**PHÂN ĐOẠN TÍN HIỆU TIẾNG NÓI VÀ KHOẢNG LẶNG DÙNG SHORT-TIME ZERO-CROSSING RATE, SHORT-TIME ENERGY VÀ MAGNITUDE AVERAGE**

**GVHD: TS.Ninh Khánh Duy**

**Ngô Công Long, Nguyễn Thành Long, Dương Văn Bảo**

**Nhóm 1 , lớp HP: 18PFIEV3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Điểm**  (dành cho  GV ghi) | **Bảng phân công nhiệm vụ**  (SV ghi càng cụ thể thì GV càng dễ đặt câu hỏi và cho điểm mỗi SV) | | **Chữ ký của SV**  (mỗi SV ký xác nhận trước khi nộp báo cáo) |
|  | Ngô Công Long (nhóm trưởng) | Đọc tài liệu, cài đặt và viết báo cáo về thuật toán X (tr. 10-11) |  |
|  | Nguyễn Thành Long | Đọc tài liệu, cài đặt và viết báo cáo về thuật toán Y (tr. 6-9) |  |
|  | Dương Văn Bảo | Đọc tài liệu và viết báo cáo về thuật toán Z (tr. 3), thống kê sai số thuật toán T (tr. 20), làm slide và thuyết trình PowerPoint |  |

Lời cam đoan: Chúng tôi, gồm các sinh viên có chữ ký ở trên, cam đoan rằng báo cáo này là do chúng tôi tự viết dựa trên các tài liệu tham khảo liệt kê ở cuối báo cáo. Các số liệu thực nghiệm và mã nguồn chương trình nếu không chỉ dẫn nguồn tham khảo đều do chúng tôi tự làm. Nếu vi phạm thì chúng tôi xin chịu trách nhiệm và tuân theo xử lý của giáo viên hướng dẫn.

TÓM TẮT— Xử lý tiếng nói là sự nghiên cứu tiếng nói của con người dưới dạng tín hiệu, các tín hiệu ngoài đời thực được thể hiện ở dạng số (xử lý tín hiệu số) . Có nhiều loại xử lý tiếng nói trong đó có phân đoạn tiếng nói là việc xác định ranh giới giữa những đoạn tín hiệu có tiếng nói (voiced) và không có tiếng nói (unvoiced) bằng một phương pháp hay một công cụ nào đó. Trong báo cáo này chúng tôi sử dụng phương pháp xử lý dựa vào các đặc trưng của tín hiệu tiếng nói trên miền thời gian sử dụng công cụ Python. Kết quả thực nghiệm của việc xử lý là xóa đi những khoảng lặng của một tín hiệu âm thanh được thu âm trước đó (vừa có tiếng nói vừa có khoảng lặng).

Từ khóa— short-term zero crossing rate, short-time energy, frame-based, magnitude average, normalize magnitude, signal segmentation

Mục lục

[I. ĐẶT VẤN ĐỀ 3](#_Toc57467884)

[II. LÝ THUYẾT XỬ LÝ TÍN HIỆU TIẾNG NÓI VÀ CÁC THUẬT TOÁN 3](#_Toc57467885)

[A. Mối liên hệ giữa tín hiệu liên tục và tín hiệu rời rạc 3](#_Toc57467886)

[B. Short-time Energy (STE) 4](#_Toc57467887)

[C. Magnitude Average (MA) 4](#_Toc57467888)

[D. Zero-crossing Rate + Short-time energy (ZCR + STE) 4](#_Toc57467889)

[E. Sơ đồ khối 5](#_Toc57467890)

[***1.* Short-time Energy 5**](#_Toc57467891)

[***2.* Magnitude Average 6**](#_Toc57467892)

[***3.* Zero-crossing rate kết hợp Short-time Energy 6**](#_Toc57467893)

[III. MÃ CHƯƠNG TRÌNH CÀI ĐẶT CÁC THUẬT TOÁN 7](#_Toc57467894)

[A. Các thư viện sử dụng 7](#_Toc57467895)

[B. Chương trình chính 7](#_Toc57467896)

[C. Hàm chuẩn hóa 8](#_Toc57467897)

[D. Hàm tính STE 8](#_Toc57467898)

[E. Hàm tính ZCR 8](#_Toc57467899)

[F. Hàm tính MA 9](#_Toc57467900)

[G. Hàm tìm biên của phương pháp dùng ngưỡng 9](#_Toc57467901)

[H. Hàm tìm biên của phương pháp kết hợp STE và ZCR 11](#_Toc57467902)

[I. Hàm tìm biên thực trên data 11](#_Toc57467903)

[IV. KẾT QUẢ THỰC NGHIỆM 12](#_Toc57467904)

[A. Hình vẽ 12](#_Toc57467905)

[B. Bảng biểu 21](#_Toc57467906)

[V. KẾT LUẬN 22](#_Toc57467907)

[VI. TÀI LIỆU THAM KHẢO 22](#_Toc57467908)

# ĐẶT VẤN ĐỀ

Trong xử lý tiếng nói, có thể xem như việc phân đoạn tín hiệu là việc phân tích, lọc tín hiệu thành những đoạn tín hiệu có tiếng nói (voiced) và những đoạn tín hiệu không có tiếng nói (unvoiced). Trong báo cáo này chúng tôi sẽ dùng 3 đặc trưng trên miền thời gian của một tín hiệu rời rạc để phân đoạn voiced và unvoiced của một tín hiệu âm thanh cho trước đó là đặc trưng về tốc độ băng qua không (zero crossing rate) và đặc trưng về năng lượng (energy). Và một đặc trưng quan trọng khác là tín hiệu tiếng nói có tính chất biến đổi chậm theo thời gian (tức là các đặc trưng tín hiệu dường như không đổi trong những khoảng thời gian ngắn 10÷30 ms) vì vậy chúng tôi lựa chọn phương pháp xử lý tín hiệu ngắn hạn (short-term processing) tức là xử lý tín hiệu theo từng khung (frame-based processing) kết hợp sử dụng đặc trưng zero-crossing rate thấp thì ở đó có tín hiệu tiếng nói, cao thì ở đó tín hiệu là khoảng lặng, còn đặc trưng energy thì ngược lại ở đoạn có năng lượng thấp thì ở đó là tín hiệu là khoảng lặng, năng lượng cao thì ở đó là là tín hiệu tiếng nói. Với mỗi đặc trưng ta có thể vẽ được một đồ thị trên miền thời gian và việc kết hợp 2 đặc trưng nói trên (chuẩn hóa 2 đồ thị zcr và energy về cùng một thang biên độ - normalized magnitude) ta được điểm giao giữa hai đồ thị đó cũng chính là biên của đoạn tiếng nói, đó cũng chính là mục tiêu của việc phân đoạn tín hiệu tiếng nói. Còn một đặc trưng nữa là Magnitude Average (trung bình biên độ) cũng dùng để phân loại khoẳng lặng và tiếng nói.

Báo cáo có bố cục như sau. Phần II trình bày về lý thuyết phục vụ cho việc xử lý tín hiệu tiếng nói, lưu đồ thuật toán. Phần III trình bày mã nguồn chương trình và các thuật toán. Phần IV trình bày kết quả thực nghiệm của việc xử lý tín hiệu tiếng nói với 6 file âm thanh cho trước.

# LÝ THUYẾT XỬ LÝ TÍN HIỆU TIẾNG NÓI VÀ CÁC THUẬT TOÁN

## Mối liên hệ giữa tín hiệu liên tục và tín hiệu rời rạc

Xử lý tín hiệu số là việc biểu diễn tín hiệu ngoài đời thực bằng dưới dạng số [1] đó cũng là công việc bắt buộc để chúng ta có thể thực hiện các bước tiếp theo để xử lý một tín hiệu. Vì vậy chúng ta cần tìm ra một mối liên hệ nào đó giữa tín hiệu liên tục và tín hiệu rời rạc (tín hiệu nhận được nhờ việc lấy mẫu) hay nói cách khác là tìm ra một công thức để biểu diễn mối quan hệ giữa biến thời gian liên tục và biến thời gian rời rạc.

xa(0)

xa(t)

t(s)

0 Ts 2Ts 3Ts 4Ts

Lấy mẫu với chu kỳ Ts



x[n]

x[0]

n

0 1 2 3 4

1. Minh họa mỗi liên hệ giữa TH liên tục và TH rời rạc thông qua việc lấy mẫu.

Từ hình vẽ trên ta có được mối quan hệ như sau:

Từ công thức trên ta có được mối liên hệ giữa số lượng mẫu N và độ dài D đơn vị (s) của một tín hiệu:

## Short-time Energy (STE)

***Short-time Energy là gì ?***

Việc tính năng lượng của một tín hiệu là một tính toán thường được sử dụng trong xử lý tín hiệu. Cũng giống như zero-crossing rate, năng lượng của tín hiệu (energy) cũng là một đặc trưng của trên miền thời gian của tín hiệu rời rạc.

Năng lượng của một tín hiệu được phát biểu như sau: “Năng lượng của một tín hiệu bằng tổng bình phương biên độ tại các mẫu.”

Năng lượng là một đặc trưng của tín hiệu tiếng nói mà tín hiệu tiếng nói có tính biến đổi chậm theo thời gian, tức là đặc trưng năng lượng cũng gần như không đổi theo thời gian trong những khoảng thời gian ngắn hạn từ 10÷30 ms, vì vậy ta có thể áp dụng kỹ thuật xử lý tín hiệu ngắn hạn. Bằng cách chia tín hiệu thành nhiều khung có thời gian bằng nhau (10÷30 ms), do đó ta có công thức hàm năng lượng ngắn hạn như sau:

Với m là chỉ số khung thứ m và n ∈ [N1m, N2m] với N1, N2 là chỉ số mẫu bắt đầu và kết thúc của khung thứ m.

***Tại sao lại dùng Short-time Energy để phân đoạn tín hiệu ?***

Theo đặt trưng của tín hiệu tiếng nói ở những vị trí có năng lượng cao thì ở đó là tín hiệu hiệu tiếng nói, ngược lại ở những vị trí có năng lượng thấp thì ở đó là tín hiệu khoảng lặng. Hình dưới đây gồm 2 đồ thị biễu diễn 2 hàm E và ZCR theo phương pháp short-time processing.

## Magnitude Average (MA)

***Magnitude Average (MA) là gì?***

Tương tự với STE, MA cũng có những đặc trưng tương tự, đều có thể dùng để phân biệt rõ ràng tiếng nói và khoảng lặng. MA của mỗi khung được tính bằng tổng các trị tuyệt đối của biên độ của tất cả những phần tư trong khung, vì vậy ta có công thức như sau:

(Với m1 là điểm đầu của khung chứa phần tử thứ n, m2 là điểm cuối của khung chứ phần tử thứ n)

***Tại sao lại dùng Magnitude Average để phân đoạn tín hiệu?***

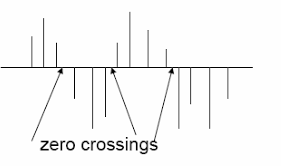
Khi xử lý tín hiệu bằng phương pháp MA, những khoảng tiếng nói có MA lớn hơn rất nhiều so với MA của những khoảng lặng. Vì thế chúng tôi dùng đặc trưng này để phân đoạn tín hiệu thành tiếng nói và khoảng lặng.

## Zero-crossing Rate + Short-time energy (ZCR + STE)

***Zero-crossing Rate là gì?***

Đầu tiên ta nói về zero-crossing rate:

Tốc độ băng qua không (ZRC) là một trong những đặc trưng trên miền thời gian của tín hiệu rời rạc. Zero- crossing rate là một thông số cho biết số lần mà biên độ tín hiệu băng qua điểm zero trong một khoảng thời gian nhất định.



1. Định nghĩa tốc độ băng qua không (zero-crossing rate)

***Tại sao lại kết hợp 2 phương pháp ZCR và STE để phân đoạn tín hiệu?***

Theo đặc trưng của tín hiệu, ở những đoạn tín hiệu có zero-crossing rate cao thì ở đó tín hiệu là khoảng lặng, ngược lại ở những vị trí có zero-crossing rate thấp thì ở đó tín hiệu là có tiếng nói. Vì tín hiệu tiếng nói có tính chất biến đổi chậm theo thời gian, có nghĩa là các đặc trưng của tín hiệu gần như không thay đổi theo những khoảng thời gian ngắn hạn từ 10÷30 ms. Áp dụng vào xử lý tiếng nói ta sử dụng phương pháp xử lý ngắn hạn (short time processing) tức là chia tín hiệu ban đầu thành những khung bằng nhau (frame-based processing), mỗi khung từ 10÷30 ms. Vì vậy ta có công thức của hàm ZCR:

Với x[n] là vector biên độ lấy được từ âm thanh được thu âm trước. Sử dụng ngôn ngữ Python và một số thư viện thích hợp như: numpy và matplotlib.pyplot ta có thể xác định vector x bằng cách dùng hàm read(“file path”), m là chỉ số khung thứ m và n [∈](https://en.wikipedia.org/wiki/%E2%88%88) [N1m, N2m] với N1, N2 là chỉ số mẫu bắt đầu và kết thúc của khung thứ m.

Vì tính chất của hàm STE và ZCR có tính chất ngược nhau nên ta có thể kết hợp 2 phương pháp này để phân đoạn tín hiệu thành những khoảng lặng và tiếng nói.

## Sơ đồ khối

### Short-time Energy

Chia khung tín hiệu

Tính tổng năng lượng ngắn hạn của mỗi khung

Chuẩn hóa đồ thị Short-Time Energy

Chọn ngưỡng cho trước

Phân đoạn tiếng nói và khoảng lặng

### Magnitude Average

Chia khung tín hiệu

Tính tổng biên độ trung bình ngắn hạn của mỗi khung

Chuẩn hóa đồ thị Magnitude Average

Chọn ngưỡng cho trước

Phân đoạn tiếng nói và khoảng lặng

### Zero-crossing rate kết hợp Short-time Energy

Dùng hàm ZCR tính số lần băng qua 0 trong mỗi khung

Dùng hàm STE tính năng lượng trong mỗi khung

Chuẩn hóa đồ thị Short-time Energy và Zero-Crossing Rate về cùng 1 tọa độ

Tìm số giao điểm và vị trí giao điểm giữa 2 đồ thị

Phân đoạn tiếng nói và khoảng lặng

# MÃ CHƯƠNG TRÌNH CÀI ĐẶT CÁC THUẬT TOÁN

## Các thư viện sử dụng

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from scipy.io.wavfile import read |

## Chương trình chính

|  |
| --- |
| # đọc file bằng hàm read của scipy  Fs, data = read('./Resources/TinHieuMau/studio\_male.wav')  # tính năng lượng ngắn hạn STE  E = CalculateSTE(Fs, data)  # chuẩn hóa STE  E = Normalize(E, min(E), max(E))  # tìm ZCR  zcr = CalculateZCR(Fs, data)  # chuẩn hóa ZCR  zcr = Normalize(zcr, min(zcr), max(zcr))  # tìm MA  MA = CalculateMA(Fs, data)  # chuẩn hóa MA  MA = Normalize(MA, min(MA), max(MA))  # tìm biên của phương pháp dùng STE và ngưỡng  EdgesE = GetEdges(E, 0.02)  # tìm biên của phương pháp dùng MA và ngưỡng  EdgesMA = GetEdges(MA, 0.1)  # tìm biên của phương pháp dùng STE và ZCR  EdgesE\_ZCR = GetEdgesE\_ZCR(E, zcr)  # Đây là block code để chuyển các biên về đơn vị giây  EdgesMAs = []  # Tạo LIST EdgesMAs để chưa các biên của MA  for i in range(len(GetRealEdges(EdgesMA, Fs))):  # Duyệt qua hết các biên của MA      # Chia các phần tử cho Fs để đưa vào LIST      EdgesMAs.append(GetRealEdges(EdgesMA, Fs)[i]/Fs)  print("MA: ", EdgesMAs)  EdgesEs = []  # Tạo LIST EdgesMAs để chưa các biên của E  for i in range(len(GetRealEdges(EdgesE, Fs))):  # Duyệt qua hết các biên của E      # Chia các phần tử cho Fs để đưa vào LIST      EdgesEs.append(GetRealEdges(EdgesE, Fs)[i]/Fs)  print("E: ", EdgesEs)  EdgesZCRSTEs = []  # Tạo LIST EdgesMAs để chưa các biên của ZCR + STE  for i in range(len(GetRealEdges(EdgesE\_ZCR, Fs))):  # Duyệt qua hết các biên của ZCR + STE      # Chia các phần tử cho Fs để đưa vào LIST      EdgesZCRSTEs.append(GetRealEdges(EdgesE\_ZCR, Fs)[i]/Fs)  print("ZCR + STE: ", EdgesZCRSTEs)  # ------------------------------------------------ |

## Hàm chuẩn hóa

|  |
| --- |
| def Normalize(data, min, max):  # Chuẩn hóa data về 0,1      res = []    # Tạo LIST res rỗng để chứa kết quả      for i in range(0, len(data)):   # Cho i chạy hết qua data          res.append((data[i]-min)/(max-min))  # Đẩy kết quả vào LIST res      return res  # Trả về LIST res cho hàm |

## Hàm tính STE

|  |
| --- |
| def CalculateSTE(Fs, data): # Hàm tính STE      dur = len(data) / Fs  # Độ dài của tín hiệu âm thanh      t = np.arange(0, dur, 0.02)  # Chia khoảng 0.02s bằng hàm arange của numpy      # Tạo mảng E toàn 0 với độ dài của t với hàm zeros của numpy với kiểu dữ liệu là float64      E = np.zeros(len(t), dtype=np.float64)      n1 = 0  # Biến n1 phụ để chạy qua mỗi khung      # Xác định tương đối 1 khoảng chia gặp bao nhiêu thơi gian lấy mẫu (= 0.02/T)      x = int(0.02\*Fs)  # x là độ dài 1 khung      for n in range(len(E)):  # Cho biến n chạy hết qua E          while ((n\*x + n1) < len(data)):  # Đảm bảo vẫn ở trong khoảng của data              E[n] += data[n\*x + n1]\*\*2   # Công thức của năng lượng ngắn hạn              n1 += 1  # Tăng biến n1 lên 1 đơn vị              if (n1 == x):   # Nếu n1 bằng x thì đang duyệt đến vị trí  cuối khung                  n1 = 0  # Đưa n1 = 0 để đến khung tiếp theo                  break   # break để dừng      return E    # Trả lại E cho hàm |

## Hàm tính ZCR

|  |
| --- |
| def CalculateZCR(Fs, data): # Hàm tính ZCR      dur = len(data) / Fs   # Độ dài của tín hiệu âm thanh      t = np.arange(0, dur, 0.02)  # Chia khoảng 0.02s bằng hàm arange của numpy      # Tạo mảng ZCR toàn 0 với độ dài của t với hàm zeros của numpy với kiểu dữ liệu là float64      ZCR = np.zeros(len(t), dtype=np.float64)      n1 = 0      # Xác định tương đối 1 khoảng chia gặp bao nhiêu thơi gian lấy mẫu ( = 0.02/T)      x = int(0.02\*Fs)  # x là độ dài 1 khung      for n in range(len(ZCR)):  # Cho biến n chạy hết qua E          while ((n\*x + n1) < len(data)-1):  # Đảm bảo vẫn ở trong khoảng của data              # Công thức tính của ZCR với hàm lấy dấu              if (data[n\*x + n1] \* data[n\*x + n1 + 1] > 0):                  ZCR[n] += 1  # Tăng lên 1 đơn vị của khung n              n1 += 1  # Tăng biến n1 lên 1 đơn vị              if (n1 == x):  # Nếu n1 bằng x thì đang duyệt đến vị trí cuối khung                  n1 = 0  # Đưa n1 = 0 để đến khung tiếp theo                  break  # break để dừng      return ZCR  # Trả lại ZCR cho hàm |

## Hàm tính MA

|  |
| --- |
| def CalculateMA(Fs, data): #Hàm tính MA      dur = len(data) / Fs  # Độ dài của tín hiệu âm thanh      t = np.arange(0, dur, 0.02)  # Chia khoảng 0.02s bằng hàm arange của numpy      # Tạo mảng MA toàn 0 với độ dài của t với hàm zeros của numpy với kiểu dữ liệu là float64      MA = np.zeros(len(t))      n1 = 0      # Xác định tương đối 1 khoảng chia gặp bao nhiêu thơi gian lấy mẫu ( = 0.02/T)      x = int(0.02\*Fs)  # x là độ dài 1 khung      for n in range(len(MA)):  # Cho biến n chạy hết qua E          while ((n\*x + n1) < len(data)):  # Đảm bảo vẫn ở trong khoảng của data              MA[n] += abs(data[n\*x + n1])     # Công thức tính của MA              n1 += 1  # Tăng biến n1 lên 1 đơn vị              if (n1 == x):  # Nếu n1 bằng x thì đang duyệt đến vị trí cuối khung                  n1 = 0  # Đưa n1 = 0 để đến khung tiếp theo                  break  # break để dừng      return MA  # Trả lại MA cho hàm |

## Hàm tìm biên của phương pháp dùng ngưỡng

|  |
| --- |
| # Hàm tìm biên của các phương pháp dùng ngưỡng  def GetEdges(data, threshold):  # Hàm tìm biên tham số vào là data và ngưỡng đã khảo sát      # tạo mảng altData toàn giá trị 0 với độ dài bằng độ dài data      altData = np.zeros(len(data))      for i in range(0, len(data)):  # Cho biên i duyệt qua hết data          if data[i] <= threshold:  # Nếu biên độ tại i nhỏ hơn ngưỡng ta đặt biên độ tại đó bằng 0              altData[i] = 0          else:   # Nếu không trả lại giá trị nguyên vẹn              altData[i] = data[i]  # Tiếp theo dùng LIST altData để xử lý tìm biên      res = []    # Tạo mảng res      for i in range(0, len(altData)-1):  # Cho i chạy hết qua data          if(altData[i+1] > 0.0 and altData[i] == 0.0):  # Lấy biên bên trái              res.append(i)   # Cho biên vào mảng res          elif(altData[i-1] > 0.0 and altData[i] == 0.0):  # Lấy biên bên phải              res.append(i)   # Cho biên vào mảng res  # Tại đây khi đã có các biên ta cần lọc những biên bị sai do sự nhiễu giữa các tiếng nói  # Đây là dãy code để lọc những biên ở quá sát nhau trong khoảng 12 đoạn      u = 0       # Cho biến phụ u = 0      while(u < len(res)):    # Điều kiện để u trong khoảng độ dài là len(res)          temp = res[u]   # Cho biến phụ temp bằng res[u]          for k in range(1, 13):  # Cho biến k chạy từ 1 đến 12              if temp + k in res:  # Nếu temp + k có trong res thì xóa temp + k bằng hàm .remove                  res.remove(temp + k)              if temp - k in res:  # Nếu temp - k có trong res thì xóa temp - k bằng hàm .remove                  res.remove(temp - k)          u += 1  # Tăng biến u lên 1 đơn vị  # -------------------------------------------------------  # Sau khi đã lọc những khoảng sát nhau thì các tiếng nói vẫn còn dư những biên còn lại chứ kho  # Block code sau sẽ loại những biên đó      temp1 = []  # Tạo mảng temp1 rỗng để chứa những phần tử cần xóa      for i in range(0, len(res)):    # Cho biến i duyệt qua res          for k in range(1, 13):  # Cho biến k              if(res[i] - k > 1 and res[i] + k < len(altData) - 1):                  if altData[res[i] + k] > 0 and altData[res[i] - k] > 0:                      temp1.append(res[i])                      break  # Tiến hành xóa những biên lỗi đi bằng hàm .remove      for i in range(0, len(temp1)):          res.remove(temp1[i])  # -----------------------------------------------------------      return res |

## Hàm tìm biên của phương pháp kết hợp STE và ZCR

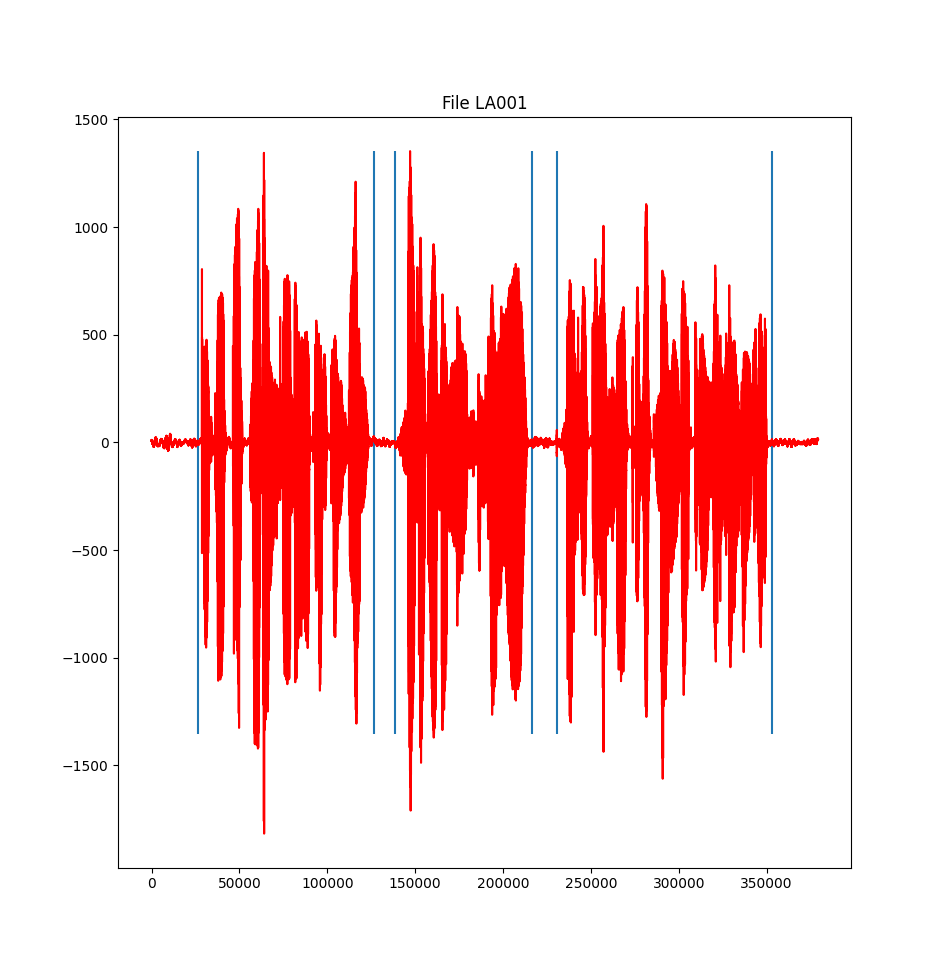
|  |
| --- |
| # hàm tìm biên của phương pháp kết hợp STE và ZCR  def GetEdgesE\_ZCR(E, ZCR):      res = []  # Tạo mảng rỗng để đánh dấu biên      check = True  # Khai báo biến kiểm tra      for i in range(1, len(E)):  # Duyệt E          if (E[i] > zcr[i] and E[i-1] < zcr[i-1]):  # Nếu thỏa mãn điều kiện là biên trái              for k in range(1, 10):  # kiểm tra 10 mẫu tiếp theo                  if(i + k < len(E)):  # Để đảm bảo i+k không bị vượt quá độ dài của E                      # Nếu có bất kì phần tử nào không thỏa mãn trong 10 mẫu, check = false và dừng kiểm tra                      if (E[i + k] < zcr[i+k]):                          check = False                          break              if (check):  # Nếu toàn bộ 10 phần tử đều thỏa mãn, đánh dấu lại vị trí của nó, cho vào LIST res                  res.append(i)              check = True  # Trả lại giá trị True cho biến kiểm tra và tiếp tục duyệt          elif (E[i - 1] > zcr[i-1] and E[i] < zcr[i]):  # Nếu điều kiện thỏa mãn là biên phải              for k in range(1, 10):  # kiểm tra 10 mẫu tiếp theo                  if(i + k < len(E)):  # Để đảm bảo i+k không bị vượt quá độ dài của E                      # Nếu có bất kì phần tử nào không thỏa mãn trong 10 mẫu, check = false và dừng kiểm tra                      if (E[i + k] > zcr[i+k]):                          check = False                          break              if (check):  # Nếu toàn bộ 10 phần tử đều thỏa mãn, đánh dấu lại vị trí của nó, cho vào LIST res                  res.append(i)              check = True  # Trả lại giá trị True cho biến kiểm tra và tiếp tục duyệt      return res |

## Hàm tìm biên thực trên data

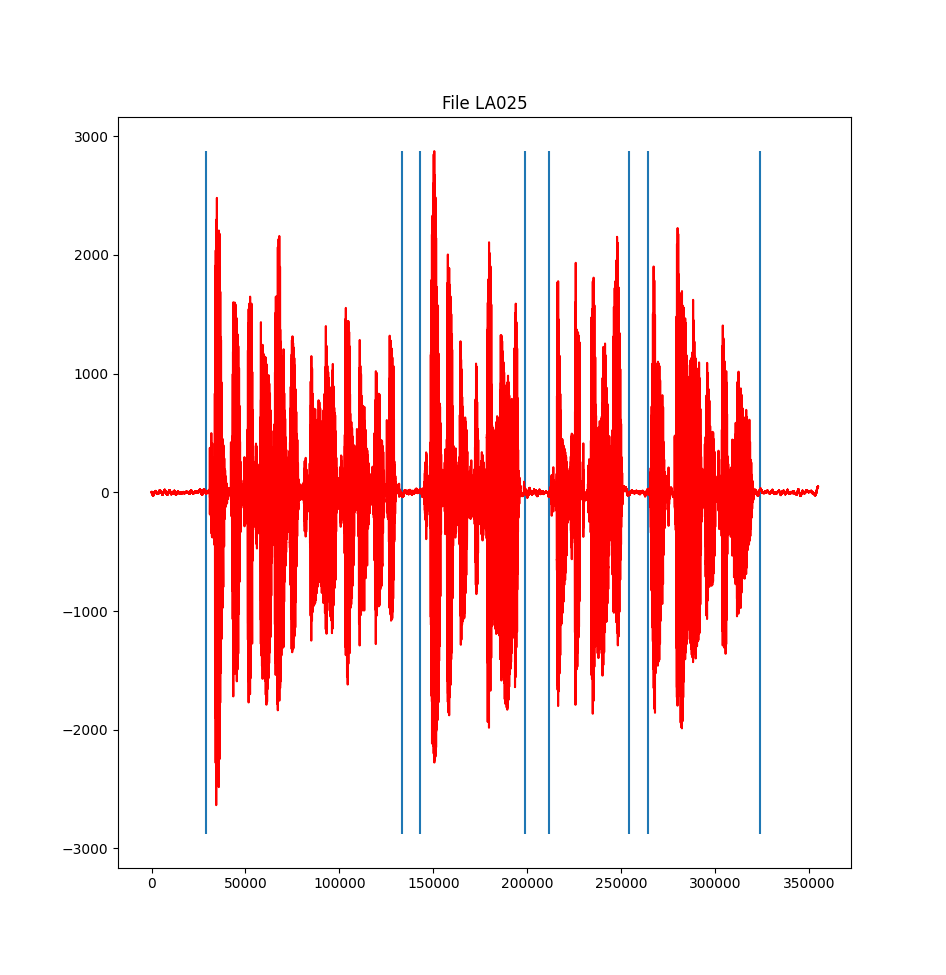
|  |
| --- |
| # hàm tìm biên thực trên data  # Input là 2 tham số gồm LIST biên được xác định trên biểu đồ của các phương pháp, và tần số lấy mẫu Fs  def GetRealEdges(edges, Fs):      res = []  # Khởi tạo mảng để lưu các biên trên data      for i in range(len(edges)):  # Duyệt các phần tử trong edges          # Theo định nghĩa của tần số lấy mẫu với độ dài mỗi khung = 0.02          res.append(edges[i]\*int(0.02\*Fs))      return res |

# KẾT QUẢ THỰC NGHIỆM

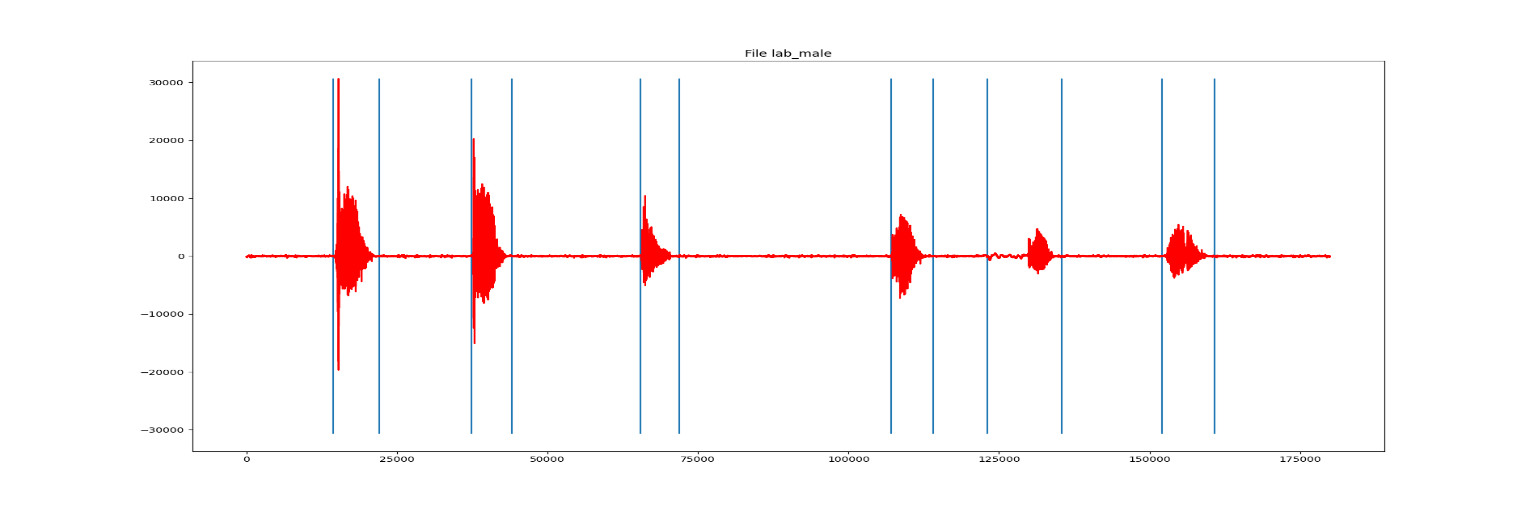
## Hình vẽ



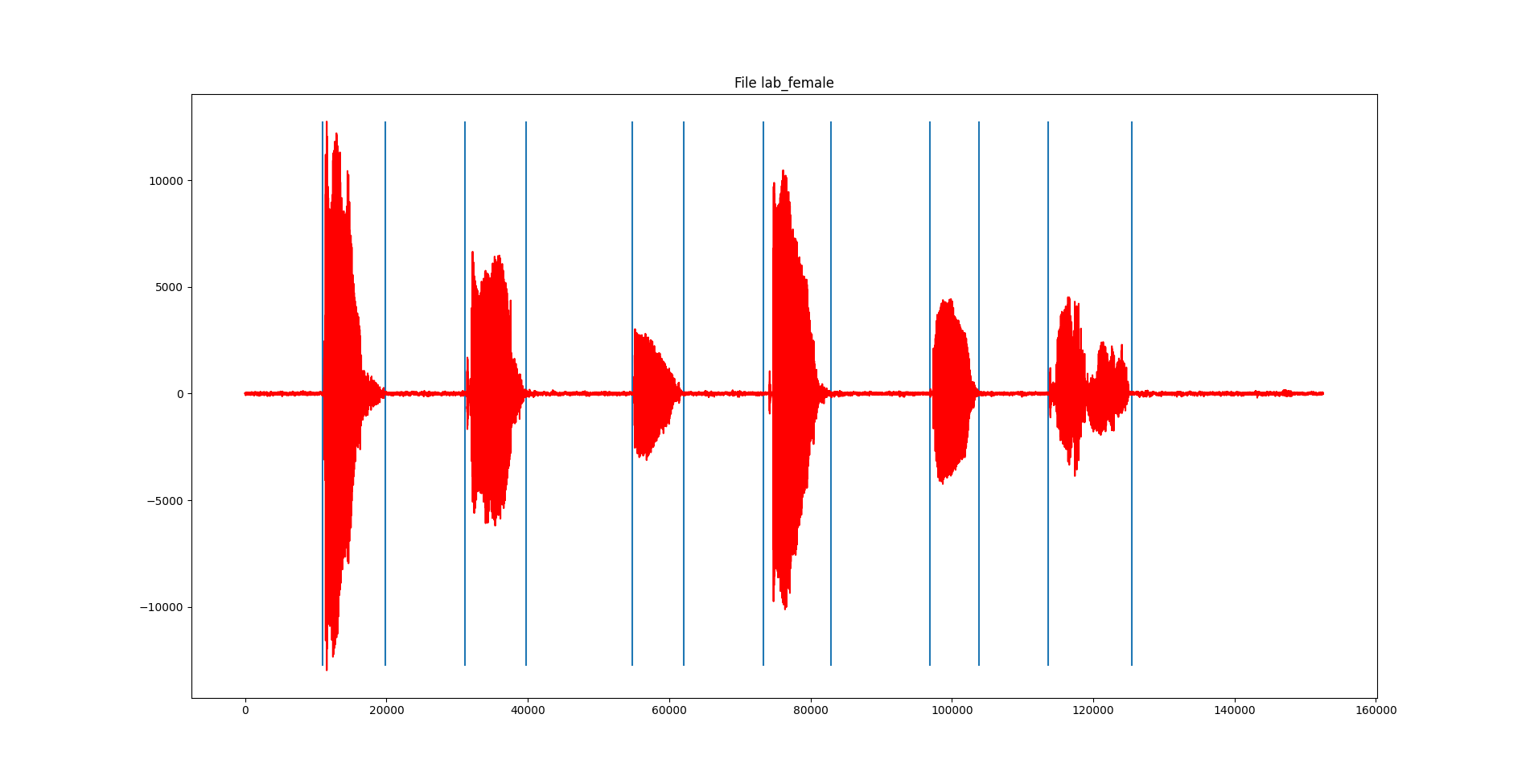
1. Biên thời gian xác định thủ công của flie LA001.wav



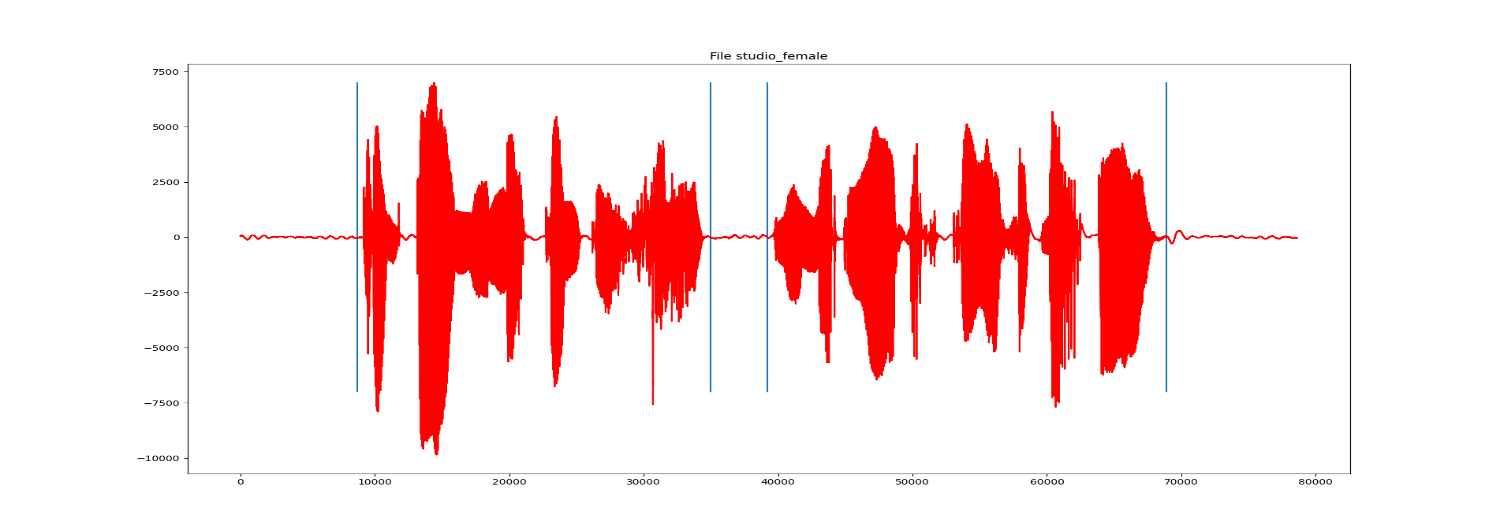
1. Biên thời gian xác định thủ công của flie LA025.wav



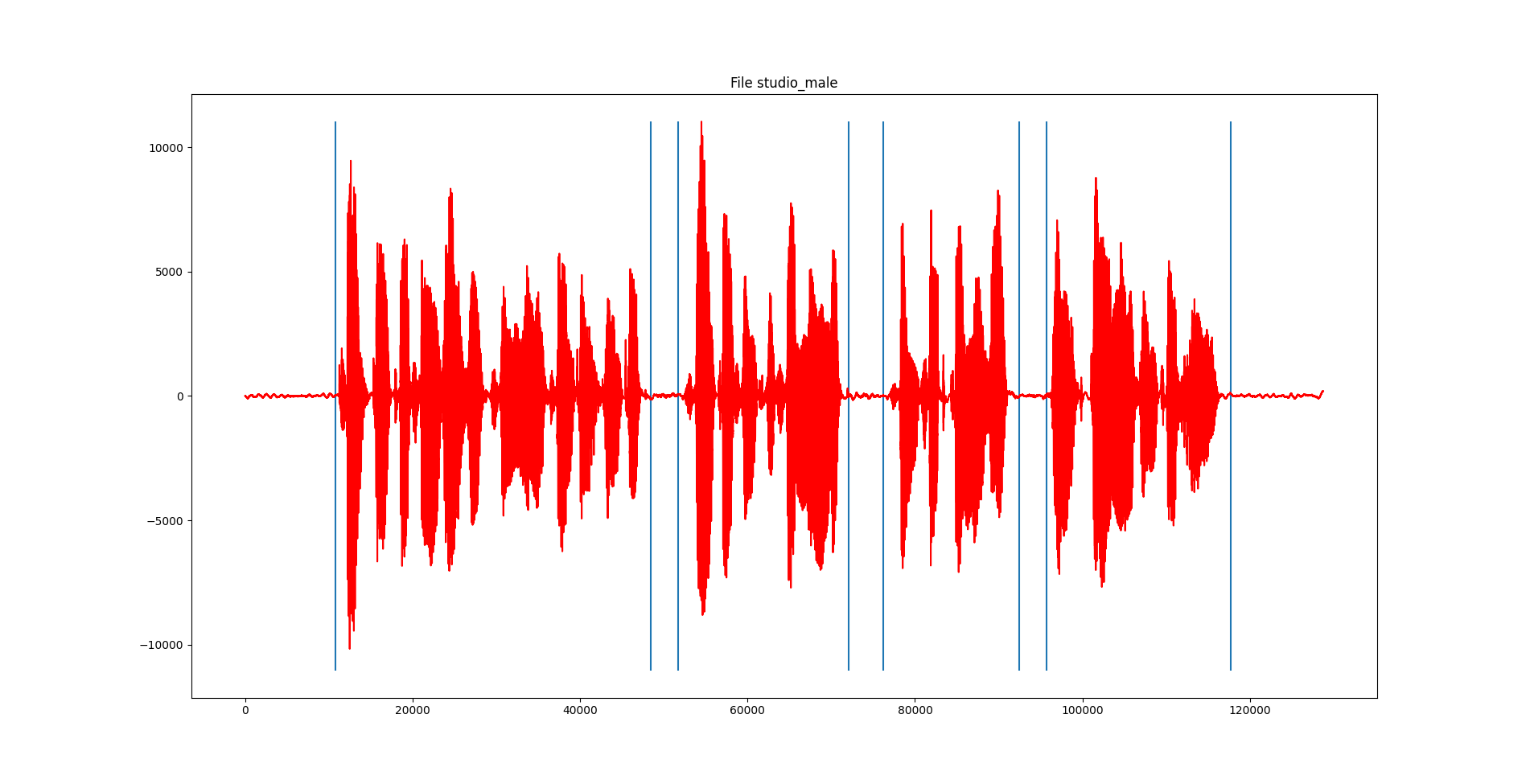
1. Biên thời gian xác định thủ công của flie lab\_male.wav



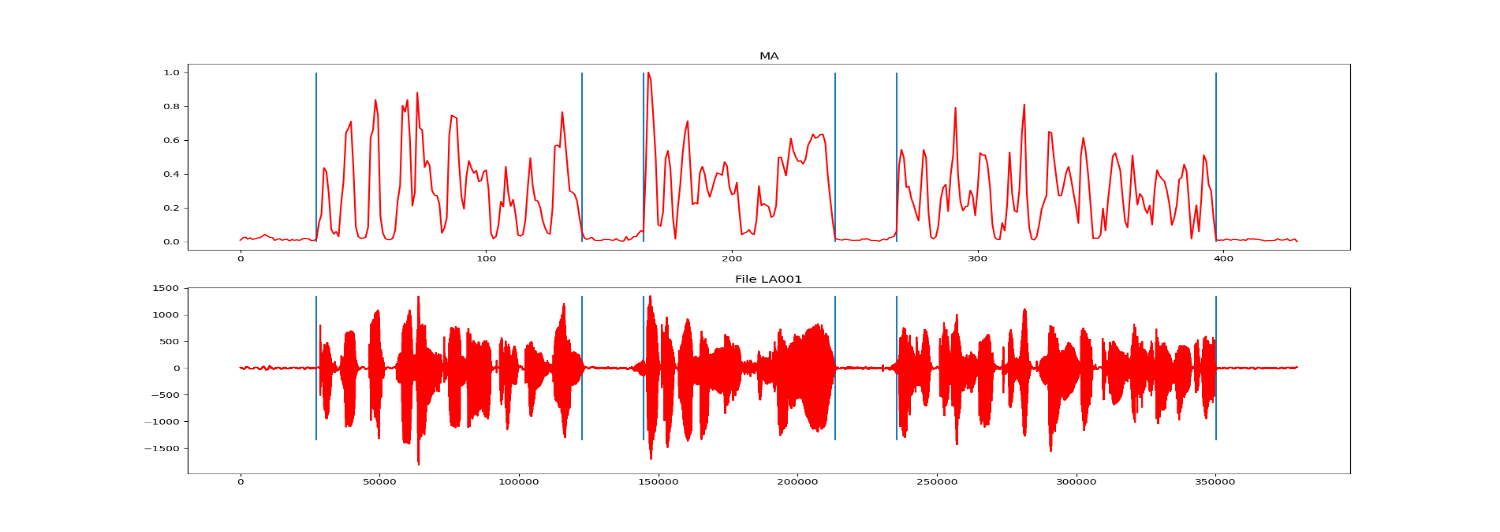
1. Biên thời gian xác định thủ công của flie lab\_female.wav



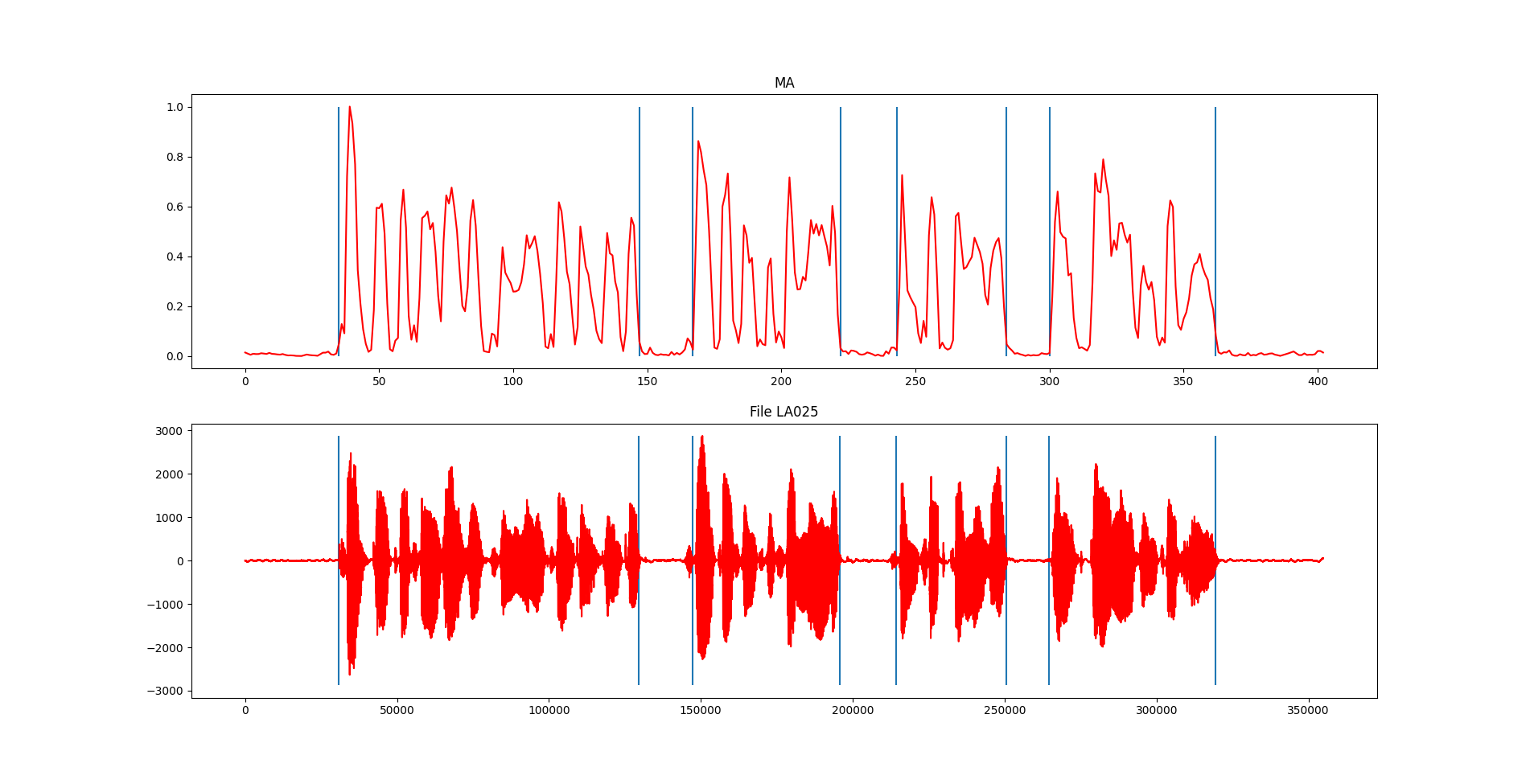
1. Biên thời gian xác định thủ công của flie studio\_female.wav



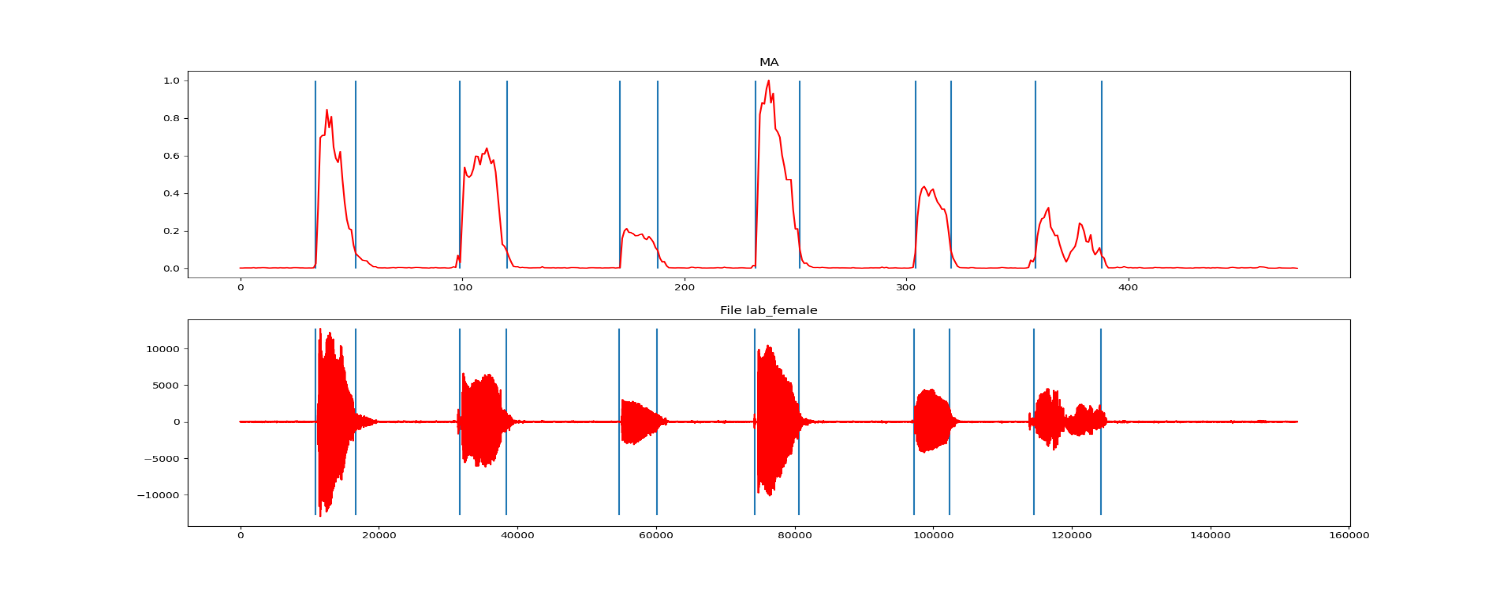
1. Biên thời gian xác định thủ công của flie studio\_male.wav



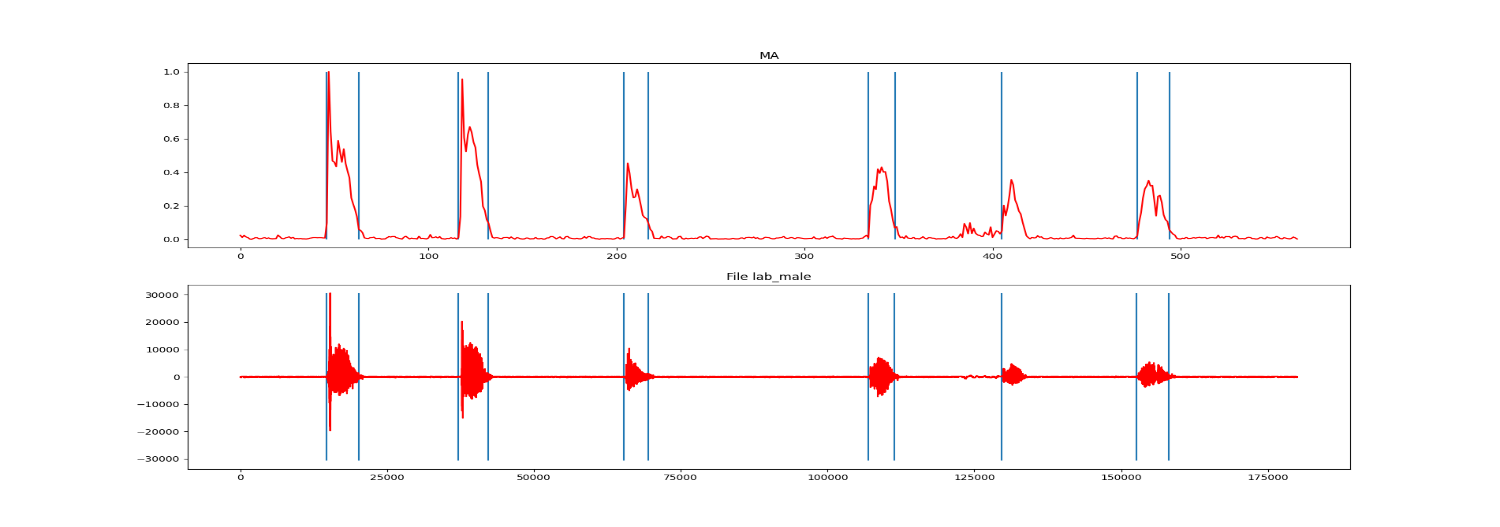
1. MA LA001



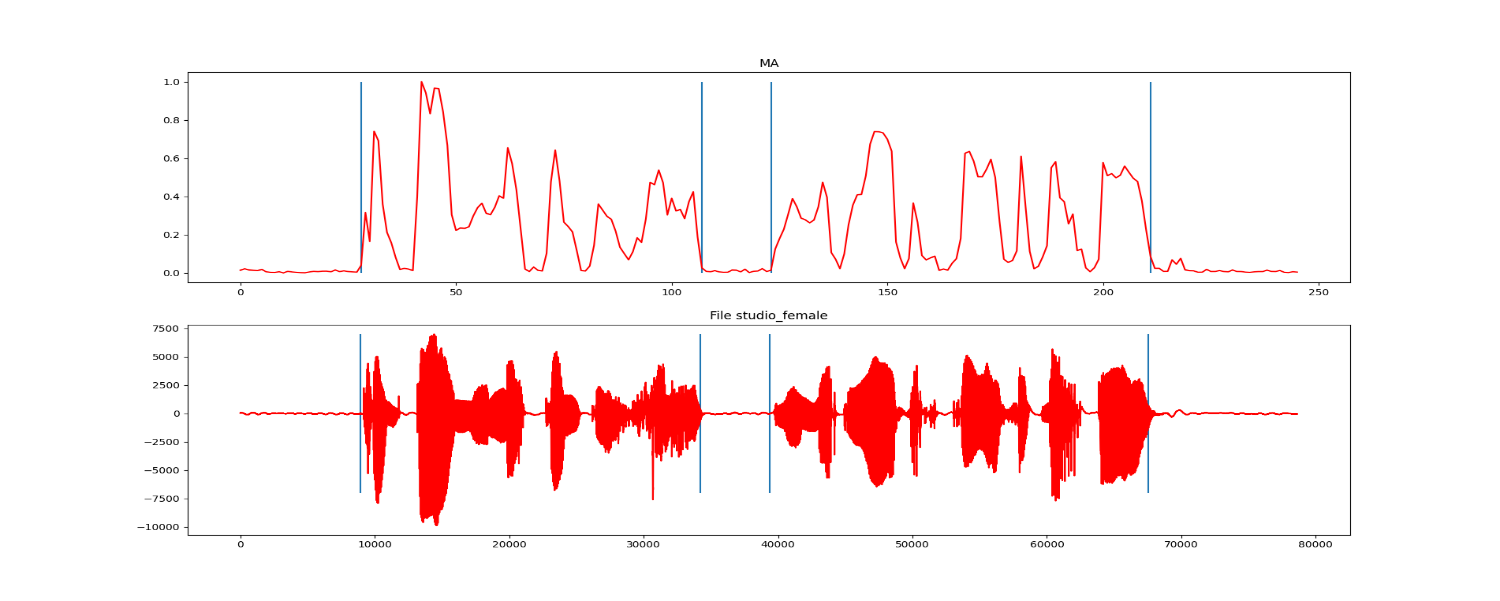
1. MA LA025



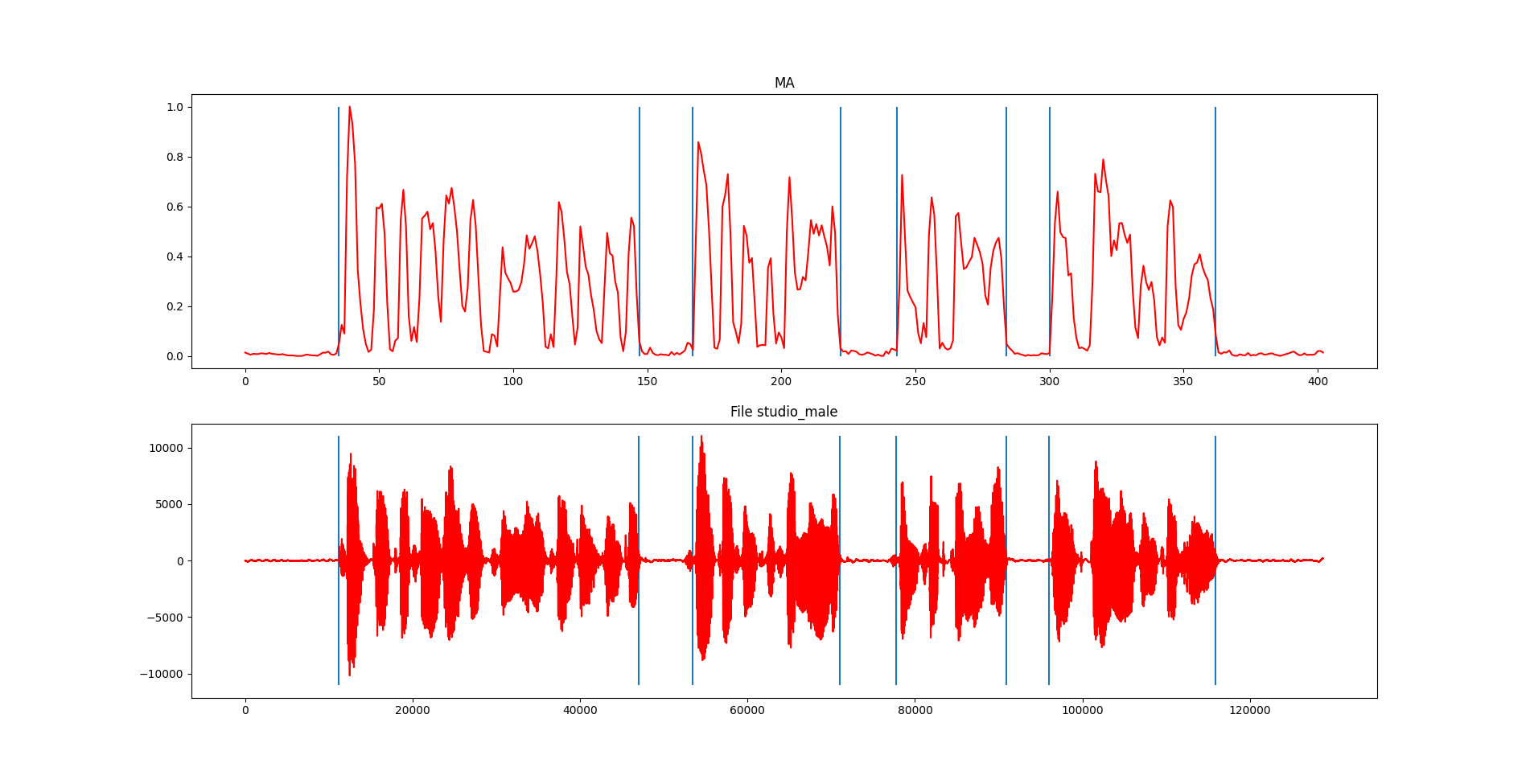
1. MA lab\_fenale



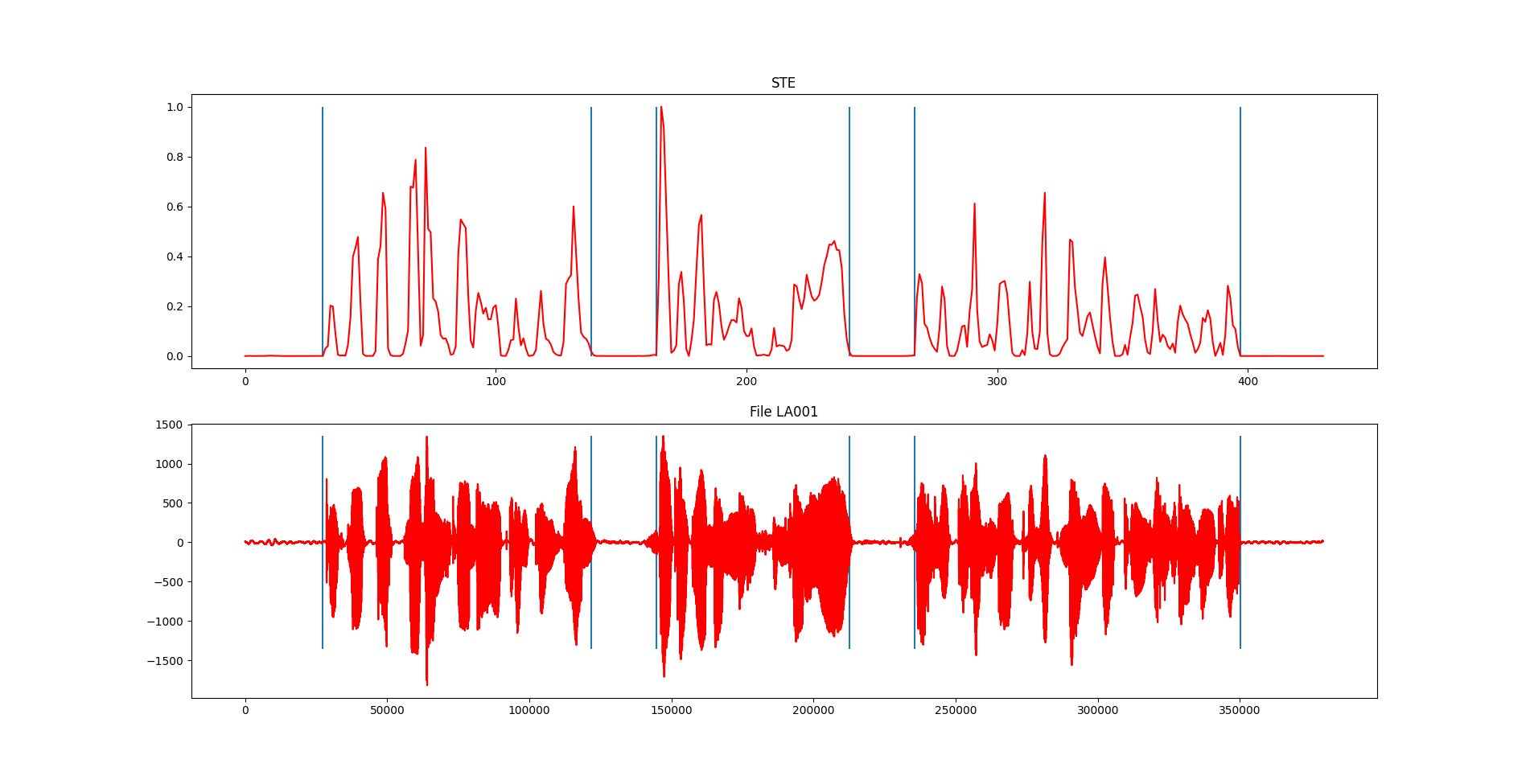
1. MA lab\_male



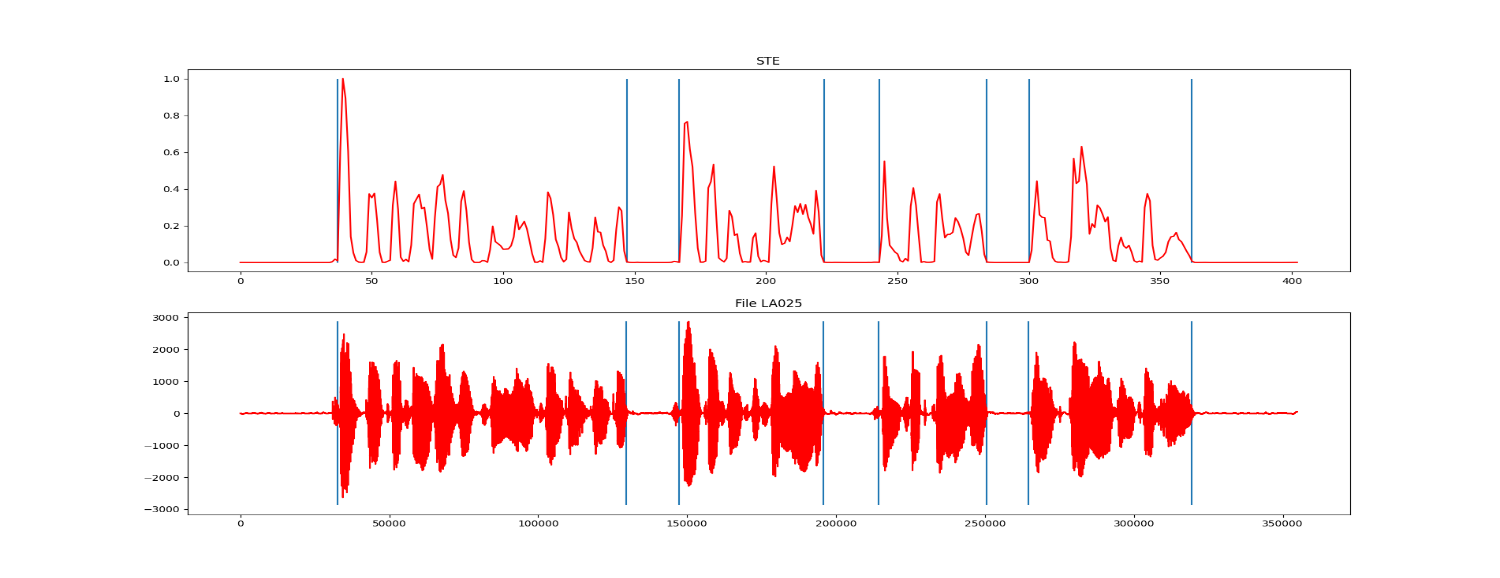
1. MA studio\_female



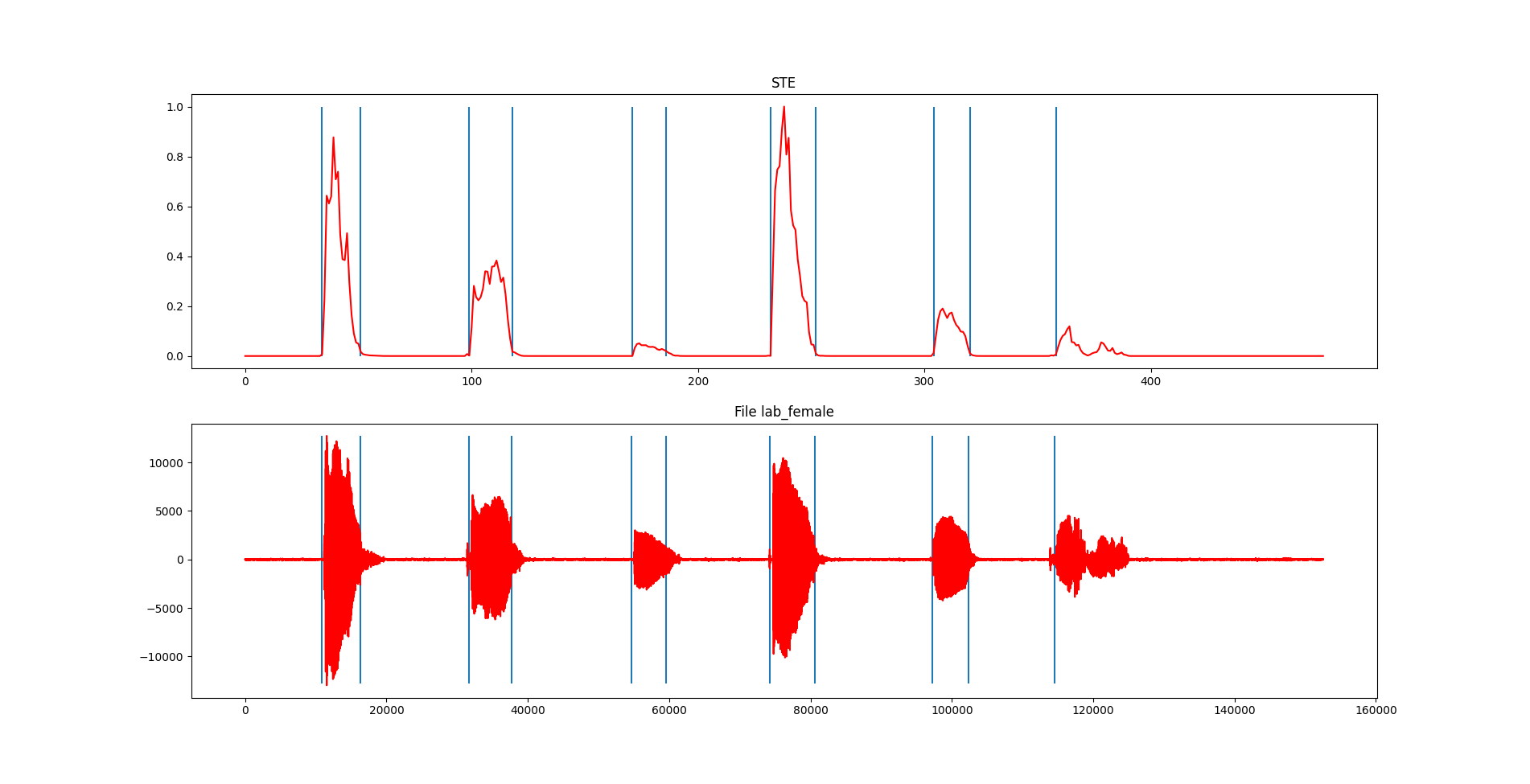
1. MA studio-male



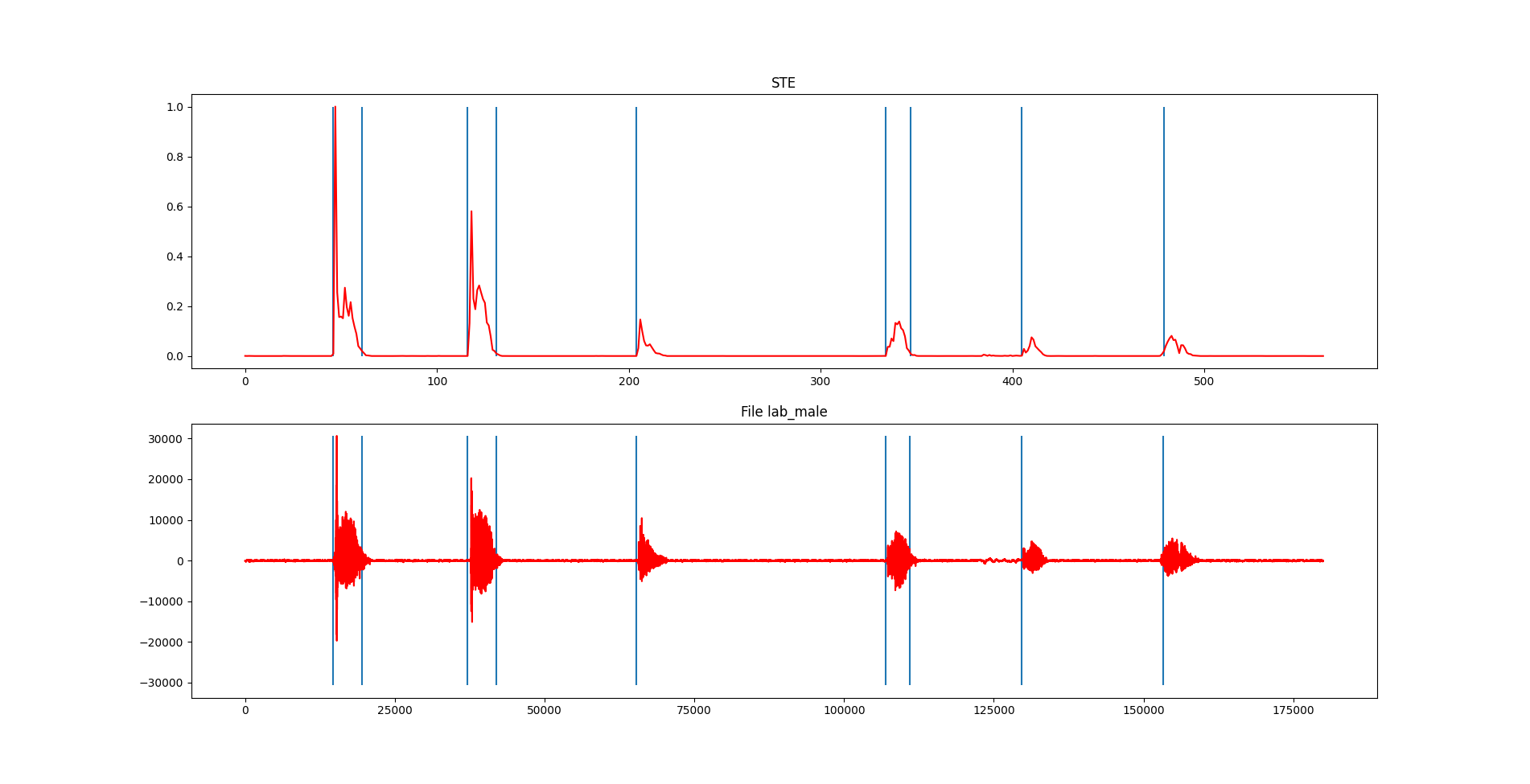
1. STE LA 001



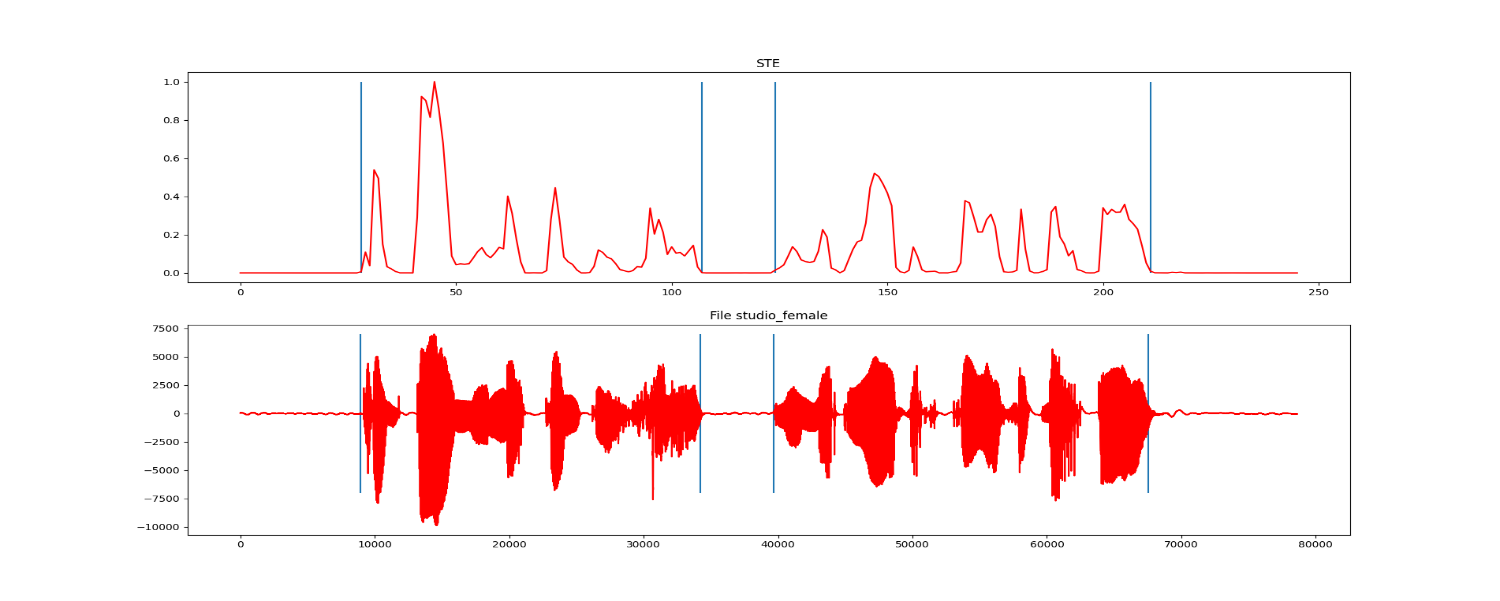
1. STE LA025



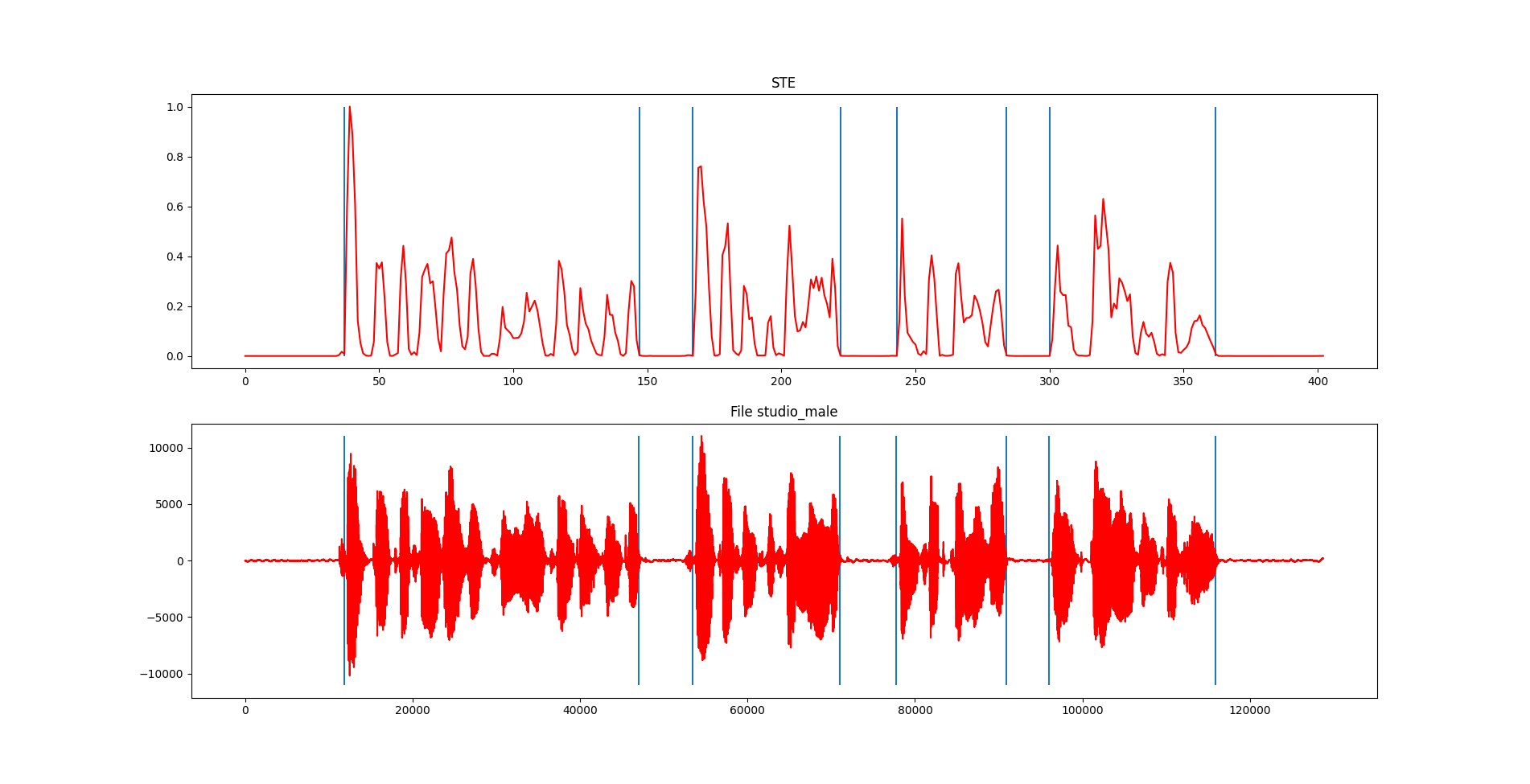
1. STE lab\_female



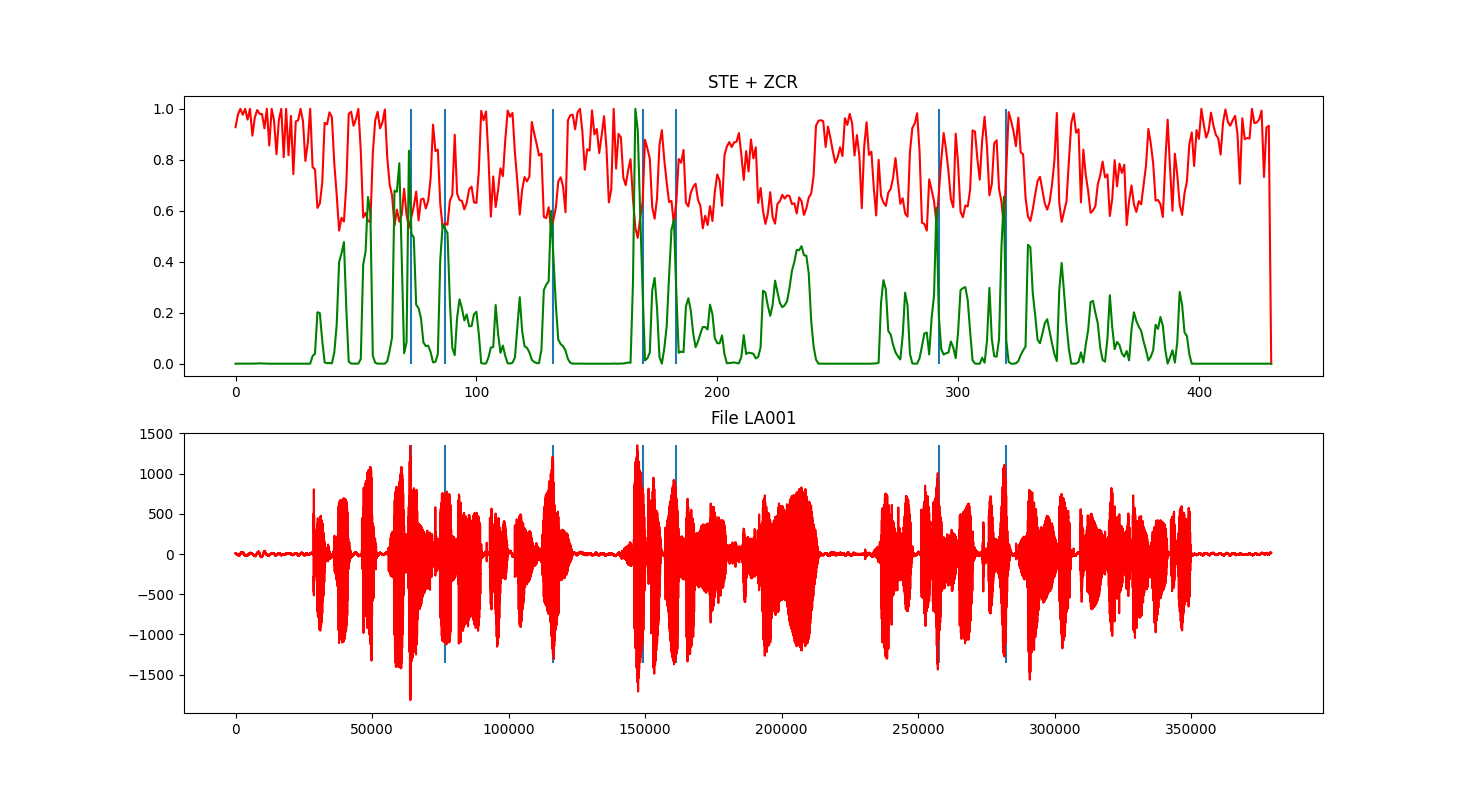
1. STE lab\_male



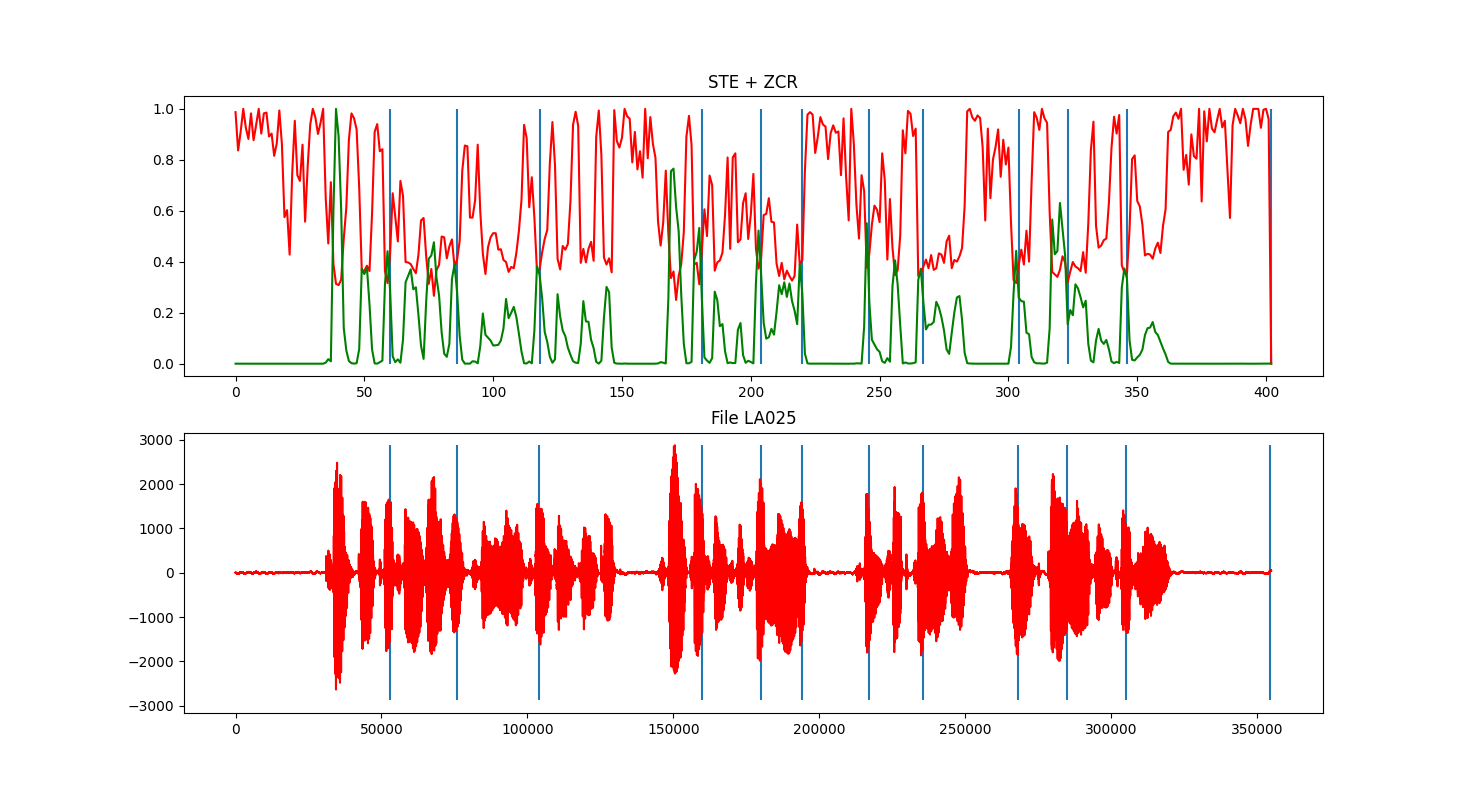
1. STE studio\_female



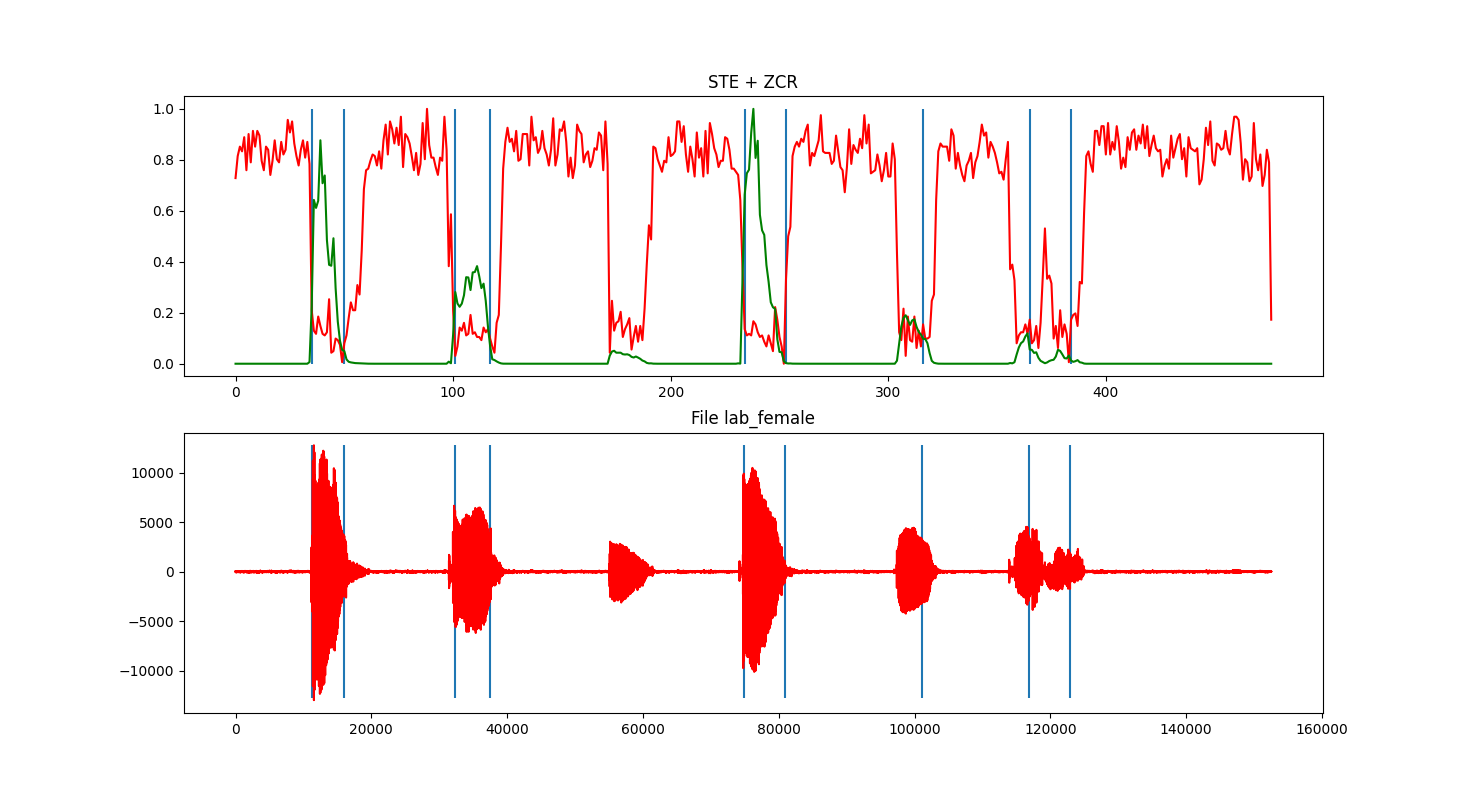
1. STE studio\_male



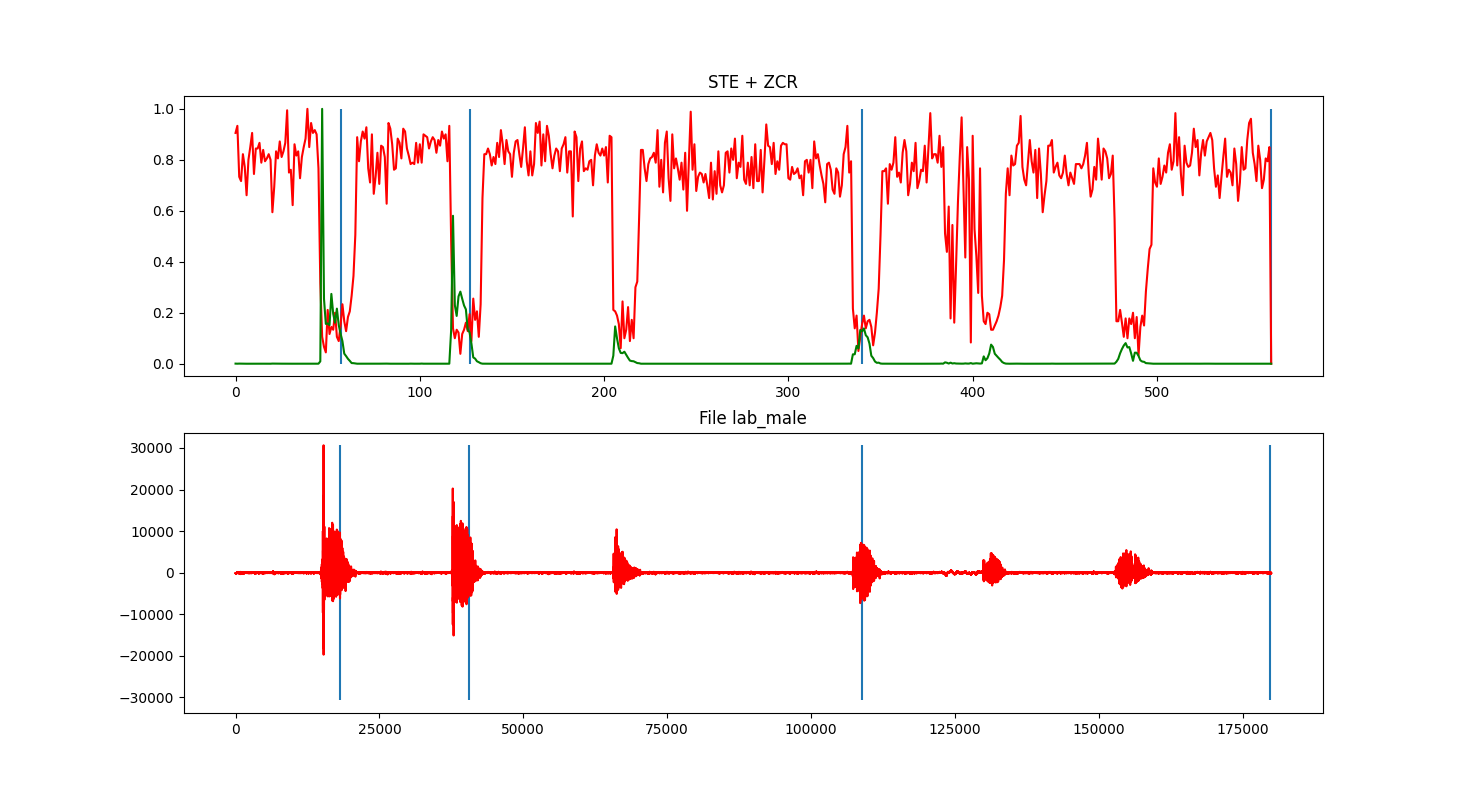
1. ZRC + STE LA001



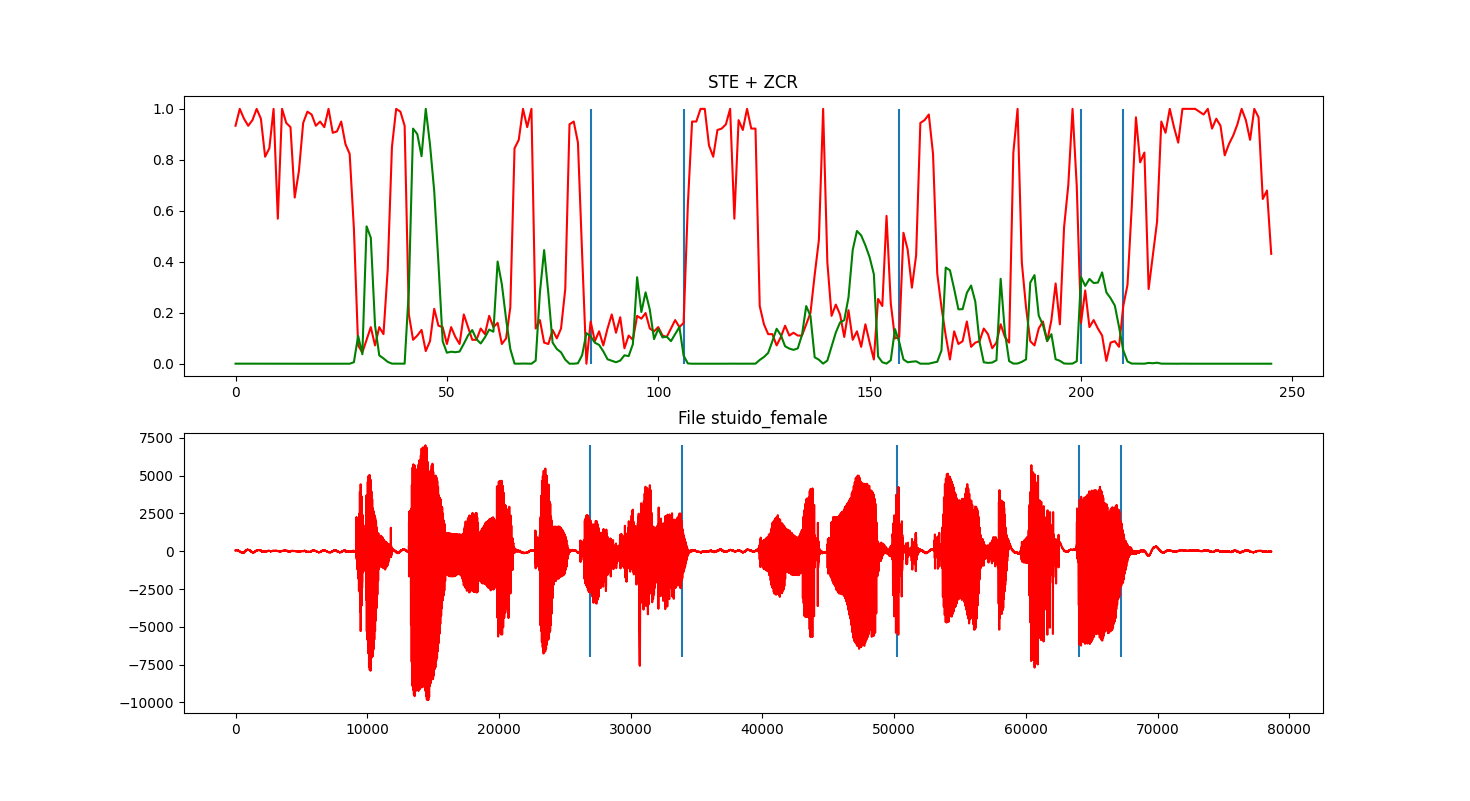
1. ZRC + STE LA025



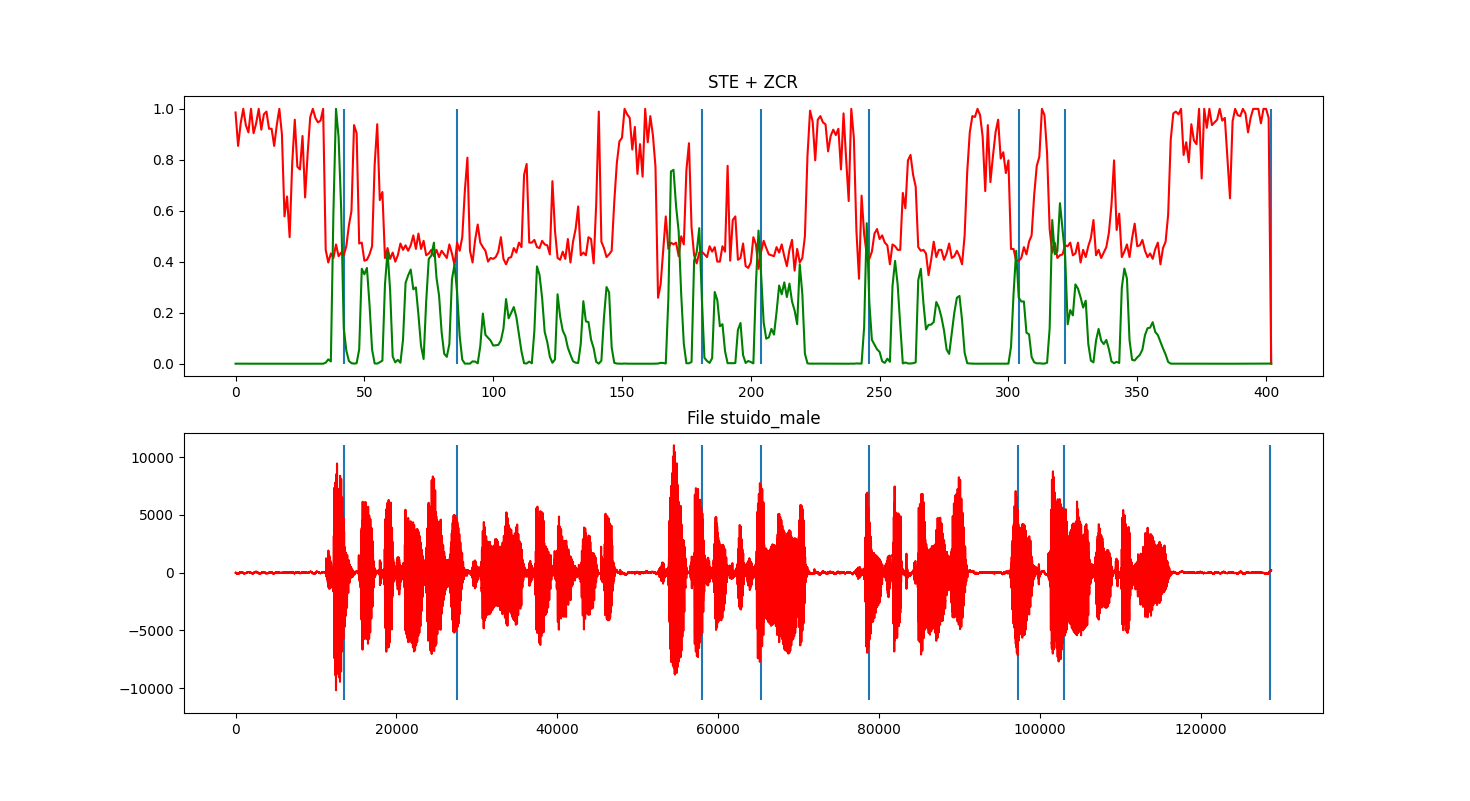
1. ZRC + STE lab\_female



1. ZRC + STE lab\_male



1. ZRC + STE studio\_female



1. ZRC + STE studio\_male

## Bảng biểu

1. Bảng số liệu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Fs | Độ dài file (s) | Tổng số mẫu (mẫu) | Độ dài 1  frame(s) | Số mẫu / 1 frame (mẫu) | Tổng số frame |
| LA001.wav | 44100 | 8.6001 | 379268 | 0.02 | 882 | 431 |
| LA025.wav | 44100 | 8.045 | 354828 | 0.02 | 882 | 403 |
| lab-female.wav | 16000 | 9.533 | 152529 | 0.02 | 320 | 477 |
| lab\_male.wav | 16000 | 11.25 | 179985 | 0.02 | 320 | 563 |
| studio\_female.wav | 16000 | 4.9136 | 78618 | 0.02 | 320 | 246 |
| studio­\_male.wav | 16000 | 8.046 | 128736 | 0.02 | 320 | 473 |

1. Bảng kết quả xác định biên theo thời gian(s)

|  |  |  |  |
| --- | --- | --- | --- |
|  | STE(s) | MA(s) | STE+ZCR(s) |
| LA001.wav | 0.62-2.76-3.28-4.82-5.34-7.94 | 0.62-2.78-3.28-4.84-5.34-7.94 | 1.46-1.74-2.64-3.38-7.66-5.84-6.4 |
| LA025.wav | 0.74-2.94-3.34-4.44-4.86-5.68-6.0-7.24 | 0.7-2.94-3.34-4.44-4.86-5.68-6.0-7.24 | 1.2-1.72-2.36-3.62-4.08-4.4-4.92-5.34-6.08-6.46-6.92-8.04 |
| lab-female.wav | 0.68-1.02-1.98-2.36-3.42-3.72-4.64-5.04-6.08-6.4-7.16 | 0.68-1.04-1.98-2.4-3.42-3.72-4.64-5.04-6.08-6.4-7.16-7.76 | 0.7-1.0-2.02-2.34-4.68-5.06-6.32-7.3-7.68 |
| lab\_male.wav | 0.92-1.22-2.32-2.62-4.08-6.68-6.94-8.1-9.58 | 0.92-1.26-2.32-2.64-4.08-4.34-6.68-6.96-8.1-9.54-9.88 | 1.14-2.54-6.8-11.24 |
| studio\_female.wav | 0.56-2.14-2.48-4.22 | 0.56-2.14-2.46-4.22 | 1.68-2.12-3.14-4.0-4.2 |
| studio­\_male.wav | 0.74-2.94-3.34-4.44-4.86-5.68-6.0-7.24 | 0.7-2.94-3.34-4.44-4.86-5.68-6.0-7.24 | 0.84-1.72-3.62-4.08-4.92-6.08-6.48-8.04 |

# KẾT LUẬN

Bài báo cáo này thực hiện việc phân đoạn tín hiệu âm thanh, chúng em đã phân đoạn được một vài tín hiệu nhưng cũng có những trường hợp không đúng kết quả mong đợi, và vẫn chưa xác định được các đoạn nhiễu có trong tín hiệu.Trong đó việc sử dụng phương pháp MA mang lại kết quả có độ chính xác cao nhất và phương phát kết hợp giữa ZCR và STE mang lại kết quả có nhiều sai sót. Trong tương lai chúng em sẽ cải thiện thuật toán để đạt được tín hiệu mong muốn.

# TÀI LIỆU THAM KHẢO

1. Link: <https://en.wikipedia.org/wiki/Speech_processing>
2. Link:[https://en.wikipedia.org/wiki/Speech segmentation](https://en.wikipedia.org/wiki/Speech%20segmentation)
3. Matthieu Hodgkinson -“CS425 Audio and Speech Processing”, Ireland, 2012
4. Link: [https://www.asee.org/documents/zones/zone1/2008/student/ASEE12008\_0044\_paper.pdf?fbclid=IwAR166Jm4iHEsBZxnFK4Hi-rXUCZ1aXEeay0fRO8LUjQfm06KJuDS-sP7gxA](%09https://www.asee.org/documents/zones/zone1/2008/student/ASEE12008_0044_paper.pdf?fbclid=IwAR166Jm4iHEsBZxnFK4Hi-rXUCZ1aXEeay0fRO8LUjQfm06KJuDS-sP7gxA)