**CLOUD COMPUTING:PHASE 4**

**CHATBOT IN VOICE COMMUNICATION COMMUNICATION USING WATSON ASSISTANT:**

**Voice chatbot using Watson:**

Creating a voice chat bot using IBM Watson's services is possible. IBM Watson offers various tools and APIs for natural language understanding and speech recognition. Here are the general steps to create a voice chat bot using Watson:

Set up IBM Watson Services: Sign up for IBM Watson services and create instances of the services you need, such as IBM Watson Assistant for chat interactions and IBM Watson Speech to Text for converting spoken language into text.

Design the Conversation Flow: Define the conversation flow that your chat bot will follow. You can use the Watson Assistant tool to design dialog trees and intents for your chat bot.

Integrate Speech to Text: Configure your application to use IBM Watson Speech to Text service. This service will be responsible for converting spoken words into text, which can then be processed by your chat bot.

Speech Recognition: When a user speaks, use the Speech to Text service to transcribe the spoken words into text. You can use the Watson SDK or API for this purpose.

Pass Text to Chat Bot: Once you have the transcribed text, send it to your Watson Assistant instance to process the user's input and determine the appropriate response.

Dialog Management: Watson Assistant will handle the dialog management based on the user's input. You'll define responses and actions for different intents and entities.

Text to Speech: If you want your chat bot to respond with voice, you can use IBM Watson Text to Speech service to convert text responses into spoken words.

Connect Speech to Text and Text to Speech: Integrate the output from Watson Assistant with the Text to Speech service to generate a voice response.

Feedback and Iteration: Continuously improve your chat bot by analyzing user interactions and refining the conversation flow and responses.

on Assistant tool to design the conversation flow, including intents, entities, and dialog nodes for your chat bot.

Voice Input (Speech to Text):

Set up the Speech to Text service to convert voice input into text. You can use the IBM Watson SDK or API to integrate this service into your application.

**Ui PART:**

const { containerBootstrap, Nlp, LangEn } = window.nlpjs

​

// shortland function

const el = document.getElementById.bind(document)

​

// delay initialization until form is created

setTimeout(async () => {

const container = await containerBootstrap()

container.use(Nlp)

container.use(LangEn)

const nlp = container.get("nlp")

nlp.settings.autoSave = false

nlp.addLanguage("en")

​

// Adds the utterances and intents for the NLP

nlp.addDocument("en", "goodbye for now", "greetings.bye")

nlp.addDocument("en", "bye bye take care", "greetings.bye")

nlp.addDocument("en", "okay see you later", "greetings.bye")

nlp.addDocument("en", "bye for now", "greetings.bye")

nlp.addDocument("en", "i must go", "greetings.bye")

nlp.addDocument("en", "hello", "greetings.hello")

nlp.addDocument("en", "hi", "greetings.hello")

nlp.addDocument("en", "howdy", "greetings.hello")

​

// Train also the NLG

nlp.addAnswer("en", "greetings.bye", "Till next time")

nlp.addAnswer("en", "greetings.bye", "see you soon!")

nlp.addAnswer("en", "greetings.hello", "Hey there!")

nlp.addAnswer("en", "greetings.hello", "Greetings!")

​

await nlp.train()

​

// form submit event

async function onMessage(event) {

if (event) event.preventDefault()

const msg = el("message").value

el("message").value = ""

if (!msg) return

const userElement = document.createElement("div")

userElement.innerHTML = "<b>User</b>: " + msg

userElement.style.color = "blue"

el("history").appendChild(userElement)

const response = await nlp.process("en", msg)

const answer = response.answer || "I don't understand."

const botElement = document.createElement("div")

botElement.innerHTML = "<b>Bot</b>: " + answer

botElement.style.color = "green"

el("history").appendChild(botElement)

}

​

// Add form submit event listener

document.forms[0].onsubmit = onMessage

})

**VOICE PART:**

const { containerBootstrap, Nlp, LangEn } = window.nlpjs

​

// shortland function

const el = document.getElementById.bind(document)

​

function capitalize(string) {

return string.charAt(0).toUpperCase() + string.slice(1)

}

​

// initialize speech recognition

const SpeechRecognition =

window.SpeechRecognition || window.webkitSpeechRecognition

const recognition = SpeechRecognition ? new SpeechRecognition() : null

​

// how long to listen before sending the message

const MESSAGE\_DELAY = 3000

​

// timer variable

let timer = null

​

let recognizing = false

​

// delay initialization until form is created

setTimeout(async () => {

const container = await containerBootstrap()

container.use(Nlp)

container.use(LangEn)

const nlp = container.get("nlp")

nlp.settings.autoSave = false

nlp.addLanguage("en")

​

// Adds the utterances and intents for the NLP

nlp.addDocument("en", "goodbye for now", "greetings.bye")

nlp.addDocument("en", "bye bye take care", "greetings.bye")

nlp.addDocument("en", "okay see you later", "greetings.bye")

nlp.addDocument("en", "bye for now", "greetings.bye")

nlp.addDocument("en", "i must go", "greetings.bye")

nlp.addDocument("en", "hello", "greetings.hello")

nlp.addDocument("en", "hi", "greetings.hello")

nlp.addDocument("en", "howdy", "greetings.hello")

​

// Train also the NLG

nlp.addAnswer("en", "greetings.bye", "Till next time")

nlp.addAnswer("en", "greetings.bye", "see you soon!")

nlp.addAnswer("en", "greetings.hello", "Hey there!")

nlp.addAnswer("en", "greetings.hello", "Greetings!")

​

await nlp.train()

​

// initialize speech generation

let synthVoice = null

if ("speechSynthesis" in window && recognition) {

// wait until voices are ready

window.speechSynthesis.onvoiceschanged = () => {

synthVoice = text => {

clearTimeout(timer)

const synth = window.speechSynthesis

const utterance = new SpeechSynthesisUtterance()

// select some english voice

const voice = synth.getVoices().find(voice => {

return voice.localService && voice.lang === "en-US"

})

if (voice) utterance.voice = voice

utterance.text = text

synth.speak(utterance)

timer = setTimeout(onMessage, MESSAGE\_DELAY)

}

}

}

​

// form submit event

async function onMessage(event) {

if (event) event.preventDefault()

const msg = el("message").value

el("message").value = ""

if (!msg) return

const userElement = document.createElement("div")

userElement.innerHTML = "<b>User</b>: " + msg

userElement.style.color = "blue"

el("history").appendChild(userElement)

const response = await nlp.process("en", msg)

const answer = response.answer || "I don't understand."

const botElement = document.createElement("div")

botElement.innerHTML = "<b>Bot</b>: " + answer

botElement.style.color = "green"

el("history").appendChild(botElement)

if (synthVoice && recognizing) synthVoice(answer)

}

​

// Add form submit event listener

document.forms[0].onsubmit = onMessage

​

// if speech recognition is supported then add elements for it

if (recognition) {

// add speak button

const speakElement = document.createElement("button")

speakElement.id = "speak"

speakElement.innerText = "Speak!"

speakElement.onclick = e => {

e.preventDefault()

recognition.start()

}

document.forms[0].appendChild(speakElement)

​

// add "interim" element

const interimElement = document.createElement("div")

interimElement.id = "interim"

interimElement.style.color = "grey"

document.body.appendChild(interimElement)

​

// configure continuous speech recognition

recognition.continuous = true

recognition.interimResults = true

recognition.lang = "en-US"

​

// switch to listening mode

recognition.onstart = function () {

recognizing = true

el("speak").style.display = "none"

el("send").style.display = "none"

el("message").disabled = true

el("message").placeholder = "Listening..."

}

​

recognition.onerror = function (event) {

alert(event.error)

}

​

// switch back to type mode

recognition.onend = function () {

el("speak").style.display = "inline-block"

el("send").style.display = "inline-block"

el("message").disabled = false

el("message").placeholder = "Type your message"

el("interim").innerText = ""

clearTimeout(timer)

onMessage()

recognizing = false

}

​

// speech recognition result event;

// append recognized text to the form input and display interim results

recognition.onresult = event => {

clearTimeout(timer)

timer = setTimeout(onMessage, MESSAGE\_DELAY)

let transcript = ""

for (var i = event.resultIndex; i < event.results.length; ++i) {

if (event.results[i].isFinal) {

let msg = event.results[i][0].transcript

if (!el("message").value) msg = capitalize(msg.trimLeft())

el("message").value += msg

} else {

transcript += event.results[i][0].transcript

}

}

el("interim").innerText = transcript

}

}

})

**AI PART:**

<html>

<head>

<title>Speak</title>

<script src="./bundle.js"></script>

<script src="./index.js"></script>

</head>

<body>

<div id="history" style="height: 300px; overflow-y: scroll;"></div>

<form>

<input id="message" placeholder="Type your message" style="width: 70%;" />

<button id="send" type="submit">Send</button>

</form>

</body>

</html>

const { containerBootstrap, Nlp, LangEn } = window.nlpjs

​

;(async () => {

const container = await containerBootstrap()

container.use(Nlp)

container.use(LangEn)

const nlp = container.get("nlp")

nlp.settings.autoSave = false

nlp.addLanguage("en")

​

// Adds the utterances and intents for the NLP

nlp.addDocument("en", "goodbye for now", "greetings.bye")

nlp.addDocument("en", "bye bye take care", "greetings.bye")

nlp.addDocument("en", "okay see you later", "greetings.bye")

nlp.addDocument("en", "bye for now", "greetings.bye")

nlp.addDocument("en", "i must go", "greetings.bye")

nlp.addDocument("en", "hello", "greetings.hello")

nlp.addDocument("en", "hi", "greetings.hello")

nlp.addDocument("en", "howdy", "greetings.hello")

​

// Train also the NLG

nlp.addAnswer("en", "greetings.bye", "Till next time")

nlp.addAnswer("en", "greetings.bye", "see you soon!")

nlp.addAnswer("en", "greetings.hello", "Hey there!")

nlp.addAnswer("en", "greetings.hello", "Greetings!")

​

await nlp.train()

​

// Test it

const response = await nlp.process("en", "I should go now")

console.log(response)

})()

const core = require("@nlpjs/core")

const nlp = require("@nlpjs/nlp")

const langenmin = require("@nlpjs/lang-en-min")

​

window.nlpjs = { ...core, ...nlp, ...langenmin }