing() {
    return this.settings.isRecognizing;
  }
  getVoice(languageCode: string) {
    let voiceIdentifiersArray = Help4You.LANGS[languageCode];
    if (!voiceIdentifiersArray) {
      console.warn(`Language ${languageCode} isn't available,
using English`);
      voiceIdentifiersArray = Help4You.LANGS["en-GB"];
```

```
    }
    let voice = null;
    let voices = speechSynthesis.getVoices();
    let voicesLength = voiceIdentifiersArray.length;
    for (let i = 0; i < voicesLength; i++) {
      let foundVoice = voices.filter((voice) => {
        return (
          (voice.name == voiceIdentifiersArray[i]) || (voice.lang
== voiceIdentifiersArray[i])
        );
      })[0];
      if (foundVoice) {
        voice = foundVoice;
        break;
      }
    }
    return voice;
  }
  setDebug(value: boolean): void {
    this.settings.isDebugActive = value;
  }
  soundex(s: string) {
    const a = s.toLowerCase().split('');
    const f = a.shift();
    let r = '';
    const codes = {
      a: "",
      e: "",
      i: "",
      o: "",
      u: "",
      b: 1,
      f: 1,
      p: 1,
      v: 1,
      c: 2,
      g: 2,
      j: 2,
      k: 2,
      q: 2,
      s: 2,
      x: 2,
      z: 2,
      d: 3,
      t: 3,
      l: 4,
      m: 5,
      n: 5,
      r: 6
    };
    r = f + a
```

```
      .map((v, i, a) => {
        return codes[v];
      })
      .filter((v, i, a) => {
        return ((i === 0) ? v !== codes[f] : v !== a[i - 1]);
      })
      .join('');
    return (r + '000').slice(0, 4).toUpperCase();
  }
  splitStringByChunks(input: string = '', chunkLength: number =
100) {
    let current = chunkLength;
    let previous = 0;
    let output = [];
    while (input[current]) {
      if (input[current++] == ' ') {
        output.push(input.substring(previous, current));
        previous = current;
        current += chunkLength;
      }
    }
    output.push(input.substr(previous));
    return output;
  }
  setRedirectRecognizedTextOutput(action: Function) {
    if (typeof action != "function") {
      console.warn("Expected argument to be a function");
      return false;
    }
    this.settings.helpers.redirectRecognizedTextOutput = action;
    return true;
  }
  async restart() {
    const copyInit: Settings = this.settings;
    try {
      await this.fatality();
      await this.initialize(copyInit)
      return Promise.resolve()
    } catch (error) {
      return Promise.reject();
    }
  }
  talk(text: string, actualChunk: number, totalChunks: number,
callbacks: SayCallbacksObject) {
    const msg = new SpeechSynthesisUtterance();
    msg.text = text;
    msg.volume = this.settings.volume;
    msg.rate = this.settings.speed;
    let availableVoice = this.getVoice(this.settings.lang);
    if (callbacks) {
      if (callbacks.hasOwnProperty("lang")) {
```

```
        availableVoice = this.getVoice(callbacks.lang);
      }
    }
    if (this.device.isMobile) {
      if (availableVoice) {
        msg.lang = availableVoice.lang;
      }
    } else {
      msg.voice = availableVoice;
    }
    if (actualChunk == 1) {
      msg.addEventListener('start', () => {
        this.settings.isSpeaking = true;
        this.debug("Event reached : " + EVENTS.SYNTHESIS_START);
        this.triggerEvent(EVENTS.SYNTHESIS_START);
        if (callbacks) {
          if (typeof (callbacks.onStart) == "function") {
            callbacks.onStart.call(msg);
          }
        }
      });
    }
    if ((actualChunk) >= totalChunks) {
      msg.addEventListener('end', () => {
        this.settings.isSpeaking = false;
        this.debug("Event reached : " + EVENTS.SYNTHESIS_END);
        this.triggerEvent(EVENTS.SYNTHESIS_END);
        if (callbacks) {
          if (typeof (callbacks.onEnd) == "function") {
            callbacks.onEnd.call(msg);
          }
        }
      });
    }
    this.debug((actualChunk) + " text chunk processed succesfully
out of " + totalChunks);
    // Save the SpeechSynthesisUtterance object in memory, to
avoid lost of it
    this.talksHolder.push(msg);
    window.speechSynthesis.speak(msg);
  }
  say(message: string, callbacks ? : SayCallbacksObject) {
    let MAX_LENGTH = 115;
    let definitive = [];
    if (this.isSpeechSupported()) {
      if (!message || !message) {
        return console.warn("Empty string");
      }
      if (message.length > MAX_LENGTH) {
        let naturalReading = message.split(/,|:|\. |;/);
        naturalReading.forEach(chunk => {
```

```
          if (chunk.length > MAX_LENGTH) {
            let processed = this.splitStringByChunks(chunk,
MAX_LENGTH);
            definitive.push.apply(definitive, processed);
          } else {
            definitive.push(chunk);
          }
        });
      } else {
        definitive.push(message);
      }
      definitive = definitive.filter((e) => e);
      definitive.forEach((chunk, index) => {
        let numberOfChunk = (index + 1);
        if (chunk) {
          this.talk(chunk, numberOfChunk, definitive.length,
callbacks);
        }
      });
    }
  }
}
```