El desarrollo del software de reconocimiento de voz para manejo de dispositivos de mesa

The development of voice recognition software for managing tabletop devices

Johan Eduardo Loaiza Quiñonez [1]
johan.loaiza00@usc.edu.co

Sally Maryuri Murillo Palacios [1]
sally.murillo00@usc.edu.co

Alejandro Marcus Martínez, Dr.2
Alejandro.marcus00@usc.edu.co

Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Tecnología en sistemas de información (1)
Universidad Santiago de Cali, Facultad de Ingeniería (2)

Resumen

El reconocimiento de voz es uno de los métodos con los que el ser humano avanza en su interacción con la tecnología, la relación hombre-máquina se desarrolla a través de acciones como el reconocimiento facial, las huellas dactilares, el reconocimiento de voz, todo lo que le permite a la tecnología reconocer a las personas, y a la vez les permite a estas comunicarse con esta para ordenar a la máquina a realizar acciones. En este trabajo se muestran dichos avances, los modelos que se diseñaron para dar respuesta a los requerimientos que se presentaron a medida que este sistema evoluciona, los métodos aplicados para estas evaluaciones, que consisten en 4 etapas, empezando con preprocesar el audio, realizando el análisis de su duración, su amplitud y la frecuencia en la que se transmite, segmentarlo para conseguir el contenido del audio, extraer sus características como la presencia de ruido de ambiente y el idioma del audio para distinguir la presencia de un acento que sea posible evaluar, y por último la comparación de los resultados obtenidos en la evaluación, todos esos procesos hechos aplicando la metodología de Mozilla Deepspeech, con la que se realiza pruebas comparando texto obtenido por la metodología y el real obtenido del audio, con el objetivo de que el sistema pueda obtener la interpretación más clara posible del audio que está recibiendo, dando como resultado el entendimiento del audio que recibe el programa, la forma en la que lo recibe, y la capacidad que tiene para analizarlo, y este resultado dará continuidad a la evolución de las capacidades de la tecnología, siendo otra demostración a como el ser humano evoluciona también para cada vez hacer más estrecha a relación con la tecnología.

Palabras Clave: tecnología; evolución; interacción; reconocimiento de voz

Abstract

Voice recognition is one of the methods with which human beings advance in their interaction with technology, the man-machine relationship is developed through actions such as facial recognition, fingerprints, voice recognition, everything that allows technology to recognize people, and at the same time allows them to communicate with it to order the machine to perform actions. In this work, we will take a look at these advances, the models that were designed to respond to the requirements that arose as this system evolved, the methods applied for these evaluations, which consist of 4 stages, starting with preprocessing the audio , performing the analysis of its duration, its amplitude and the frequency in which it is transmitted, segmenting it to obtain the content of the audio, extracting its characteristics such as the presence of ambient noise and the language of the audio to distinguish the presence of an accent that It is possible to evaluate, and finally the comparison of the results obtained in the evaluation, all those processes made applying the Mozilla Deepspeech methodology, with which tests are carried out comparing the text obtained by the methodology and the real one obtained from the audio, with the objective that the system can obtain the clearest possible interpretation of the audio it is receiving, resulting in the understanding of the au He said that he receives the program, the way in which he receives it, and the capacity he has to analyze it, and this result will give continuity to the evolution of the capabilities of technology, being another demonstration of how the human being also evolves for each time make a closer relationship with technology.

Keywords: technology; evolution; interaction; speech recognition

INTRODUCCIÓN

La comunicación más natural desarrollada a través del tiempo en los seres humanos es la voz, en un proceso que se puede ver como una cadena de habla, con la cual dos personas, una en calidad de emisor y otra en calidad de receptor, se pueden comunicar entre sí (Ramírez, 2002). En los últimos años, los avances tecnológicos han llevado a la humanidad a la implementación de herramientas capaces de tomar decisiones con base a tareas previamente establecidas. Por lo cual, se realiza el esfuerzo de ingenieros y científicos por reemplazar una de las dos personas que establecen la comunicación por sistemas inteligentes con la capacidad de recibir o dar respuestas coherentes en un entorno específico, mediante el reconocimiento de comandos simples de voz (Lee, 2015).

El reconocimiento de voz se realiza comúnmente mediante algoritmos implementados en software, aprovechando las ventajas que ofrecen los lenguajes de alto nivel para este tipo de aplicaciones y el poder de cómputo de los procesadores actuales. Las aplicaciones que involucran reconocimiento de voz forman parte de la vida cotidiana, encontrándose en gran cantidad de dispositivos tales como los teléfonos inteligentes, computadores, electrodomésticos, vehículos, etc. Aunque hay varios sistemas comerciales de reconocimiento de voz disponibles en el mercado, el problema del reconocimiento automático de voz no debe considerarse resuelto de ningún modo, incluso si los medios de comunicación y también algunos investigadores a veces dan esa impresión (Ruiz & del Mar, 2017).

Por su parte, Python, fue lanzado por primera vez en 1991 por Guido van Rossum, científico de la computación de la universidad de Amsterdam, y que en la actualidad continúa siendo desarrollado por la Python Software Foundation (Challenger-Pérez, Díaz-Ricardo, & Becerra-García, 2014).

Python es un lenguaje de programación, que tiene la facultad de trabajar con cualquier tipo de programa, es ese sentido, con el uso de ese lenguaje se puede desarrollar software para app científicas, para comunicaciones de red, para app de escritorio con interfaz gráfica de usuario (GUI); pero también es adaptable para los juegos de video, para smartphones, para inteligencia artificial y para la programación web (Intelligent, 2018).

En cuanto al problema de trabajo, al pensar en los desafíos que se hacen presentes en el ejercicio del reconocimiento de voz se puede obtener los conceptos de los diversos obstáculos que se afrontan, desafíos presentes desde su origen, y que conforme se fue desarrollando este sistema, se fue haciendo más fácil la solución de estos desafíos detallados a continuación: el ruido, es en una situación normal será difícil que el hablante se encuentre en un ambiente completamente limpio, es decir sin ruido que se origine en su ambiente, este ruido de ambiente se convierte en un obstáculo pues se hace necesario aislar la el ruido generado por la voz del hablante, del ruido generado en el ambiente, proceso del cual depende la capacidad de la máquina de obtener el mensaje con claridad; mientras que, los ecos generados por la propia voz del hablante en espacios cerrados o por cualquier objeto en el que las ondas sonoras puedan rebotar se vuelve otro obstáculo, pues reduce la claridad que la máquina tiene para entender el mensaje al estar recibiendo múltiples ondas de la misma voz (Striuk, 2019).

Por su parte, los acentos se aprecian como otro obstáculo para un reconocimiento de voz claro, pues al ser posible decir una misma palabra con una gran diversidad de maneras, la pronunciación y fonética estará sujeta a cambios, provocando dificultad a la máquina de entenderlas (Striuk, 2019); mientras que los sonidos similares, también podría considerarse como redundancia, pues los casos en los que al hablar se mencionan palabras u oraciones que se escuchan de manera similar pueden confundir a la máquina y entorpecer su función de codificación y decodificación, también, indicar que, el punto vital en el proceso de reconocimiento de voz, las máquinas aún son capaces de presentar desde un 8% hasta un 12% de errores al procesar la voz, significando que los sistemas aún no son perfectos, sin embargo, claro está que estos errores proveen los datos necesarios para continuar mejorando estos sistemas; por lo tanto, una persona normal habla una enorme cantidad de palabras, a más palabras habladas, más datos deberá analizar la máquina, si se habla de una forma enredada, la máquina no podrá analizar el habla correctamente para distinguir palabras clave con las que determinar una acción, lo cual afecta en su desempeño.

Queda claro que las máquinas tampoco son perfectas para realizar estos trabajos, se requiere de mucha experimentación para lograr resultados adecuados que permitan mejorar su desempeño, sin embargo, gracias a los rápidos avances que se presentan en los campos de la ciencia y la tecnología, un futuro en el que las máquinas se puedan comunicar abiertamente

es una visión que cada vez está más cerca de cumplirse.

En este este orden de ideas, este trabajo tiene como objetivo general, escribir los principales modelos de reconocimiento de comandos simples de voz, para facilitar el manejo de dispositivos, para ellos se buscó realizar tres fases: la primera, analizar la problemática inicial que impulsó el desarrollo de sistemas de reconocimiento de voz; la segunda, estudiar las diferentes fases de evolución que tuvieron los sistemas de reconocimiento de voz; por último, la tercera hace alusión a Exponer los principales modelos desarrollados para el análisis de voz, incluyendo principales técnicas y resultados.

EVOLUCIÓN HISTÓRICA

El reconocimiento de voz tiene sus orígenes en la investigación hecha en los *laboratorios BELL* en la década de 1950 con el desarrollo del primer software capaz de reconocer voz, conocido como *Audrey*; este era un hardware, que para la época era demasiado extenso y ocupaba una habitación entera, pero solo tenía la capacidad de reconocer únicamente 9 dígitos(números del 1 al 9) que su diseñador pronunciaba previamente, sin embargo, y para la época, hacia este análisis con una precisión del 90% (Soto, 2018).

Durante esta época su propósito era ayudar a los operadores de peaje a tomar llamadas, pero debido a los elevados costos, y la corta gama de palabras que podía reconocer, lo hicieron poco práctico para este proceso; posteriormente, siguiente paso en la evolución del reconocimiento de voz llegó en los años 1960, cuando la empresa IBM desarrolla Shoebox (ver figura 1), un sistema que además de tener la capacidad de reconocer una mayor gama de palabras, era capaz de reconocer comandos aritméticos, como "mas" y "total", además de poder usar esta capacidad para comunicarse con una calculadora que luego se encargaría de hacer el cálculo, además, era capaz de entender 16 palabras del idioma inglés y posteriormente, para 1963 los laboratorios NEC construyeron un reconocedor de dígitos.

Figura 1. IBM ingeniero William C. Dersch, mostrado en 1961

Nota. Tomado de (IBM, 2022)

Para los años de 1970, surgen las técnicas de parametrización de la señal de voz, basado en las técnicas de programación dinámica los laboratorios Bell, este demostró que, las ideas que se habían realizado para la Codificación Predictiva Lineal (LPC) ya podían de forma más ágil, implementarse en los sistemas de reconocimiento través del uso *de una medida de distancia apropiada* (Soto, 2018)*;* a partir de allí, los laboratorios de IBM fueron desarrollando sistemas de reconocimiento de voz de amplio vocabulario, mientras que, por su lado, los laboratorios AT&T Bell ya ideaban sistemas de reconocimiento de voz independientes del locutor (Sakoe & Chiba, 1978).

Para el año de 1970, el profesor Gunnar Real Instituto de Tecnología de Estocolmo, planteó un nuevo modelo de reconocimiento de voz, pero poseía cuatro características: a)detección de segmentos; b)transcripción fonética; c) identificación de palabras; y, d) interpretación semántica; luego surge el sistema de reconocimiento HEARSAY, que de forma paralela trabajaba con tres reconocedores; uno acústico, uno sintáctico y otro semántico. "La frase que se deseaba conocer era pre-procesada para extraer sus características, el resultado era procesado por los tres módulos y finalmente un

sistema mediador que almacenaba toda la información controlaba el sistema y daba el resultado" (Soto, 2018, p.13).

También se destacan otros avances, en las décadas posteriores, que pueden resumirse así (Ver tabla 1):

Tabla 1. Evolución histórica del reconocimiento de voz

Proyecto	Año	Función
Gunnar Fant(1919-2009) realiza un modelo de reconocimiento dividido en 5 fases: extracción de parámetros, detección de segmentos, transcripción fonética, identificación de palabras e interpretación semántica	1970	El aporte del señor Fant plantaría las bases del modus operandi que se aplicaría a los sistemas de reconocimiento de voz en el futuro.
ARPA-SUR (Advanced Research Projects Agency-Speech Understanding Research)	1971	Tenía la capacidad de reconocer discursos continuos, adaptarse a un diccionario de 10,000 palabras, y poseía un error inferior al 10% y una respuesta en tiempo real.
IBM desarrollo la estructura de un modelo de lenguaje (gramatical)- (modelo de n-gramas)	1980	A partir de una secuencia de símbolos de idioma, podía aparecer en la señal de voz. Definía la probabilidad de ocurrencia de una secuencia ordenada de "n" palabras.
Las redes neuronales artificiales (Artificial Neural Networks)	1980	Compartían con los modelos ocultos de Markov su carácter inductivo, pero su aprendizaje se basaba a partir de muestras.
Los costos de las aplicaciones de reconocimiento de voz se reducen	1990	Los costos decrecen y los vocabularios extensos se hacen normales. Las aplicaciones independientes del locutor y de flujo continuo se hacen más comunes.
AT&T introduce Voice Recognition Call Processing System	1992	Sistema de llamadas por cobrar, a finales de 1993 procesaba de 50 millones de llamadas mensuales
"Habla Espontánea: Corpus y Tecnología de Procesamiento. (modelado acústico flexible)	2000	Tenía un aproximado de 7 millones de palabras, que correspondían a 700 horas de discurso
En 2010, Google añadió el reconocimiento personalizado a la búsqueda por voz en teléfonos Android, para que el software pudiera registrar las búsquedas de voz de los usuarios producir un modelo más preciso	2010	El desarrollo de la tecnología de reconocimiento de voz permitió la llegada de la aplicación Google Voice Search para el iPhone.
Chrome ideo el corpus de 10 a 100 palabras.	2011	Hoy en día el sistema de búsqueda por voz en inglés de Google incorpora 230 mil millones de palabras para las consultas reales de los usuarios.

Nota. Elaboración propia. Adaptado de (Soto, 2018).

ESTADO DEL ARTE

El lenguaje de programación utilizado para la evaluación del reconocimiento de voz es Python, este es un lenguaje multiplataforma y de código abierto que es aplicado tanto al desarrollo web, como a la creación de software y el procesamiento de datos (Intelligent, 2018).

Para entrar un poco en contexto sobre Python, fue lanzado por primera vez en 1991 por Guido van Rossum, científico de la computación de la universidad de Amsterdam, y que en la actualidad continúa siendo desarrollado por la Python Software Foundation (Challenger-Pérez, Díaz-Ricardo, & Becerra-García, 2014).

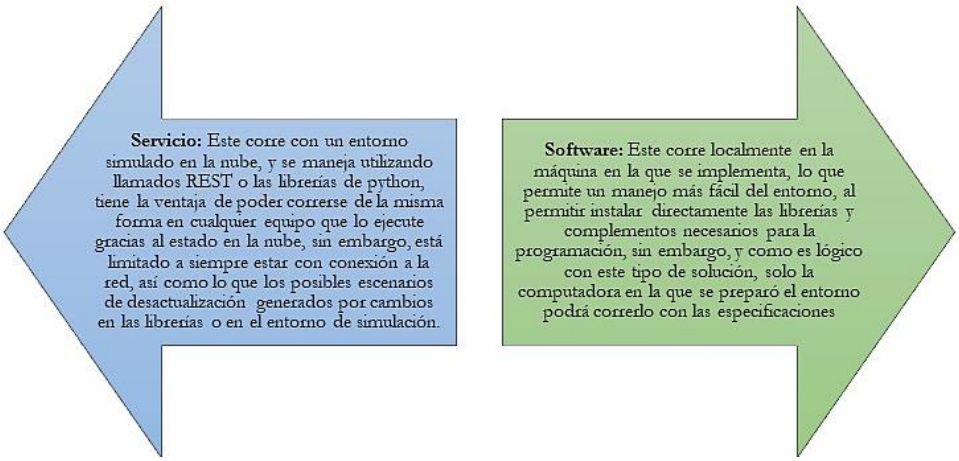
A continuación se presentara un modelo desarrollado para la evaluación del reconocimiento de voz utilizando el lenguaje de programación Python, estos modelos fueron programados en el entorno de Google Colab, que es un entorno de simulación que utiliza este lenguaje para desarrollar código sin necesidad de instalar lenguaje Python ni sus librerías en el dispositivo, aunque comprendiendo las posibles limitaciones que esto conlleva debido a la situación que es que este entorno corre en versiones antiguas de Python, esto bien no presenta un gran impedimento, se pueden presentar múltiples inconvenientes conforme pase el tiempo.

Reconocimiento de voz con Python

Python es un lenguaje de programación, que tiene la facultad de trabajar con cualquier tipo de programa, es ese sentido, con el uso de ese lenguaje se puede desarrollar software para app científicas, para comunicaciones de red, para app de escritorio con interfaz gráfica de usuario (GUI); pero también es adaptable para los juegos de video, para smartphones,

para inteligencia artificial y para la programación web (Intelligent, 2018); por otro lado, Python posee dos tipos de soluciones (ver figura 2):

Figura 2. Soluciones del Python



Nota. Elaboración propia, adaptado de (Mosquera, 2019).

Figura 3. Compensaciones de usar el servicio de nube de voz frente al alojamiento propio de un paquete de software ASR

COMPENSASIONES	SERVICIO	SOFTWARE
Gastos generales de recopilación de datos y entrenamiento de modelos	👍	👎
Gastos generales de implementación, mantenimiento y escalado del servicio	👍	👎
Flexibilidad de personalización/sesgo del modelo	●	👍
Capacidad de implementación en el dispositivo / borde	👎	👍
Optimización de costos de operación	👎	👍

Nota. Lo representado sobre Servicio respecto a la flexibilidad representa el estado neutro en que se encuentra al respecto. Elaboración propia, adaptado de (Satish, 2020).

El reconocimiento de voz también trabaja haciendo uso de dos sistemas de APIs:

- **Batch:** Trabaja utilizando un archivo de audio que es enviado como parámetro, sea llamado desde una ubicación local o en la nube, y realiza una transcripción inmediata de las palabras contenidas en el audio.

- **Streaming**: Trabaja transmitiendo repetidamente el audio, transcribe parcialmente el texto en cada vez, lo que permite obtener resultados intermedios.

METODOLOGÍA

Este trabajo es de tipo documental, basado en las fuentes de información secundaria, basada en estudios y recopilación en bases de datos electrónicas disponible mediante las bases de datos, para la búsqueda y selección de esta información bibliográfica, su rastreo se hará por medio de las palabras claves tecnología; evolución; interacción; reconocimiento de voz. Durante la búsqueda de artículos, se seleccionó como criterio para la selección e inclusión de documentos estas características: estudios de investigación de los últimos 12 años de bases de datos indexados, tesis de investigación, estudios

institucionales, documentos y revistas científicas, páginas institucionales, salvo aquellos que se consideran teóricos expertos, y válidos para este trabajo; además, se utilizó textos son los idiomas español – inglés – portugués, considerando que estos tres, es donde se reposan los mayores estudios en materia bibliografía sobre el reconocimiento de voz.

MODELOS DESARROLLADOS PARA EL ANÁLISIS DE VOZ

Como resultado de la búsqueda documental y realización de pruebas en un entorno simulado, a continuación, se muestran los diferentes modelos utilizados para realizar el ejercicio del reconocimiento de voz, se explicarán sus cualidades y el modo para aplicarlos:

Servicio:

1: Speech Google Cloud

2: Speech Microsoft Azure

3: Speech IBM Watson

Software:

4: Speech CMU Sphinx

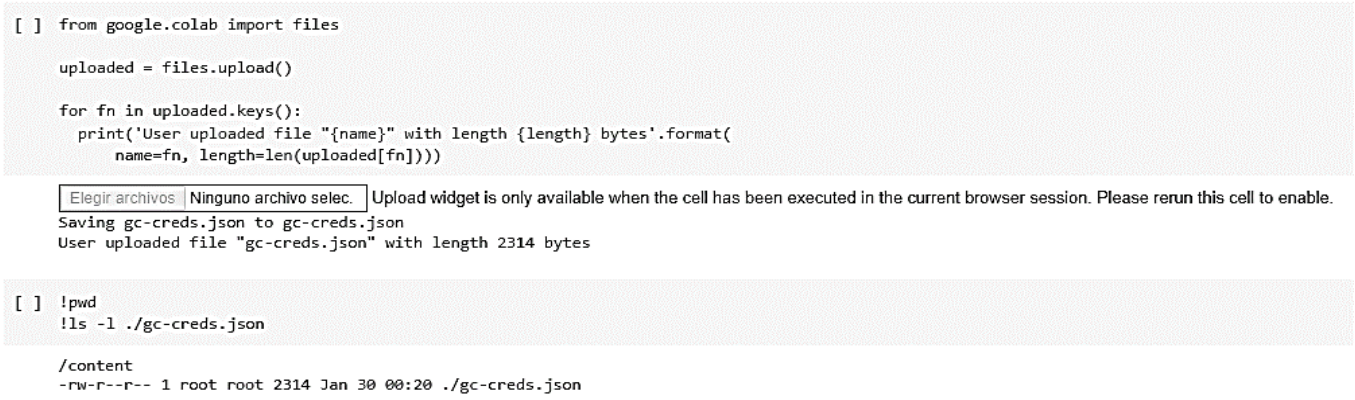
5: Mozilla DeepSpeech

6: Python Speech Recognition

Speech Google Cloud

El servicio Speech-to-Text integrado de google cloud es un modelo de reconocimiento de voz que es utilizado en las librerías de C#, Go, Java, JavaScript, Python, PHP y Ruby; por otro lado, permite la integración sencilla de tecnologías de reconocimiento de voz de Google en aplicaciones para desarrolladores (Anggraini, Kurniawan, Wardhani, y Hakim, 2020). Para empezar a trabajar con este módulo se hace necesario iniciarlo con credenciales que autoricen su uso, estas credenciales pueden buscarse como parámetro directamente desde la web, o traerlo como un archivo local (figura 4).

Figura 4. Obtención de credenciales localmente



Luego, una vez obtenidas las credenciales se genera el área de trabajo (ver figura 5).

Figura 5. Generación del ambiente

```
[ ]  import os
     os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/content/gc-creds.json'

     !ls -l $GOOGLE_APPLICATION_CREDENTIALS

     -rw-r--r-- 1 root root 2314 Jan 30 00:20 /content/gc-creds.json
```

Luego de la generación del ambiente o la preparación del entorno, se realiza la programación requerida por el modelo para ejecutar las API Batch y el Streaming como lo visualizan en el ejemplo de la figura 6.

Figura 6. API Batch, API Streaming y API Streaming conforme repite los audios cargados

Batch API

```
[ ]  from google.cloud import speech_v1
     from google.cloud.speech_v1 import enums

     def google_batch_stt(filename: str, lang: str, encoding: str) -> str:
         buffer, rate = read_wav_file(filename)
         client = speech_v1.SpeechClient()

         config = {
             'language_code': lang,
             'sample_rate_hertz': rate,
             'encoding': enums.RecognitionConfig.AudioEncoding[encoding]
         }

         audio = {
             'content': buffer
         }

         response = client.recognize(config, audio)
         # For bigger audio file, the previous line can be replaced with following:
         # operation = client.long_running_recognize(config, audio)
         # response = operation.result()

         for result in response.results:
             # First alternative is the most probable result
             alternative = result.alternatives[0]
             return alternative.transcript
```

```
[ ] from google.cloud import speech
    from google.cloud.speech import enums
    from google.cloud.speech import types

    def response_stream_processor(responses):
        print('interim results: ')

        transcript = ''
        num_chars_printed = 0
        for response in responses:
            if not response.results:
                continue

            result = response.results[0]
            if not result.alternatives:
                continue

            transcript = result.alternatives[0].transcript
            print('{0}final: {1}'.format(
                '' if result.is_final else 'not ',
                transcript
            ))

        return transcript
```

```
# Run tests
for t in TESTCASES:
    print('\naudio file="{0}"    expected text="{1}"'.format(t['filename'], t['text'])
    print('google-cloud-streaming-stt: "{}"'.format(
        google_streaming_stt(t['filename'], t['lang'], t['encoding'])
    ))
```

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
interim results:
not final: next
not final: iSpy
not final: Aspira
not final: Xperia
not final: Experian
not final: experience
not final: experience proved
not final: experience proves
not final: experience proves the
not final: experience proves that
not final: experience
final: experience proves this
google-cloud-streaming-stt: "experience proves this"
```

Nota. API Batch (Arriba), API Streaming (izquierda) y API Streaming conforme repite los audios cargados (derecha).

Como se puede observar, el modelo trabaja analizando y ejecutando repetidamente la grabación utilizada en el ejemplo, con el fin de mejorar su capacidad para entender las palabras pronunciadas, se imprimen los resultados que obtiene en cada ejecución del audio, afinando la exactitud para emparejar las palabras que el modelo genera con las que se pronuncian en el audio, hasta ser capaz de replicar perfectamente la oración (Google Cloud, 2022).

Speech Microsoft Azure

Servicio speech-to-text integrado en las funciones de Microsoft azure, trabaja con los mismos métodos Batch y Streaming que Google Cloud, este posee tres fases (ver figura 7): la instalación; el método Batch; método Streaming; y, por último, los resultados, este trabaja con el Speech Studio, caracterizado por ser un conjunto de herramientas basadas en la interfaz de usuario para crear e integrar características del servicio Voz de Azure Cognitive Services en las aplicaciones (Kim y Lui, 2019).

Figura 7. Proceso de **Speech Microsoft Azure**

Microsoft Azure

Microsoft Azure Speech Services have Speech to Text service.

Setup

1. **Install azure speech package**

```
[ ] !pip3 install azure-cognitiveservices-speech

    Collecting azure-cognitiveservices-speech
      Downloading https://files.pythonhosted.org/packages/a0/d8/690896a3543b7bed058029b1b3450f4ce2e952d19347663fe570e6dec72c/azure_cognitiveservices_speech-1.9.0-cp36-cp36m-manylinux1_x86_64.whl (3.9MB)
        |████████████████████████████████| 3.9MB 3.4MB/s
    Installing collected packages: azure-cognitiveservices-speech
    Successfully installed azure-cognitiveservices-speech-1.9.0
```

2. **Set service credentials**

You can enable Speech service and find credentials for your account at Microsoft Azure portal. You can open a free account here.

```
[ ] AZURE_SPEECH_KEY = 'YOUR AZURE SPEECH KEY'
    AZURE_SERVICE_REGION = 'YOUR AZURE SERVICE REGION'
```

Batch API

```python
import azure.cognitiveservices.speech as speechsdk

def azure_batch_stt(filename: str, lang: str, encoding: str) -> str:
    speech_config = speechsdk.SpeechConfig(
        subscription=AZURE_SPEECH_KEY,
        region=AZURE_SERVICE_REGION
    )
    audio_input = speechsdk.AudioConfig(filename=filename)
    speech_recognizer = speechsdk.SpeechRecognizer(
        speech_config=speech_config,
        audio_config=audio_input
    )
    result = speech_recognizer.recognize_once()

    return result.text if result.reason == speechsdk.ResultReason.RecognizedSpeech else None

# Run tests
for t in TESTCASES:
    print('\naudio file="{0}"    expected text="{1}"'.format(t['filename'], t['text']))
    print('azure-batch-stt: "{}"'.format(
        azure_batch_stt(t['filename'], t['lang'], t['encoding'])
    ))
```

Streaming API

```python
[ ] import time
    import azure.cognitiveservices.speech as speechsdk

    def azure_streaming_stt(filename: str, lang: str, encoding: str) -> str:
        speech_config = speechsdk.SpeechConfig(
            subscription=AZURE_SPEECH_KEY,
            region=AZURE_SERVICE_REGION
        )
        stream = speechsdk.audio.PushAudioInputStream()
        audio_config = speechsdk.audio.AudioConfig(stream=stream)
        speech_recognizer = speechsdk.SpeechRecognizer(
            speech_config=speech_config,
            audio_config=audio_config
        )

        # Connect callbacks to the events fired by the speech recognizer
        speech_recognizer.recognizing.connect(lambda evt: print('interim text: "{}"'.format(evt.result.text)))
        speech_recognizer.recognized.connect(lambda evt:  print('azure-streaming-stt: "{}"'.format(evt.result.text)))
```

Con Azure es posible observar el mismo funcionamiento que con el método de Google, pero como se muestra a continuación, presenta diferencias notables en cuanto a la cantidad de veces que el modelo necesita repetir el audio hasta conseguir coincidir sus resultados con la oración dada por el audio, esto gracias a un mejor sistema que le brinda un mejor entendimiento de las palabras que analiza.

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
interim text: "experience"
interim text: "experienced"
interim text: "experience"
interim text: "experience proves"
interim text: "experience proves this"
azure-streaming-stt: "Experience proves this."

audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
interim text: "huaisheng"
interim text: "white"
interim text: "whi should"
interim text: "whi should one"
interim text: "whi should one halt"
interim text: "whi should one halt on"
interim text: "whi should one halt on the"
interim text: "whi should one halt on the way"
azure-streaming-stt: "Whi should one halt on the way."

audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
interim text: "you're"
interim text: "your"
interim text: "your power"
interim text: "your"
interim text: "your power is"
interim text: "your power is sufficient"
interim text: "your power is sufficient i"
interim text: "your power is sufficient i said"
azure-streaming-stt: "Your power is sufficient I said."
```

Las fases comprenden la *instalación*, en esta es donde se planea el producto y las funciones de Microsoft azure; más abajo, aparece el *método Batch*, este es el proceso mediante el cual una computadora trabajo por lotes de trabajos, a menudo simultáneamente, en orden secuencial y sin parar; lo importante este este comando es que, garantiza que los trabajos grandes para que se calculen en partes pequeñas, para mejorar la eficiencia durante el proceso de depuración (Gomar, 2018); más abajo está el *método Streaming,* que básicamente es un tipo de tecnología multimedia que envía contenidos de vídeo y audio a su dispositivo conectado a Internet, su funcionalidad permite que se pueda acceder a otros equipos y tecnológicas, o softwares en cualquier momento y desde un PC o un móvil (Pérez, 2020).

Speech IBM Watson

El método desarrollado por IBM, la misma empresa que desarrolló la primera máquina capaz de reconocer voz, este método permite reconocimiento y lectura rápida y precisa de múltiples idiomas en distintos casos de uso, maneja sus propios métodos Batch y Streaming para realizar esta labor.

La tecnología IBM Watson Speech to Text, es una aplicación muy veloz y sobre todo preciso, usa la transcripción de voz en varios idiomas para diversos casos de usos, como por el ejemplo el autoservicio de clientes, la asistencia de agente y la analítica de voz; este método se desarrolla con los *métodos Batch* y el *método Streaming* (ver figura 8).

Figura 8. Método Speech IBM Watson

Batch API

```
[ ]  import os

     from ibm_watson import SpeechToTextV1
     from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

     def watson_batch_stt(filename: str, lang: str, encoding: str) -> str:
          authenticator = IAMAuthenticator(WATSON_API_KEY)
          speech_to_text = SpeechToTextV1(authenticator=authenticator)
          speech_to_text.set_service_url(WATSON_STT_URL)

          with open(filename, 'rb') as audio_file:
               response = speech_to_text.recognize(
                    audio=audio_file,
                    content_type='audio/{}'.format(os.path.splitext(filename)[1][1:]),
                    model=lang + '_BroadbandModel',
                    max_alternatives=3,
               ).get_result()

          return response['results'][0]['alternatives'][0]['transcript']

     # Run tests
     for t in TESTCASES:
          print('\naudio file="{0}"     expected text="{1}"'.format(t['filename'], t['text']))
          print('watson-batch-stt: "{}"'.format(
               watson_batch_stt(t['filename'], t['lang'], t['encoding'])
          ))
```

Streaming API

Streaming API works over websocket.

```
[ ]  import json
     import logging
     import os
     from queue import Queue
     from threading import Thread
     import time

     from ibm_watson import SpeechToTextV1
     from ibm_watson.websocket import RecognizeCallback, AudioSource
     from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

     # Watson websocket prints justs too many debug logs, so disable it
     logging.disable(logging.CRITICAL)

     # Chunk and buffer size
     CHUNK_SIZE = 4096
     BUFFER_MAX_ELEMENT = 10

     # A callback class to process various streaming STT events
     class MyRecognizeCallback(RecognizeCallback):
         def __init__(self):
             RecognizeCallback.__init__(self)
             self.transcript = None
```

```
     def on_listening(self):
         # print('Service is listening')
         pass

     def on_hypothesis(self, hypothesis):
         # print('hypothesis: {}'.format(hypothesis))
         pass

     def on_data(self, data):
         self.transcript = data['results'][0]['alternatives'][0]['transcri
         print('{0}final: {1}'.format(
             '' if data['results'][0]['final'] else 'not ',
             self.transcript
         ))

     def on_close(self):
         # print("Connection closed")
         pass

def watson_streaming_stt(filename: str, lang: str, encoding: str) -> str:
     authenticator = IAMAuthenticator(WATSON_API_KEY)
     speech_to_text = SpeechToTextV1(authenticator=authenticator)
     speech_to_text.set_service_url(WATSON_STT_URL)

     # Make watson audio source fed by a buffer queue
     buffer_queue = Queue(maxsize=BUFFER_MAX_ELEMENT)
     audio_source = AudioSource(buffer_queue, True, True)

     # Callback object
```

Con Watson se vuelve a observar el mismo patrón de trabajo, que consiste en realizar un análisis múltiples veces al audio, cada vez que el audio se corre y se analiza se genera una palabra, y dicha palabra es comparada con la que se encuentra en el audio, repitiendo el proceso hasta que la frase es completada.

```
    audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
    watson-batch-stt: "experience proves this "

    audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
    watson-batch-stt: "why should one hold on the way "

    audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
    watson-batch-stt: "your power is sufficient I set "
```

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
not final: X.
not final: experts
not final: experience
not final: experienced
not final: experience prove
not final: experience proves
not final: experience proves that
not final: experience proves this
final: experience proves this
watson-cloud-streaming-stt: "experience proves this "

audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
not final: why
not final: what
not final: why should
not final: why should we
not final: why should one
not final: why should one whole
not final: why should one hold
not final: why should one hold on
not final: why should one hold on the
not final: why should one hold on the way
final: why should one hold on the way
watson-cloud-streaming-stt: "why should one hold on the way "

audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
not final: your
not final: your power
not final: your power is
not final: your power is the
not final: your power is sufficient
not final: your power is sufficient I
not final: your power is sufficient I saw
not final: your power is sufficient I said
not final: your power is sufficient I set
final: your power is sufficient I set
watson-cloud-streaming-stt: "your power is sufficient I set "
```

Nota. Imágenes de procedimientos y resultados tomadas de Speech Recognition wash Python Satish 2020

Speech CMU Sphinx

Es una metodología de reconocimiento de voz que inició su desarrollo en el año 2000 en la universidad de Carnegie Mellon, funciona haciendo uso conjunto de los demás sistemas de reconocimiento de voz de Sphinx (Sphinx 2 - 4) y un entrenador de modelo acústico(SphinxTrain); todos los modelos de Sphinx integrados en esta metodología cumplen la misma función de permitir el reconocimiento de habla, cada uno siendo más sofisticado que el anterior (ver figura 9).

Figura 9. Método Speech CMU Sphinx

```
[ ] !pip3 install pocketsphinx
    !pip3 list | grep pocketsphinx

    Collecting pocketsphinx
      Downloading https://files.pythonhosted.org/packages/cd/4a/adea55f189a81aed88efa0b0e1d25628e5ed22622ab9174bf696dd4f9474/pocketsphinx-0.1.15.tar.gz (29.1MB)
         |████████████████████████████████| 29.1MB 102kB/s
    Building wheels for collected packages: pocketsphinx
      Building wheel for pocketsphinx (setup.py) ... done
      Created wheel for pocketsphinx: filename=pocketsphinx-0.1.15-cp36-cp36m-linux_x86_64.whl size=30126870 sha256=d111bc1a768251e9b8b4bea71f05b498955eda209f5d5650f7e68cc
      Stored in directory: /root/.cache/pip/wheels/52/fd/52/2f62c9a0036940cc0c89e58ee0b9d00fcf78243aeaf416265f
    Successfully built pocketsphinx
    Installing collected packages: pocketsphinx
    Successfully installed pocketsphinx-0.1.15
    pocketsphinx                 0.1.15
```

Batch API

```
def sphinx_batch_stt(filename: str, lang: str, encoding: str) -> str:
    buffer, rate = read_wav_file(filename)
    decoder.start_utt()
    decoder.process_raw(buffer, False, False)
    decoder.end_utt()
    hypothesis = decoder.hyp()
    return hypothesis.hypstr

# Run tests
for t in TESTCASES:
    print('\naudio file="{0}"    expected text="{1}"'.format(t['filename'], t['text']))
    print('sphinx-batch-stt: "{}"'.format(
        sphinx_batch_stt(t['filename'], t['lang'], t['encoding'])
    ))
```

Streaming API

```
[ ]  def sphinx_streaming_stt(filename: str, lang: str, encoding: str) -> str:
        buffer, rate = read_wav_file(filename)
        audio_generator = simulate_stream(buffer)

        decoder.start_utt()
        for chunk in audio_generator:
            decoder.process_raw(chunk, False, False)
        decoder.end_utt()

        hypothesis = decoder.hyp()
        return hypothesis.hypstr

    # Run tests
    for t in TESTCASES:
        print('\naudio file="{0}"    expected text="{1}"'.format(t['filename'], t['text']))
        print('sphinx-streaming-stt: "{}"'.format(
            sphinx_streaming_stt(t['filename'], t['lang'], t['encoding'])
        ))
```

CMU Sphinx presenta una gran diferencia con los demás modelos vistos hasta ahora, con resultados inmediatos, presenta una mayor exactitud al momento de realizar la transcripción del audio en un primer intento., esto lo logra gracias al uso de un sistema decodificador integrado en el modelo con el que le hace un análisis más profundo al audio.

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
sphinx-streaming-stt: "experience proves this"

audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
sphinx-streaming-stt: "why should one hold on the way"

audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
sphinx-streaming-stt: "your paris sufficient i said"
```

El bloque de decodificación tiene tres componentes: el administrador de búsqueda, el lingüista y el anotador acústico, así, estos trabajan en tándem para realizar la decodificación. En las siguientes secciones describimos cada uno de estos componentes con mayor detalle (Lamere y Kwok, 2019).

Mozilla Deepspeech

Esta metodología fue desarrollada aplicando tecnología Machine Learning creado por el equipo del buscador Mozilla, como todos los demás, es una metodología que se encarga de hacer reconocimiento de voz automático(ASR por sus siglas en inglés) que se desarrolló con el propósito de hacer un modelo de uso abierto para desarrolladores; cabe resaltar que, Mozilla no solo se dedica a mejorar y potencializar el navegador web, sino que también tiene como propósito una serie de proyectos

tecnológicos, entre ellos, *DeepSpeech;* reconocido por ellos mismos como un motor de reconocimiento de voz que implementa la arquitectura homónima de reconocimiento (Carmona, 2020); su uso se puede mostrar en la figura 10.

Figura 10. Método Mozilla Deepspeech

3. **Test that it all works**

Examine the output of the last three commands, and you will see results *"experience proof less"*, *"why should one halt on the way"*, and *"your power is sufficient i said"* respectively. You are all set.

```
[ ] !deepspeech --model deepspeech-0.6.0-models/output_graph.pb --lm deepspeech-0.6.0-models/lm.binary --trie ./deepspeech-0.6.0-models/trie --audio ./audio/2830-3980-0043.wa

Loading model from file deepspeech-0.6.0-models/output_graph.pb
TensorFlow: v1.14.0-21-ge77504a
DeepSpeech: v0.6.0-0-g6d43e21
Warning: reading entire model file into memory. Transform model file into an mmapped graph to reduce heap usage.
2020-01-30 00:27:46.675441: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Loaded model in 0.13s.
Loading language model from files deepspeech-0.6.0-models/lm.binary ./deepspeech-0.6.0-models/trie
Loaded language model in 0.000221s.
Running inference.
experience proof less
Inference took 2.418s for 1.975s audio file.
```

Batch API

```
[ ]  import numpy as np

     def deepspeech_batch_stt(filename: str, lang: str, encoding: str) -> str:
         buffer, rate = read_wav_file(filename)
         data16 = np.frombuffer(buffer, dtype=np.int16)
         return model.stt(data16)

     # Run tests
     for t in TESTCASES:
         print('\naudio file="{0}"    expected text="{1}"'.format(t['filename'], t['text']))
         print('deepspeech-batch-stt: "{}"'.format(
             deepspeech_batch_stt(t['filename'], t['lang'], t['encoding'])
         ))
```

Streaming API

```
[ ]  def deepspeech_streaming_stt(filename: str, lang: str, encoding: str) -> str:
         buffer, rate = read_wav_file(filename)
         audio_generator = simulate_stream(buffer)

         # Create stream
         context = model.createStream()

         text = ''
         for chunk in audio_generator:
             data16 = np.frombuffer(chunk, dtype=np.int16)
             # feed stream of chunks
             model.feedAudioContent(context, data16)
             interim_text = model.intermediateDecode(context)
             if interim_text != text:
                 text = interim_text
                 print('inetrim text: {}'.format(text))

         # get final resut and close stream
         text = model.finishStream(context)
         return text
```

Nota. Imágenes de procedimientos y resultados tomadas de Speech Recognition wash Python Satish 2020

El modelo Mozilla entrega resultados que se asemejan al audio, pero presenta en su transcripción fallas gramaticales, esto bien no es un problema que afecte a su capacidad de comprender el audio, pues de la misma forma que azure, puede realizar la transcripción con pocos intentos de análisis, sin embargo, presentando fallos con la exactitud del texto que genera.

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
inetrim text: i
inetrim text: e
inetrim text: experi en
inetrim text: experience pro
inetrim text: experience proof les
deepspeech-streaming-stt: "experience proof less"

audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
inetrim text: i
inetrim text: why shou
inetrim text: why should one
inetrim text: why should one haul
inetrim text: why should one halt
inetrim text: why should one halt on the
deepspeech-streaming-stt: "why should one halt on the way"

audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
inetrim text: i
inetrim text: your p
inetrim text: your power is
inetrim text: your power is suffi
inetrim text: your power is sufficient i
inetrim text: your power is sufficient i said
deepspeech-streaming-stt: "your power is sufficient i said"
```

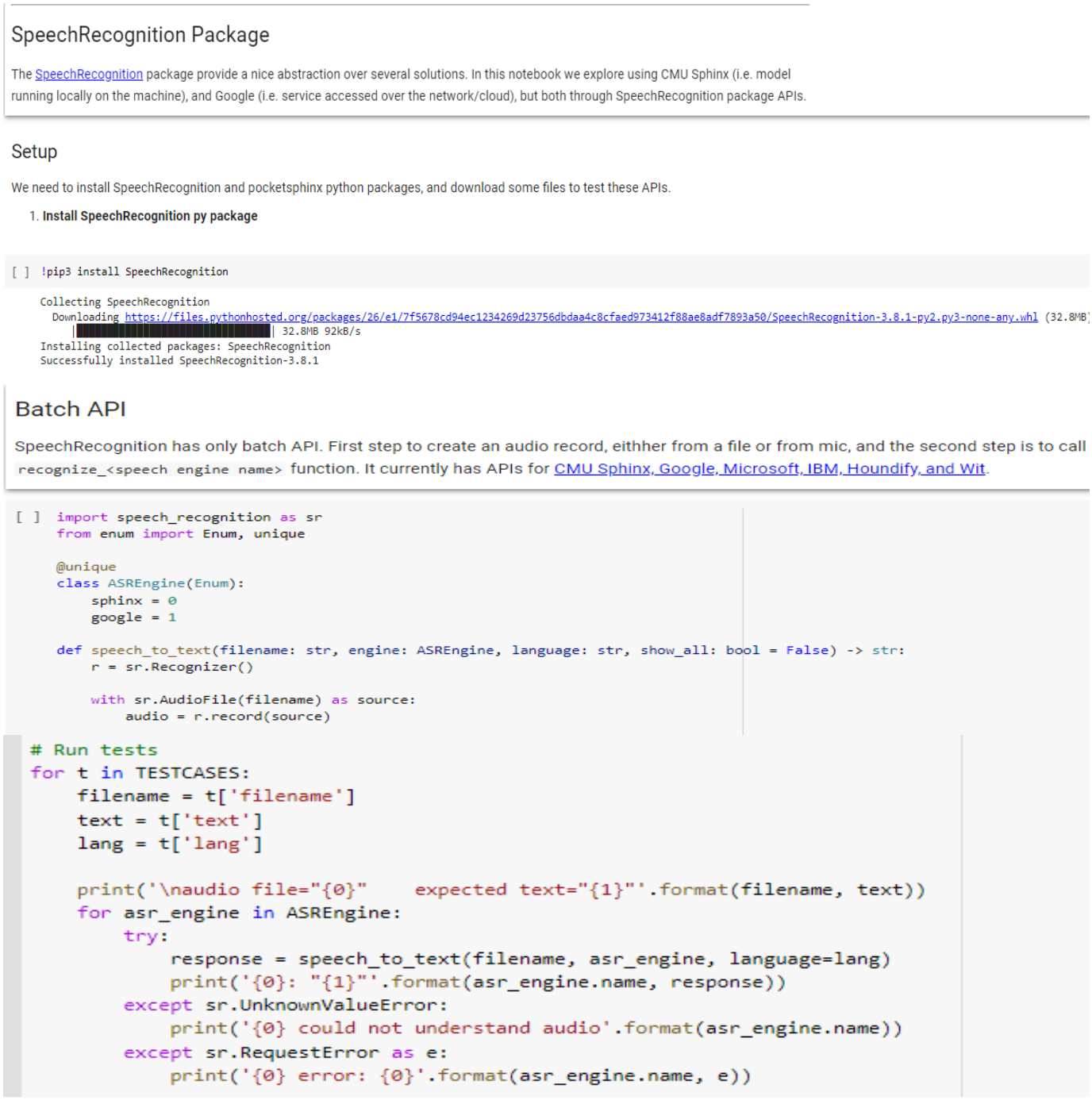
Nota. Imágenes de procedimientos y resultados tomadas de Speech Recognition with Python Satish 2020

El modelo terminado se entrega solo para el idioma inglés, pero para otros idiomas de acuerdo con las instrucciones adjuntas, se puede entrenar el sistema utilizando los datos de voz recopilados por el proyecto Common Voice; esto significa que*, DeepSpeech* es mucho más simple que los sistemas tradicionales pero eso no significa que no se de mayor calidad de reconocimiento, esto porque no utiliza modelos acústicos tradicionales y el concepto de fonemas; por su parte como estrategia para mejorarlo, utiliza un sistema de aprendizaje automático mediante una red neuronal, "que elimina la necesidad de desarrollar componentes separados para modelar diversas desviaciones, como el ruido, el eco y las características del habla" (Carmona, 2020, párr.5).

Python Speech Recognition

El paquete de reconocimiento de voz desarrollado por el propio Python, se encuentra integrado en el programa mismo y puede ser instalado en cualquier entorno donde se corra Python, principalmente haciendo uso del sistema de comandos. A diferencia de los demás métodos este únicamente es capaz de hacer pruebas con la API Batch; su función se muestra en la siguiente figura 11.

Figura 11. Método Mozilla Deepspeech



Nota. Imágenes de procedimientos y resultados tomadas de Speech Recognition wash Python Satish 2020

El modelo de reconocimiento de voz propio de Python ofrece un análisis simple con resultados simples, su exactitud no es tampoco la mejor pero tampoco es la peor, realizando análisis rápidos a los audios cargados recibe el texto del audio y genera el suyo, adicionalmente para este caso se realiza la comparación del texto generado por el modelo con el texto generado por un sistema de google sobre el mismo audio. Pueden reconocer el habla de múltiples hablantes y tienen una enorme cantidad de vocabulario en numerosos idiomas (Can, Martínez, Papadopoulos, Narayanan, 2018).

```
audio file="audio/2830-3980-0043.wav"    expected text="experience proves this"
sphinx: "experience proves that"
google: "experience proves this"

audio file="audio/4507-16021-0012.wav"    expected text="why should one halt on the way"
sphinx: "why should one hold on the way"
google: "why should one halt on the way"

audio file="audio/8455-210777-0068.wav"    expected text="your power is sufficient i said"
sphinx: "your paris official said"
google: "your power is sufficient I said"
```

Nota. Imágenes de procedimientos y resultados tomadas de Speech Recognition with Python Satish 2020

El reconocimiento de voz mediante la *Python Speech Recognition* es muy accesible para él usó en tecnológicas y equipos de ayuda para las personas mayores y las personas con discapacidades físicas y visuales les permite interactuar con productos y servicios de última generación; el reconocimiento de voz en un proyecto de Python es realmente simple, además, se reconoce que, 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 (Amos, 2020).

Finalmente, a continuación, se presenta tabla comparativa de las diferentes metodologías utilizadas para el reconocimiento de voz:

Tabla 2. Comparativo entre los diferentes modelos para el reconocimiento de voz

Metodologías/modelos	Tipo	Precisión	Idioma	Tiempo
Speech Google Cloud	Servicio	Permite la integración a tecnologías de reconocimiento de voz de Google en aplicaciones, el modelo trabaja analizando y ejecutando repetidamente la grabación.	Adaptable a varios idiomas, principalmente inglés.	Tiempo corto para el reconocimiento
Speech Microsoft Azure	Servicio	Está integrado en las funciones de Microsoft azure, y trabaja con métodos Batch y Streaming que Google Cloud, permite el reconocimiento de voz a través de vídeo y audio.	Adaptable a varios idiomas, principalmente inglés.	Requiere de menos cantidades de veces para repetir el audio hasta conseguir coincidir con el reconocimiento.
Speech IBM Watson	Servicio	Permite reconocimiento y lectura rápida y precisa de múltiples idiomas	Usa la transcripción de voz en varios idiomas.	Es una aplicación muy veloz y preciso.
Speech CMU Sphinx	Software	Permite reconocimiento con una mayor exactitud al momento de realizar la transcripción del audio básicamente en un solo intento	Usa la transcripción de voz en varios idiomas.	Es una metodología que obtiene reconocimiento inmediato
Mozilla Deepspeech	Software	Este se encarga de hacer reconocimiento de voz automático, pero presenta en su transcripción fallas gramaticales en el proceso, pero elimina la necesidad de desarrollar	Solo para el idioma inglés	Requiere de varios intentos, por lo que es mas lenta que las anteriores.

		componentes separados.		
Python Speech Recognition	Software	Ofrece un análisis simple con resultados simples, pero su exactitud no es la mejor, realizando análisis rápidos a los audios.	Una un modelo texto generado por un sistema de google, por lo que si idioma es adaptable. Pueden reconocer el habla de múltiples hablantes y tienen una enorme vocabularios en numerosos idiomas.	Es una aplicación muy veloz, pero no con la mayor exactitud.

Nota. Elaboración propia.

CONCLUSIONES

Después de la investigación realizada, se puede concluir que:

- El reconocimiento de voz se fue transformando con el pasar de los años, en sus inicios en la década de 1950 con el desarrollo del primer software capaz de reconocer voz, conocido como *Audrey* no era capaz de reconocer más de 9 dígitos. En 1971 ARPA-SUR tenía la capacidad de reconocer discursos continuos, adaptarse a un diccionario de 10,000 palabras, y poseía un error inferior al 10% y una respuesta en tiempo real. Hoy en día con el reconocimiento de voz de Google se incorpora n230 mil millones de palabras para las consultas reales de los usuarios.

- Los seres humanos cada día están inmersos en la tecnología debido a que facilita muchas de las tareas diarias, siendo la voz la comunicación más natural, el reconocimiento de comandos simples de voz que permite de una forma más fácil de interactuar con los dispositivos.

- El reconocimiento de voz está sujeto a diversos desafíos, tales como: ruido, ecos, acentos, sonidos similares, precisión y habla organizada. Estos factores obstaculizan la correcta ejecución de los sistemas de reconocimiento de voz. No obstante, con los avances tecnológicos cada vez las máquinas son más precisas, esto permite que se puedan adaptar a las necesidades, pero, ante la posibilidad de trabajar con otros programas, puede ayudar a mejorar la precisión.

- Se encontró que, los diferentes sistemas de reconocimiento de voz que existen tienen como objetivo facilitar la interacción de los seres humanos con los dispositivos, no obstante, no todos ofrecen la accesibilidad que se requiere, ya sea porque unos son mas lentos, no están adaptados a todos los idiomas, requieren de softwares adicionales o su capacidad de reconocimiento no es la mejor, ya sea por exactitud o por la presencia de fallas gramaticales.

- Las técnicas descritas permiten una lectura rápida y precisa de la voz, algunas de ellas, teniendo muy poco margen de error, pero otras todavía tienen procesos al analizar audio, algunas otras en su traducción, presentan fallas gramaticales, sobre todo en su capacidad de comprender el audio, no obstante, la mayoría están capacitados para realizar la transcripción con pocos intentos de análisis.

- Todas las metodologías en comparativo tienen beneficios, pero de las mas destacadas son las de Speech IBM Watson, porque permite reconocimiento y lectura rápida y precisa de múltiples idiomas, pero por ejemplo la Mozilla Deepspeech además de necesitar varios intentos para el reconocimiento, solo permite el acceso para el idioma inglés, lo que indica una desventaja en comparación con las otras metodologías.

REFERENCIAS

Amos, D. (2020, 05 12). *The Ultimate Guide to Speech Recognition with Python.* Retriever from Eeal Python: Recuperado de https://realPython.com/Python-speech-recognition/

Anggraini, N., Kurniawan, A., Wardhani, L. K., & Hakim, N. (2020). Speech recognition application for the speech impaired using the android-based google cloud speech Api. *Telkomnika (Telecommunication Computing Electronics and Control)*, *16*(6), 2733-2739.

Can, D., Martínez, V. R., Papadopoulos, P., & Narayanan, S. S. (2018, April). Pykaldi: A python wrapper for kaldi. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5889-5893). IEEE.

Carmona, J. (2020, 11 12). *DeepSpeech: el motor de reconocimiento de voz de Mozilla.* Retrieved from Desde Linux.

Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. (2014). El lenguaje de programación Python. *Ciencias Holguín, 20(2)*, 1-13.

Gomar, J. (2018, 11 25). *Qué es el procesamiento batch o por lotes.* Retrieved from Recuperado de https://www.profesionalreview.com/2018/11/25/que-es-el-procesamiento-batch/

Google Cloud. (2022, 05 12). *Transcripción de voz a texto con la API de Cloud Speech.* Retrieved from Recuperado de https://www.cloudskillsboost.google/focuses/2187?locale=es&parent=catalog

IBM. (2022, 06 14). *IBM Shoebox.* Retrieved from Recuperado de https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html

Intelligent. (2018, 12 5). *Python: el lenguaje de programación más usado por grandes compañías como Google, Facebook o Netflix.* Retrieved from Recuperado de https://itelligent.es/es/que-es-Python/

Kim, J. Y., y Liu. (2019). A comparison of online automatic speech recognition systems and the nonverbal responses to unintelligible speech. *arXiv preprint arXiv:1904.12403*

Lamere, P., y Kwok, P. (2019, April). The CMU SPHINX-4 speech recognition system. In *Ieee intl. conf. on acoustics, speech and signal processing (icassp 2019), hong kong*, 1. p. 2-5.

Lee, C. (2015). Speech Recognition and Production by Machines. In J. Wright, *International Encyclopedia of the Social & Behavioral Sciences (Second Edition)* (pp. 259-263). Oxford: Elsevier.

Mosquera, D. (2019). *Proyecto de diseño de una aplicación para el control de movimientos de un robot ABB mediante reconocimiento de voz.* Catalunya: (Master's thesis, Universitat Politécnica de Catalunya).

Pérez, J. (2020, 11 12). *¿Qué es el Streaming y cómo funciona?* Retrieved from AVAST.

Ramírez, J. (2002). La expresión oral. *Contextos Educativos, 5*, 57-75.

Ruiz, M., & del Mar, M. (2017). *Modelo de privacidad digital en inteligencia ambiental basado en sistemas multiagente.* Madrid: (Doctoral dissertation). Universidad Carlos III de Madrid.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *EEE Trans. Acoustics, Speech, Signal Proc., ASSP-26 (1)*, 43-49.

Soto, M. (2018). *Comparación de técnicas de parametrización espectral para reconocimiento de voz en idioma español.* Zacatecas: Repositorio Universidad Autónoma de Zacatecas. Recuperado de http://ricaxcan.uaz.edu.mx/jspui/handle/20.500.11845/1786.

Striuk, A. (2019). *La voz como fenómeno: palabras aladas, cuerpos sonoros y escucha digital.* Barcelona: (Tesis de Maestría). TFM de Recerca en Art i Disseny. Recuperado de https://diposit.eina.cat/handle/20.500.12082/986.

Woodland, P. (1998). Speech Recognition. *IEE Colloquium (Digest), 499*, 207–214. https://doi.org/10.1007/978-3-540-

71770-6_13.