Bài tập lớn Xử lý tiếng nói

Báo cáo

Đề bài: Tìm điệp khúc của bài hát

(Đề bài cũ: Nhập một bài hát bất kì và xuất ra lời bài hát)

1. Mở đầu

Mặc dù em đã chọn đề bài là xuất lời bài hát, nhưng em đã gặp khó khăn trong việc tạo dữ liệu (tách các đoạn có giọng nói trong bài hát một cách tự động) và việc train dữ liệu, nên em đã chuyển sang đề bài thầy gợi ý cho em ban đầu là xác định được điệp khúc.

1. Giới thiệu

Về bài tập xác định điệp khúc của bài hát, em có tìm hiểu theo 2 hướng, đó là thuật toán fingerprinting của Shazam và thao tác trực tiếp với phổ tần số. Lý do em phải tìm hiểu theo 2 hướng này là bởi ban đầu em thấy thuật toán của Shazam mặc dù mặc định dùng cho việc so sánh dữ liệu từ database với dữ liệu thu ngoài nhưng cũng có thể sử dụng theo một cách nào đấy đối với bài tập lớn của em. Tuy nhiên qua quá trình làm thì em phát hiện một số vấn đề khiến em phải ngừng đi theo hướng này và tìm hiểu hướng khác. Và thời gian để em cố gắng khắc phục vấn đề với thuật toán của Shazam khiến em không có đủ thời gian để hoàn chỉnh bài tập của em. Em rất xin lỗi thầy và em cũng có phần thất vọng với kết quả của em. Có thể thuật toán của Shazam có thể áp dụng vào bài tập của em nhưng do cách em ứng dụng chưa đúng nên kết quả chưa được như mong đợi.

1. Shazam

Giống với tên gọi của thuật toán thì kết quả của thuật toán fingerprint sẽ tạo ra các khối dữ liệu rất đặc trưng và rất khó để có được sự giống nhau giữa 2 khối dữ liệu khác nhau (giống như hashing, một hàm mà Shazam cũng sử liệu trong thuật toán này).

1. Xử lý dữ liệu đầu vào:

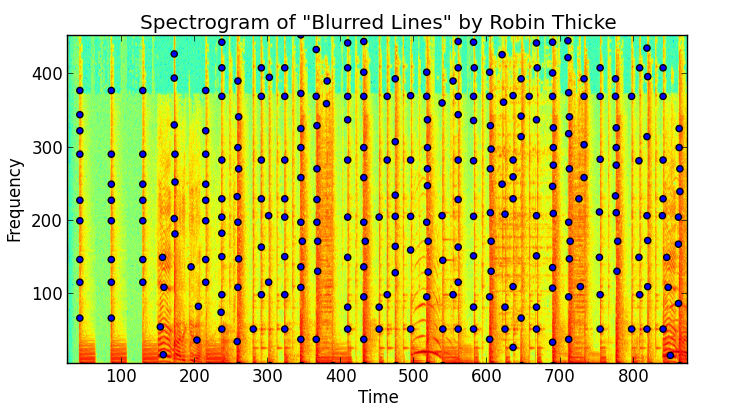
Để tạo database thì đầu tiên Shazam cần chuyển đổi dữ liệu miền thời gian của file âm thanh về miền tần số để triết tách dữ liệu làm fingerprint. Ở đây Shazam sẽ sử dụng hàm stft (Short-Time Fourier Transform) để chuyển đổi dữ liệu về miền tần số mà vẫn giữ được dữ liệu miền thời gian. Đặc điểm của hàm stft của Shazam:

* Sample rate: 44100Hz
* n\_fft: 4096 = 2\*\*12
* Overlap: 50%

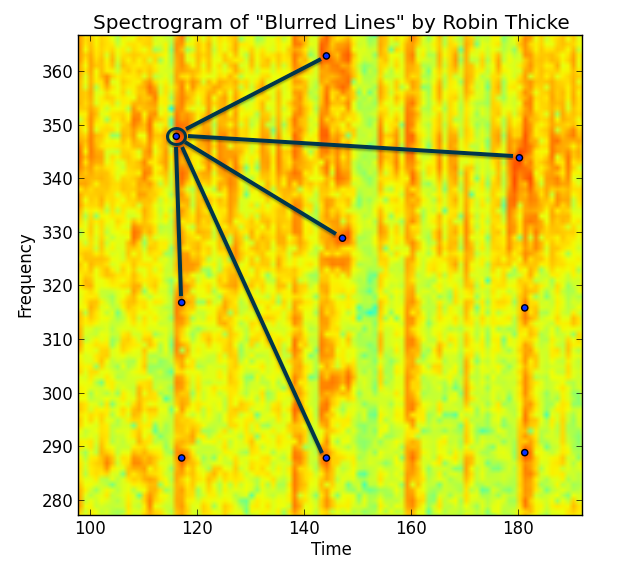
Điều này sẽ đảm bảo lượng dữ liệu vừa đủ để xử lý. n\_fft quá nhỏ thì sẽ gây nhiều dữ liệu và không đủ tần số sample. n\_fft qúa lớn thì mặc dù lượng tần số sample có thể nhiều hơn (giúp cho việc fingerprinting) nhưng sẽ không có ít sample để tạo dữ liệu hơn.

1. Tạo fingerprint:

Sau khi đã chuyển đổi dữ liệu xong thì sẽ đến bước tinh giảm dữ liệu nhằm mục đích tạo fingerprint. Shazam sẽ chia lượng tần số sample thu được thành các miền khác nhau và tiếp theo tìm tần số có biên độ lớn nhất của từng miền (gọi tắt là cực đại). Việc sử dụng cực đại để tạo fingerprint là bởi các cực đại này sẽ giúp dễ phân biệt hơn trong trường hợp có nhiều tiếng ồn hoặc nhiễu từ nền.

**

Sau đó tương ứng với mỗi cực đại, sẽ sử dụng một cửa sổ để chọn ra những cực đại lân cận nằm trong cửa sổ đó để tạo ra sự riêng biệt. Khoảng cách từ cực đại đang xét đối với từng cực đại lân cận sẽ được tính toán để đưa ra một kết quả nhất định.



Và với mỗi cửa sổ trượt sẽ có số lượng kết quả tương ứng với số miền được chia. Và cũng từ những kết quả đó Shazam sẽ sử dụng hàm hash để tạo ra một giá trị độc nhất cho từng cửa sổ trượt. Và cuối cùng dữ liệu sẽ được lưu dưới dạng (giá trị hash, thời điểm bắt đầu của cửa sổ trượt, tên bài hát). Khi sử dụng ứng dụng để thu dữ liệu đối chiếu để nhận biết bài hát, dữ liệu thu được cũng sẽ được xử lí tương tự, và việc chọn cực đại sẽ phát huy tác dụng tại đây bởi môi trường thu sẽ có thể có rất nhiều tiếng ồn và nhiễu và việc chọn cực đại sẽ giúp loại bỏ nhiễu âm thanh một các hiệu quả.

1. Ứng dụng vào bài tập:

Đối với bài tập của em, em sẽ sử dụng fingerprint để tạo dữ liệu cho bài hát. Tuy nhiên thay vì so sánh với dữ liệu thu ngoài thì em sẽ so sánh chính dữ liệu vừa tạo được để tìm các điểm tương đồng, từ đó xác định điệp khúc. Về cách tạo fingerprint vẫn tương tự với Shazam, chỉ khác về các thông số sử dụng. Bởi Shazam không công khai chi tiết thuật toán nên em đã thử với các thông số khác nhau (n\_fft, sample rate, các miền tần số sample,…). Về chọn các cực đại lân cận theo cửa sổ, thì em thay thế bằng năm cực đại của năm cửa sổ gần nhất để tạo giá trị hash. Em cũng chưa thử phân biệt khoảng cách của các cực đại giữa các miền tần số khác nhau.

Một số vấn đề em gặp phải là trong một số lần thử thì em phát hiện các cửa sổ trượt có vị trí gần nhau thì cho kết quả giống nhau cao hơn so với các cửa sổ trượt ở các điệp khúc. Và vấn đề này em chưa tìm được hướng giải quyết.

1. Thao tác với tần số:

Về hướng này thì thay vì sinh fingerprint từ phổ tần số thì sẽ sử dụng luôn phổ tần số để phân tích vị trí điệp khúc.

1. Phân tích phổ tần số:

Đầu tiên em vẫn dùng stft để biến đổi dữ liệu từ miền thời gian về miền tần số. Sau đó em sẽ đơn giản dữ liệu bằng cách dùng hàm chroma của librosa để chuyển dữ liệu stft về tần số các nốt nhạc. Việc này sẽ làm giảm việc phân tích dữ liệu đi nhiều bởi không phải quan tâm đến tất cả dải tần số 20-20000Hz nữa mà chỉ còn 12 nốt (kể cả nốt thăng và giáng) theo thang âm nhạc của Mỹ. Tiếp theo em sẽ tạo một ma trận đối xứng thời gian – thời gian của phổ nốt này. Ma trận này có đặc điểm những phần giống nhau của bài hát sẽ tạo thành một đường chéo trên ma trận. Từ đó ta có thể nhận biết được phần nào là điệp khúc. Và để đơn giản hóa hơn việc nhận biết thì em sẽ tạo ra một ma trận thời gian – độ trễ, ở đây các đường chéo thể hiện sự tương đồng sẽ được chuyển thành đường thẳng để thao tác nhận biết được dễ hơn.

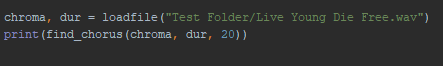
1. Đếm:

Tiếp theo chúng ta sẽ đếm xem trên ma trận này có bao nhiêu đường thẳng từ đó suy ra được số điệp khúc và thời điểm bắt đầu điệp khúc. Bởi đây là ma trận đối xứng có kích thước nxn (n là số cửa sổ trượt) nên số điệp khúc sẽ bằng số đoạn thẳng đếm được. Đầu tiên ta sẽ làm các đoạn thẳng nằm ngang trên ma trận rõ hơn bằng cách giảm giá trị của các đoạn thẳng đứng và đoạn thẳng xiên. Tiếp theo ta sẽ xét từng đoạn thẳng nằm ngang tìm được và xem có thỏa mãn điều kiện không (ở đây ta phải nhập tiền điều kiện là độ dài ngắn nhất của điệp khúc nhằm loại bỏ các kết quả ngắn hơn tìm được).

1. Ứng dụng:

Bởi hướng đi này chỉ quan tâm đến điểm tương đồng giữa các thời điểm trong bài hát nên sẽ không để ý đến các phần cũng tương đồng nhưng không được coi là điệp khúc (bridge, các verse lặp lại, có trường hợp cả bài hát chỉ lặp lại một verse hoặc hơn nữa là một bar, …). Đồng thời do việc tìm cách khắc phục vấn đề của thuật toán Shazam gây mất thời gian nên em chưa xây dựng được ứng dụng hoàn chỉnh.

1. Sử dụng:



Nhập địa chỉ file âm nhạc cần xác định vào hàm “loadfile” và chạy.