Problema 1 – Aplicación cliente – servidor que simule el juego “Adivina Quien” usando aplicaciones de internet y reconocimiento de voz.

Introducción

El reconocimiento de voz tiene sus raíces en la investigación realizada en los Laboratorios Bell a principios de la década de 1950. Los primeros sistemas se limitaban a un solo hablante y tenían vocabularios limitados de aproximadamente una docena de palabras. Los sistemas modernos de reconocimiento de voz han recorrido un largo camino desde sus contrapartes antiguas. Pueden reconocer el habla de múltiples hablantes y tienen enormes vocabularios en numerosos idiomas. El primer componente del reconocimiento de voz es, por supuesto, el habla. La voz debe convertirse del sonido físico a una señal eléctrica con un micrófono, y luego a datos digitales con un convertidor analógico a digital. Una vez digitalizado, se pueden usar varios modelos para transcribir el audio al texto. La mayoría de los sistemas modernos de reconocimiento de voz se basan en lo que se conoce como un modelo oculto de Markov (HMM). Este enfoque funciona bajo el supuesto de que una señal de voz, cuando se ve en una escala de tiempo lo suficientemente corta (por ejemplo, diez milisegundos), puede aproximarse razonablemente como un proceso estacionario, es decir, un proceso en el que las propiedades estadísticas no cambian con el tiempo. En un HMM típico, la señal de voz se divide en fragmentos de 10 milisegundos. El espectro de potencia de cada fragmento, que es esencialmente un gráfico de la potencia de la señal en función de la frecuencia, se asigna a un vector de números reales conocidos como coeficientes cepstrales . La dimensión de este vector suele ser pequeña, a veces tan baja como 10, aunque los sistemas más precisos pueden tener una dimensión 32 o más. La salida final del HMM es una secuencia de estos vectores. Para decodificar el discurso en texto, los grupos de vectores se combinan con uno o más fonemas, una unidad fundamental del discurso. Este cálculo requiere capacitación, ya que el sonido de un fonema varía de un hablante a otro, e incluso varía de un enunciado a otro por el mismo hablante. Luego se aplica un algoritmo especial para determinar la palabra (o palabras) más probables que producen la secuencia dada de fonemas. Uno puede imaginar que todo este proceso puede ser computacionalmente costoso. En muchos sistemas modernos de reconocimiento de voz, las redes neuronales se utilizan para simplificar la señal de voz utilizando técnicas para la transformación de características y la reducción de la dimensionalidad antes del reconocimiento de HMM. Los detectores de actividad de voz (VAD) también se

utilizan para reducir una señal de audio solo a las partes que probablemente contengan voz. Esto evita que el reconocedor pierda tiempo analizando partes innecesarias de la señal. Afortunadamente, como programador de Python, no tiene que preocuparse por nada de esto. Varios servicios de reconocimiento de voz están disponibles para su uso en línea a través de una API, y muchos de estos servicios ofrecen SDK de Python.

1. Elegir un paquete de reconocimiento de voz de Python

Existen varios paquetes para el reconocimiento de voz en PyPI. Algunos de ellos incluyen:

- apiai
- asamblea
- google-cloud-speech
- Pocketsphinx
- SpeechRecognition
- Watson-Developer-Cloud
- ingenio

Algunos de estos paquetes, como ingenio y apiai, ofrecen funciones integradas, como el procesamiento del lenguaje natural para identificar la intención de un hablante, que van más allá del reconocimiento básico del habla. Otros, como google-cloud-speech, se centran únicamente en la conversión de voz a texto. Hay un paquete que se destaca en términos de facilidad de uso: SpeechRecognition. Reconocer el habla requiere entrada de audio, y SpeechRecognition hace que recuperar esta entrada sea realmente fácil. En lugar de tener que crear scripts para acceder a micrófonos y procesar archivos de audio desde cero, SpeechRecognition lo pondrá en funcionamiento en solo unos minutos. La biblioteca SpeechRecognition actúa como un contenedor para varias API de voz populares y, por lo tanto, es extremadamente flexible. Una de estas, la API de Google Web Speech, admite una clave API predeterminada que está codificada en la biblioteca SpeechRecognition. Eso significa que puede levantarse sin tener que inscribirse en un servicio. La flexibilidad y la facilidad de uso del paquete SpeechRecognition lo convierten en una excelente opción para cualquier proyecto de Python. Sin embargo, el soporte para cada característica de cada APIque envuelve no está garantizado. Deberá pasar un tiempo investigando las opciones disponibles para averiguar si SpeechRecognition funcionará en su caso particular.

2. Desarrollo

Desarrollar una aplicación servidor que pueda atender múltiples clientes usando sockets y programación concurrente. El usuario debe indicar el número de clientes (jugadores) que participarán. Una vez que se conectan todos los clientes, el servidor puede iniciar el juego.

El servidor debe implementar un mecanismo de turnos para garantizar que cada cliente pueda mandar una suposición en el turno que le corresponde.

El juego consiste en adivinar un personaje mediante el envío de mensajes de voz. El servidor tiene un repositorio de 10 personajes y cada uno de ellos tiene cinco características principales. Pueden ser características físicas (color de cabello, ojos, etc), características sobresalientes en algún campo (artes visuales, música, etc.) o algún otro conjunto de características. El juego no debe mezclar características, las cinco deben ser físicas o de alguna categoría específica.

El servidor recibe el mensaje en forma de audio y usa un paquete de reconocimiento de voz para obtener el mensaje del cliente.

El servidor elige de manera aleatoria un personaje y recibe los mensajes del cliente en forma de pregunta. Si la pregunta empata con la característica física entonces el servidor contesta sí, en caso contrario responde no.

El servidor manda la actualización a todos los clientes para que puedan ver la pregunta y la respuesta.

Cuando un cliente manda un nombre el servidor lo compara con el personaje que ha elegido al inicio de la partida. Si el nombre es el mismo que el personaje, el jugador gana el juego. El juego termina cuando un jugador adivina el personaje.

El servidor debe notificar a todos los jugadores que el juego terminó, quién fue el ganador y cuánto tiempo tardó la partida.

En la aplicación cliente se debe ingresar la ip y el puerto del servidor. Debe mostrar un tablero con 10 personajes enlistando sus características.

La aplicación cliente permite mandar un audio con una pregunta de alguna característica del personaje.

La aplicación cliente puede ocultar un personaje a petición del usuario cuando se de cuenta que todas las características fueron contestadas con un "no" del servidor