Práctica 2 - Detección de actividad vocal (VAD)

1st Jaume Colom Hernandez
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
jaumecolomhernandez@gmail.com

2nd Miquel Martinez Blanco
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
mbmiquel@gmail.com

***Abstract*—En esta práctica se implementará un detector de voz (VAD) en situaciones de ruido moderado hecho en C. Posteriormente se comprobará su eficacia y aprenderemos como utilizar la librería `soundfile` para la lectura y escritura de ficheros.**

***Index Terms*—Procesado, audio, voz, C, programación.**

I. Introducción

Los objetivos de esta práctica son: implementar un sistema detector de voz (VAD) mediante un autómata de estados finitos y medir su calidad utilizando una base de datos de audios, aprender a utilizar la libreria `soundfile` para escribir el fichero de audio pero sin ruido en aquellos tramos donde no haya voz y una introducción a programación en `bash` para la automatización de las tareas de la consola de comandos.

II. Descripción del diseño

A. Funciones del programa

El programa principal tiene la tarea de abrir el fichero de audio, leerlo, calcular las métricas, determinar para cada trama analizada si se trata de voz o silencio, exportar la tabla de estados en formato `.vad` y posteriormente en la ampliación grabar un archivo de audio con silencio en aquellos momentos donde el locutor no hablaba.

B. Esquema funcionamiento

1. Abrimos el fichero de audio.
2. Leemos la cabecera del fichero de audio.
3. Leemos un bloque de datos y calculamos los parámetros de esas muestras.
4. Analizando los parámetros se determina si se trata de voz o silencio
5. Repetimos los pasos 3 y 4 hasta que se acabe el fichero `.wav`.
6. Exportamos los tiempos de inicio y final de cada estado en un documento `.vad`.
7. Ampliación: A medida que van llegando los distintos bloques de muestras se va grabando en un fichero `.wav` el audio resultante de eliminar las tramas de ruido sin voz.

III. Aspectos implementación

A. Acceso y loop de lectura del fichero

Para leer el fichero utilizamos la función `sf_read_float(sndfile_in, buffer, frame_size)` la cuál abre el fichero de audio declarado en la variable `sndfile_in` y copia un total de `frame_size` muestras en el `buffer`, donde iremos almacenando todo el archivo. A medida que vamos analizando cada bloque ejecuta la función `vad(vad_data, buffer, silence_time, count, t_up, t_down)` y dentro se llamará a la función `compute_features(float x, int N)` (desarrollada en la primera práctica). Finalmente con los datos de la potencia se determinará si el estado que corresponde a ese bloque es voz o silencio. Por último se exportan las etiquetas que hemos dado a cada bloque en un fichero `.vad` para poderlas comparar en el `wavesurfer` con las etiquetas colocadas manualmente del archivo `.lab`.

B. Implementación ventana de silencio

Debido a que el cambio de un estado a otro es inmediato cuando se supera un cierto `threshold` de valores de potencia, a la hora de decidir el estado de un bloque se producen tramas de silencio muy pequeñas que incluso llegan a marcar las pausas entre palabras de una misma frase. Por ello se ha realizado una modificación respecto al VAD, la cual consiste en añadir la llamada ventana de silencio, esta consiste en una nueva condición para determinar silencio cuando la trama anterior era voz. Para ello se determina que tendrá que haber un determinado número de bloques con la potencia por debajo del `treshold` antes de marcar el primer bloque de silencio. Después de considerar una ventana óptima de 350 ms se ha podido observar como la calidad ha aumentado considerablemente.

C. Treshold de potencia personalizado

Con la finalidad de poder generalizar y utilizar nuestro VAD en todo tipo de grabaciones se optó por añadir una nueva mejora, en cada archivo de audio se calculan los valores de `threshold` a partir de promediar la potencia del silencio de los tres primeros bloques (de 160 muestras cada uno) en cada archivo, de esta manera se consigue personalizar los `threshold` adaptándose a las situaciones de ruido de fondo de cada archivo de audio.

D. Proceso de grabación del archivo sin ruido

Usamos la librería `sndfile` para escribir el fichero. Mediante la función `sf_write_float()` escribimos en el fichero el buffer que le pasemos. Dependiendo del estado, le

pasamos un buffer con el audio con voz o pasamos un buffer con ceros para introducir silencio.

E. Automatización del proceso de evaluación

De manera parecida a como hicimos en la práctica 1 hemos creado un par de scripts que nos permiten hacer pruebas e iterar de manera muy rápida. El script `exec.sh` compila el código, lanza el VAD en todos los ficheros y ejecuta el evaluador. Para ejecutar el vad en todos los ficheros hemos creado el script `run_vad.py` que busca en todas las carpetas los ficheros `.wav` y lanza el vad con los parámetros adecuados.

IV. Evaluación

Para evaluar como de preciso es nuestro programa a la hora de etiquetar tramos de voz y silencio se nos ha proporcionado un ejecutable en formato `perl` encargado de abrir los ficheros `.lab` correspondientes al etiquetado manual, y los `.vad` realizados por nuestro programa. Como resultado a la ejecución se obtienen las métricas de precisión de voz y silencio, es decir, el programa nos da un porcentaje de puntuación dependiendo de cuantos segundos de los que nuestro programa ha etiquetado son correctos. Para la correcta optimización de los valores de `threshold` del programa, se ha tenido en cuenta no solo la precisión si no también el recall. Esto es importante remárcalo ya que si no se podría haber obtenido una precisión muy alta pero haber detectado muy poca cantidad.
Mediante el uso del script de evaluación lanzado en toda la base de datos se ha podido configurar todas las variables del programa.

V. Discusión de los resultados

Finalmente las mejores métricas que conseguimos con el script de evaluación en nuestro audio fueron de una precisión del silencio del 94,32% y una precisión de voz del 85,10%. A continuación hacemos un análisis cualitativo del resultado de nuestro audio.

```
*************** ./audios/ours/output-16k-mono.lab ***************
HIT S.  8.44s/8.95s.    94.32%
HIT V.  13.47s/15.83s.  85.10%
```

Fig. 1. Precisión de silencio y voz de nuestro audio

En referencia a la Figura 1. se pueden observar las etiquetas de la transcripción colocadas por nosotros, correspondientes al archivo `.lab`, y las que ha creado de manera automática nuestro `VAD` y que habían sido guardadas en el archivo de formato `.vad`. Como se puede observar el etiquetado del `VAD` es muy similar al original, pese a ello la principal diferencia radica en la rapidez de cambio, así como el realizado por el `VAD` es inmediato haciendo un cambio para ir de silencio a voz para hacer el proceso inverso, detectar silencio viniendo de voz, hay un cierto `delay` debido a la implementación de la `ventana de silencio`. Aplicar esta medida pese a empeorar el `recall` del silencio consigue evitar perder tramas con voz, lo cuál se ha considerado mucho mas importante.

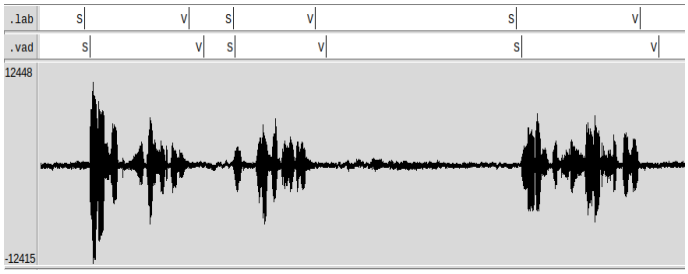

Fig. 2. Transcripción manual y del VAD.

Así como en la parte principal del audio la colocación de etiquetas del `VAD` ha sido muy parecida, cuando se analiza el final se puede observar como se han creado ventanas de voz y silencio mucho más pequeñas y seguidas pese a que nosotros habíamos marcado esas zonas como silencio, tal y como se puede observar en la Figura 2. Esto es debido a que en esta parte del audio hay un gran ruido de fondo y se escucha a otras personas hablar en determinados momentos, y aunque nosotros sabemos que no estaba hablando ninguno de los interlocutores deseados para el `VAD` al superarse el umbral de potencia es condición suficiente para etiquetar como voz.

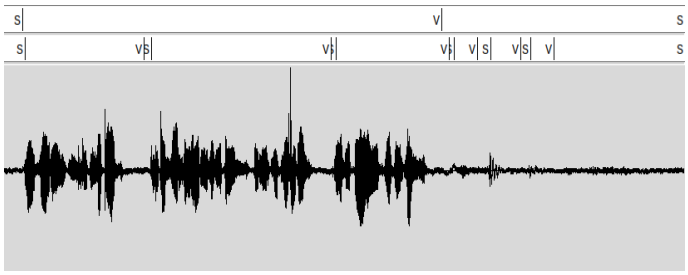

Fig. 3. Transcripción manual y del VAD parte final del audio.

Posteriormente se ha comprobado la eficacia de nuestro VAD con los audios de la base de datos que se nos ha proporcionado. En media la precisión del silencio ha sido del 89,47% y del 75,46% en el caso de la voz. Cualitativamente, pese a no tener precisión del 100%, se obtienen unas detecciones muy buenas y comparando con el original tienen unos resultados muy positivos.

```
**************** Summary ****************
HIT S.   159.38s/178.14s.        89.47%
HIT V.   364.02s/482.43s.        75.46%
```

Fig. 4. Precisión de silencio y voz

Finalmente y como ampliación se ha grabado el mismo archivo de audio insertando los trozos de voz y poniendo silencio (muestras con valor 0) dependiendo del etiquetado del `VAD`. Tal y como se pude observar en la Figura 3. el resultado ha sido muy satisfactorio y al escucharlo no se han detectado cortes en la voz ni perdida de información.

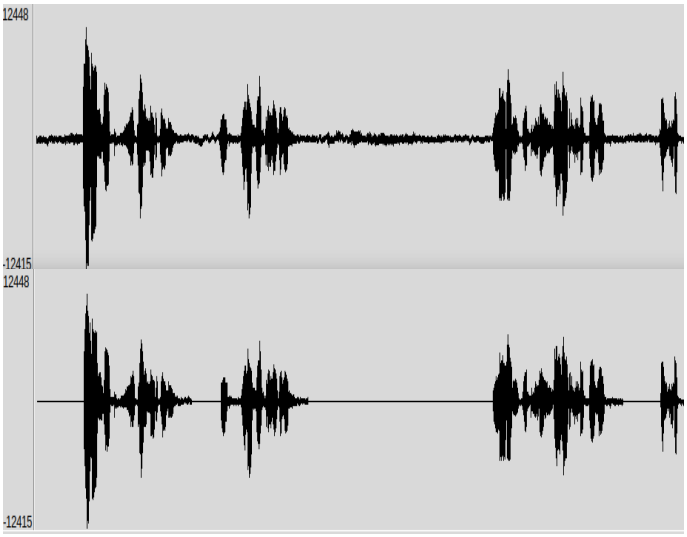

Fig. 5. Audio original arriba y modificado por el VAD abajo.

VI. Conclusiones del trabajo

Mediante esta práctica se conseguido obtener conocimientos básicos sobre la implementación de un autómata de estados finitos en C, pese a no ser un lenguaje orientado a objetos. Hemos aprendido a realizar transcripciones de audios en `Wavesurfer` añadiendo etiquetas de voz o silencio y también se no ha instado a probar e implementar distintas mejoras para nuestro VAD y así conseguir que fuese más eficiente. Por ultimo hemos aprendido a utilizar la librería `soundfile` con la cuál hemos sido capaces de leer y escribir archivos de audio, dando como resultado el `.wav` inicial pero sin los trozos con ruido. Todos los conocimientos adquiridos durante esta practica podrian ser implementados en sistemas de comunicación de audio y así mejorar la calidad de esos servicios de una manera senzilla y computacionalmente poco exigente.

References

[1] C library. libsndfile. Febrero 2019. http://www.mega-nerd.com/libsndfile/api.html
[2] Wavesurfer. Wavesurfer User Manual. Febrero 2019. https://www.speech.kth.se/wavesurfer/man.html
[3] Tutorials Point. C library function fread. Febrero 2019 https://www.tutorialspoint.com/c-standard-library/c-function-fread.htm