BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA PHÁT TRIỂN NÔNG THÔN

------ᘓ✧ᘑ------

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

Đề Tài

CÁC PHƯƠNG PHÁP TÌM KIẾM NHẠC CÓ LỜI

Giảng viên hướng dẫn
TS.Nguyễn Thanh Hải

Sinh viên: Lê Thị Diệu Hiền
MSSV: B1812793
Ngành: Công nghệ thông tin
Khóa: 44

Cần Thơ, 5/2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA PHÁT TRIỂN NÔNG THÔN

------ᘓ✧ᘑ------

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

Đề Tài

CÁC PHƯƠNG PHÁP TÌM KIẾM NHẠC CÓ LỜI

Giảng viên hướng dẫn
TS.Nguyễn Thanh Hải

Sinh viên: Lê Thị Diệu Hiền
MSSV: B1812793
Ngành: Công nghệ thông tin
Khóa: 44

Cần Thơ, 5/2023

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

XÁC NHẬN CHỈNH SỬA LUẬN VĂN THEO YÊU CẦU CỦA HỘI ĐỒNG

Tên luận văn (tiếng Việt và tiếng Anh): Các phương pháp tìm kiếm nhạc có lời, Approaches for Searching songs with lyrics

Họ tên sinh viên: Lê Thị Diệu Hiền MASV: B1812793

Mã lớp: HG18V7A2

Đã báo cáo tại hội đồng ngành:

Ngày báo cáo: 16/5/2023

Hội đồng báo cáo gồm:

1. TS. Trần Công Án — Chủ tịch hội đồng
2. TS. Thái Minh Tuấn — Thành viên
3. TS. Nguyễn Thanh Hải — Thư ký

Luận văn đã được chỉnh sửa theo góp ý của Hội đồng.

Cần Thơ, ngày 23 tháng 5 năm 2023

Giáo viên hướng dẫn

(Ký và ghi họ tên)

Nguyễn Thanh Hải

LỜI CẢM ƠN

Em xin gửi lời cảm ơn đến toàn thể thầy cô của trường Đại học Cần Thơ, quý thầy cô của trường Công Nghệ Thông Tin và Truyền Thông. Đặc biệt là tất cả các thầy cô của khoa Công Nghệ Thông tin đã truyền đạt những kiến thức để em có nền tảng để tiếp tục đi trên con đường sắp tới.

Em xin cảm ơn đến thầy Nguyễn Thanh Hải đã cung cấp tài liệu và tận tình hướng dẫn em để em có thể hoàn thành tốt bài luận văn trong thời gian qua. Em cảm ơn thầy rất nhiều.

Em xin cảm ơn đến cố vấn của em là thầy Cao Hoàng Tiến và cô Sử Kim Anh đã đồng hành và hỗ trợ em trong suốt thời gian học đại học.

Trong quá trình làm đề tài vì em còn hạn chế một số kiến thức nên có nhiều thiếu sót. Em mong sẽ nhận được tất cả những ý kiến đóng góp, những lời chỉ dạy của thầy, cô để em có thể học hỏi và rút kinh nghiệm để có thể hoàn thành tốt bài luận văn hơn.

Đồng cảm ơn đến các tác giả trong các quyển sách báo, internet, anh chị đi trước đã tìm tòi, nghiên cứu đúc kết kinh nghiệm làm tài liệu để em có thể tham khảo trong quá trình thực hiện đề tài.

Em xin kính chúc tất cả các thầy, cô, những người đã đồng hành cùng em mạnh khỏe, vui vẻ và nhiều thành công trên con đường sắp tới.

Cần Thơ, tháng 5 năm 2023

Sinh viên thực hiện

Lê Thị Diệu Hiền

LỜI CAM ĐOAN

Tôi xin cam kết luận văn này hoàn thành dựa trên kết quả nghiên cứu của tôi và các kết quả này là trung thực, không sao chép từ bất kỳ nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

Tôi xin hoàn toàn chịu trách nhiệm về kết quả và các số liệu được trình bày trong luận văn này.

Cần thơ, ngày tháng năm 2023

Sinh viên thực hiện

Lê Thị Diệu Hiền

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Cần Thơ, tháng 5 năm 2023

Giáo viên hướng dẫn

Nguyễn Thanh Hải

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Cần Thơ, tháng 5 năm 2023

Giáo viên phản biện

TÓM TẮT

Hiện nay, Internet ngày càng phát triển cùng với sự gia tăng của thông tin đa phương tiện dưới nhiều dạng khác nhau. Do đó, người dùng có nhiều cơ hội lựa chọn và tận hưởng những gì đa phương tiện đem lại. Trong số đó , âm thanh kỹ thuật số kết hợp với các dịch vụ có ý nghĩa thực tiễn và quan trọng trong nhiều lĩnh vực kinh tế xã hội như các dịch vụ truyền hình, đài phát thanh, hội thảo truyền hình qua mạng, xem video, nghe nhạc,… và đặc biệt với phương diện mạng xã hội như Youtube, Facebook, Tik Tok.

Cùng với sự phát triển của Internet thì âm nhạc cũng phát triển với đa dạng nhiều thể loại như nhạc dân gian, nhạc điện tử, nhạc hiện đại. Kèm theo tích hợp cùng với mạng xã hội để phục vụ cho nhu cầu của con người - nhu cầu giải trí. Với số lượng bài hát và nhạc càng ngày càng đồ sộ thì việc sắp xếp, truy tìm các nội dung, cũng như việc tìm kiếm ngày càng khó khăn. Giả sử nếu người dùng có thể nhớ được tên bài hát đó thì việc tìm kiếm trở nên dễ dàng, nhưng nếu người dùng chỉ biết lời bài hát hoặc có một đoạn ghi âm của bài hát đó thì việc tìm kiếm trở nên khó khăn hơn.

Với vấn đề trên, đề tài "Các phương pháp tìm nhạc có lời" là nghiên cứu để tìm ra các phương pháp tìm kiếm bài hát dựa trên một đoạn audio hoặc một số từ khóa mà người dùng cung cấp.

Đề tài nghiên cứu phương pháp thứ nhất là trích xuất các đặc trưng giọng nói (MFCCs), thư viện python_speech_features để trích xuất các đặc trưng của đoạn audio người dùng tải lên cùng với thuật toán Approximate Nearest Neighbors (ANN) để hỗ trợ tìm kiếm lân cận và cho kết quả chính xác cao.

Phương pháp thứ hai là sử dụng Google API với thư viện speech recognition để xuất ra văn bản từ đoạn audio mà người dùng đưa vào. Sau đó tìm kiếm lời bài hát có độ tương đồng với văn bản được xuất ra trong thư viện lời nhạc đã thu thập trước đó.

Bài nghiên cứu thực hiện với ngôn ngữ Python

Từ khóa: python_speech_features, nhận dạng giọng nói, trích xuất âm thanh đặc trưng MFCCs, Google API, tìm kiếm lân cận ANN, nhạc có lời, phương pháp tìm kiếm nhạc có lời.

ABSTRACT

Now, the Internet has grown in tandem with the rise of multimedia information in various forms. Consequently, users have many opportunities to choose from and enjoy what multimedia has to offer. Among them, digital audio combines with services that are practical and important in many socio-economic areas such as television services, radio, video conferencing, video watching, music listening, ... and especially with social media such as Youtube, Facebook, Tik Tok.

As the Internet grew, music also diversified into folk, electronic, and modern music. Accompanying integration with social networks to serve the needs of people - entertainment needs. With the growing number of songs and music, it became increasingly difficult to sort and retrieve the contents, as well as to find them. If the user can remember the name of the song, the search becomes easy, but if the user knows only the lyrics or has a recording of the song, the search becomes more difficult.

The subject "Approaches for Searching songs with lyrics" is a study that seeks to find song-finding methods based on a particular audio or keyword segment that the user provides.

The first method was to extract voice characteristics (MFCCs), a python_speech_features library to extract user-uploaded audio characteristics along with the Nearest Neighbors (ANN) algorithm in support of a search and high-precision result.

The second method is to use a Google API with speech recognition library to output text from the audio that the user inserted. Then search for lyrics that are similar to the text published in the previously collected lyrics library.

Research done with Python language.

Tag: python_speech_features, speech recognition, MFCCs, Google API,Approximate Nearest Neighbors, song with lyric, approaches for searching songs with lyrics.

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH MỤC TỪ CHUYÊN NGÀNH

VIẾT TẮT	Giải thích
ANN	Thuật toán tìm kiếm lân cận (Approximate Nearest Neighbors)
API	Là phương thức hay cơ chế cho phép 2 thành phần của phần mềm giao tiếp với nhau (Application Programming Interface)
A/D Conversion	Analog-to-Digital Converter là hệ thống mạch thực hiện chuyển đổi một tín hiệu analog (tín hiệu tương tự) liên tục
Cover	bài hát được làm lại hoặc hát lại từ một ca khúc đã hát và thu âm trước đó
Lyric	Lời bài hát
MFCCs	Phương pháp trích xuất âm thanh đặc trưng (Mel Frequency Cepstral Coefficients)
ML	Machine Learning - Máy học
VS Code	Phần mềm lập trình Visual Studio Code

CHƯƠNG 1: GIỚI THIỆU

1.1. Đặt vấn đề

Internet ngày càng phát triển cùng với sự gia tăng của thông tin đa phương tiện dưới nhiều dạng khác nhau. Do đó, người dùng có nhiều cơ hội lựa chọn và tận hưởng những gì đa phương tiện đem lại. Trong số đó , âm thanh kỹ thuật số kết hợp với các dịch vụ có ý nghĩa thực tiễn và quan trọng trong nhiều lĩnh vực kinh tế xã hội như các dịch vụ truyền hình, đài phát thanh, hội thảo truyền hình qua mạng, xem video, nghe nhạc,… và đặc biệt với phương diện mạng xã hội. Âm nhạc cũng phát triển với đa dạng nhiều thể loại như nhạc dân gian, nhạc điện tử, nhạc hiện đại tích hợp cùng với mạng xã hội như Youtube, Facebook, TikTok, … để phục vụ cho nhu cầu giải trí của con người. Với số lượng bài hát và nhạc càng ngày càng đồ sộ thì việc sắp xếp, truy tìm các nội dung, cũng như việc tìm kiếm ngày càng khó khăn. Với số lượng bài hát gốc đã đồ sộ như vậy nhưng hiện nay càng ngày có nhiều những bài nhạc đã được remix (phối lại - phiên bản khác của một ca khúc thu âm gốc) cũng làm người dùng khó nhớ được tên bài hát. Giả sử nếu người dùng có thể nhớ được tên bài hát đó thì việc tìm kiếm trở nên dễ dàng, nhưng nếu người dùng chỉ biết lời bài hát hoặc có một đoạn ghi âm của bài hát đó thì việc tìm kiếm trở nên khó khăn hơn.

Chính vì nguyên nhân trên đã khiến em muốn thực hiện và phát triển đề tài “Các phương pháp tìm nhạc có lời” để tìm ra những phương pháp tìm kiếm nhạc nhanh chóng và có độ chính xác cao cũng như là một đề tài mới và phù hợp với tình trạng hiện nay.

1.2. Mục tiêu của đề tài

Mục tiêu của đề tài là nghiên cứu để tìm ra các phương pháp tìm kiếm bài hát dựa trên một đoạn audio mà người dùng cung cấp.

1.3. Nội dung nghiên cứu

- Tìm hiểu phương pháp phân tích âm thanh qua số liệu và kiểm tra thực nghiệm các tham số để tổng kết được những kết quả tốt nhất.
- Tìm hiểu về thuật toán được sử dụng trong bài như ANN và các thư viện khác.

- Tìm hiểu về ngôn ngữ lập trình python được sử dụng trong đề tài này.

1.4. Đối tượng và phạm vi nghiên cứu

- Các khái niệm cơ bản về đặc trưng âm thanh.
- Thuật toán ANN, ngôn ngữ Python và các công cụ hỗ trợ khác.
- Phạm vi nghiên cứu: nhạc Việt có lời (.mp3 và .wav).
- Nghiên cứu chương trình tìm kiếm bằng đoạn nhạc

1.5. Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu

- Luận văn nghiên cứu kỹ thuật tìm kiếm bài theo từng phương pháp.
- Cài đặt thử nghiệm các kỹ thuật và giải quyết bài toán và thu được các kết quả thử nghiệm với các tham số khác nhau.

1.6. Bố cục luận văn

Nội dung quyển báo cáo bao gồm:

Chương 1: Giới thiệu

Chương 2: Lược khảo tài liệu

Chương 3: Phương pháp nghiên cứu

Chương 4: Kết quả

Chương 5: Kết luận và hướng phát triển

CHƯƠNG 2: LƯỢC KHẢO TÀI LIỆU

2.1. Tài liệu nghiên cứu liên quan

Shazam [1] một ứng dụng được phát triển bởi Shazam Entertainment Ltd. Cuối năm 2017, Apple mua lại Shazam Entertainment Ltd. Shazam có thể giúp bạn xác định nhạc phim, quảng cáo, bất kì bài hát nào trên mạng xã hội dựa trên một đoạn phát ngắn và sử dụng micro của thiết bị. Với giao diện dễ sử dụng và dùng để nhận dạng bản thu âm giọng nói thông qua hệ thống mạng điện thoại di động. Người dùng có thể tìm kiếm và cung cấp toàn bộ lời bài hát kèm theo và có thể lưu trữ lại các bài hát đã được nhận diện để có tốc độ nhận diện và mức độ chính xác của bài hát cao hơn.

SoundHound [13] cũng là một trong những ứng dụng tìm kiếm nhạc bằng giai điệu phổ biến hiện nay. Điểm đặc biệt của ứng dụng này là phát hiện bài hát dựa trên giai điệu ngân nga thay vì cần cả lời nhạc như Shazam. Và kèm theo chức năng cho kết quả thông tin về nghệ sĩ cũng như phát bài hát.

Google, Siri, Alexa [13] là những cái tên quen thuộc và đều có chức năng tìm kiếm nhạc mà người dùng thường xuyên sử dụng nhất. Mọi người có thể tìm kiếm bài hát thông qua trợ lý ảo bằng cách nhấn vào micro, đưa giai điệu vào để Google, Siri, Alexa nhận diện bài hát và sau đó sẽ trả về các kết quả chính xác nhất mà người dùng cần tìm kiếm.

2.2. Mô tả bài toán

Đề tài “Các phương pháp tìm nhạc có lời” là nghiên cứu tìm kiếm bài hát dựa trên một đoạn audio hoặc một số từ khóa mà người dùng cung cấp với hai phương pháp chính:

- Phương pháp thứ nhất là trích xuất các đặc trưng giọng nói (MFCCs), thư viện python_speech_features thuật toán Approximate Nearest Neighbors (ANN).

- Phương pháp thứ hai là sử dụng Google API với thư viện speech recognition để xuất ra văn bản từ đoạn audio mà người dùng đưa vào. Sau đó tìm kiếm lời bài hát có độ tương đồng với văn bản được xuất ra trong thư viện lời nhạc đã thu thập trước đó.

2.3. Hướng nghiên cứu

2.3.1. Hướng nghiên cứu 1

Đề tài "Các phương pháp tìm nhạc có lời" xây dựng trên phương pháp trích xuất âm thanh đặc trưng MFCCs. Âm thanh có những đặc trưng và những đặc trưng đó là những đặc điểm để so sánh với đoạn âm thanh khác để từ đó sẽ đưa ra kết quả trùng với đoạn nhạc cần tìm. Sử dụng thuật toán ANN và thư viện Annoy để hỗ trợ tìm kiếm dựa trên cây thư mục tạo ra trước đó.

2.3.2. Hướng nghiên cứu 2

Đề tài "Các phương pháp tìm nhạc có lời" dựa trên phân tích đoạn audio bằng Google API với thư viện Speech recognition để hỗ trợ chuyển âm thanh thành văn bản. Sau khi chuyển được đoạn âm thanh thành văn bản lời bài hát thì chúng ta sẽ tìm kiếm độ tương đồng của văn bản với các lời bài hát đã lưu trong kho dữ liệu. Quá trình chuyển đổi từ âm thanh có thể không hoàn toàn chính xác hết được so với bài hát gốc nên việc tìm kiếm bằng cách này còn nhiều hạn chế.

2.4. Cơ sở lý thuyết

2.4.1. Xử lý dữ liệu âm thanh

- Thư viện python hỗ trợ xử lý âm thanh[2][25][26][27]: librosa, scipy, torchaudio (Hình 2.1).

Hình 2.1. Logo thương hiệu của các thương hiệu hỗ trợ xử lý âm thanh

- Âm thanh [4] là các dao động cơ học (biến đổi vị trí qua lại) của các phân tử, nguyên tử hay các hạt làm nên vật chất và lan truyền trong vật chất như các sóng. Âm thanh là sự lan truyền áp suất không khí trong không gian.Âm thanh, giống như nhiều sóng, được đặc trưng bởi tần số, bước sóng, chu kì, biên độ và vận tốc lan truyền (tốc độ âm thanh).
- Đặc trưng vật lý của âm thanh:

- Tần số là số dao động mà nguồn âm có thể thực hiện được trong 1 giây. Đơn vị tần số là Hertz(Hz). Tần số âm được xem là đại lượng quan trọng nhất của âm thanh.
- Cường độ âm (I) là năng lượng được sóng âm truyền qua mỗi đơn vị diện tích được đặt vuông góc với phương truyền sóng trong mỗi đơn vị thời gian. Đơn vị đo cường độ âm là W/m². Mức cường độ âm $L = log\frac{I}{I_0}$ (I_0 là cường độ âm chuẩn) đo bằng đơn vị Ben(B) hay trong thực tế người ta thường dùng đơn vị dexiben(dB) do giá trị đại lượng này khá nhỏ.
- Đồ thị giao động là tập hợp các đồ thị dao động của tất cả các họa âm trong cùng một nhạc âm.

- Đặc trưng sinh lý của âm thanh: Cảm giác về âm thanh không chỉ phụ thuộc vào các đặc trưng vật lý của âm thanh mà còn phụ thuộc vào sinh lý của tai. Mỗi đặc trưng sinh lý phụ thuộc vào một đặc trưng vật lý nhất định, bao gồm:
  - Độ cao - tương ứng với tần số.
  - Độ to - tương ứng với mức cường độ âm.
  - Âm sắc - tương ứng với đồ thị dao động.
- Số hóa âm thanh: [6] Công nghệ được sử dụng để ghi chép, lưu trữ, tạo ra, chỉnh sửa và tái tạo âm thanh bằng cách sử dụng các tín hiệu được mã hóa dưới dạng bit (0 và 1). Sau đó có thể xử lý được trong máy tính.
- Lấy mẫu âm thanh là quá trình tạo ra tín hiệu âm thanh rời rạc hoặc tín hiệu số từ tín hiệu âm thanh dạng tương tự.
  - Tốc độ lấy mẫu [7] (Sampling Rate) hay Tần số lấy mẫu (Sampling Frequency): là số lượng mẫu (Sample) được lấy trong mỗi giây. Khoảng thời gian mà quá trình lấy mẫu lặp lại gọi là chu kỳ lấy mẫu. Ví dụ: Tần số lấy mẫu là 16000Hz nghĩa là 1 giây ta thu được 16000 mẫu và 1 mili giây ta thu được 16000/1000 = 16 mẫu.
  - Theo định lý lấy mẫu Nyquist–Shannon [11]: Để đảm bảo thu được tín hiệu số hoá trung thực trong mức cho phép với tín hiệu lấy mẫu, tần số lấy mẫu phải tối thiểu lớn hơn hai lần tần số lớn nhất xuất hiện trong tín hiệu lấy mẫu. Các âm thanh số hóa tiêu chuẩn thường được lấy mẫu với các tần số từ 6000 đến 192000Hz, và thường là các tần số 6000, 8000,

11025 , 22050, 44100, 48000, 96000Hz. Tần số tiêu chuẩn mà con người nghe được trong khoảng 16-20000Hz. Tai người đặc biệt nhạy cảm với những tần số trong tín hiệu tiếng nói chứa thông tin phù hợp nhất với việc liên lạc (những tần số xấp xỉ 250 – 4000Hz). Do vậy, theo định lý lấy mẫu Shannon, tần số lấy mẫu cho tiếng nói chỉ cần cỡ 11025Hz, 16000Hz hoặc 22050 Hz. Nếu lấy mẫu với tần số quá cao thì số lượng mẫu thu được rất lớn và gây khó khăn hơn trong việc xử lý chúng.

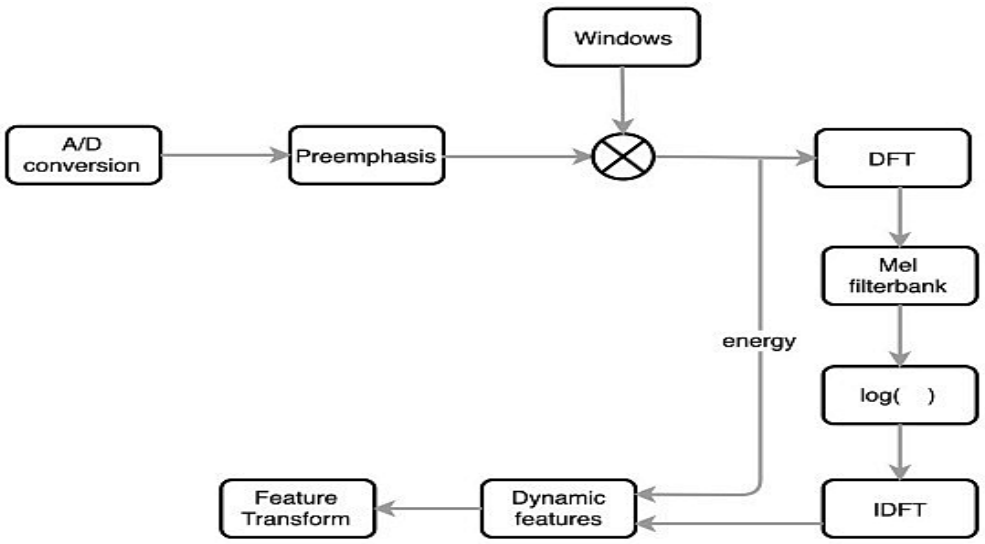
2.4.2. Approximate Nearest Neighbors (ANN)

Nearest neighbor search (NNS) [9] hay tìm kiếm lân cận là bài toán tối ưu hóa để tìm điểm gần nhất (hoặc giống nhất) với một điểm nhất định. Mức độ tìm kiếm thường được thể hiện dưới dạng hàm không giống nhau: các đối tượng càng ít giống nhau, giá trị hàm càng lớn dẫn tới tốn thời gian, tốn bộ nhớ và độ trễ thấp.

ANN [8] là một biến thể của NNS giúp phỏng đoán điểm gần nhất với một điểm nhất định, đôi lúc có thể không đảm bảo là sẽ trả về điểm gần nhất thực tế nhưng bù lại cải thiện được tốc độ và tiết kiệm bộ nhớ.

2.4.3. MFCCs

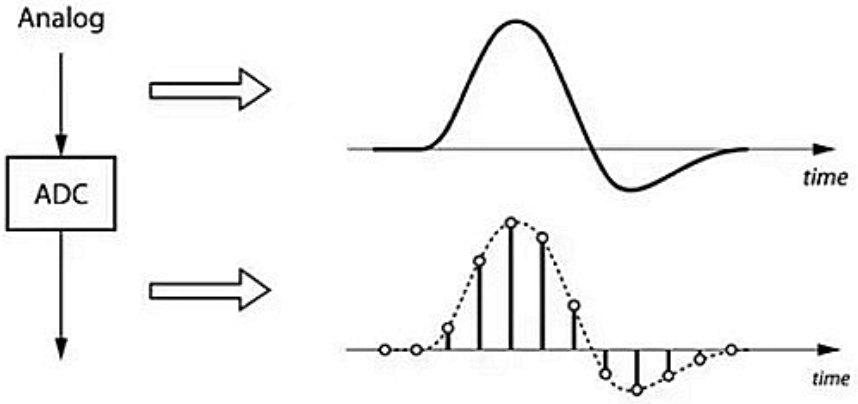
[17] Một trong những phương pháp lấy đặc trưng âm thanh dựa trên phổ tần số phổ biến trong các hệ thống nhận dạng giọng nói, tổng hợp tiếng nói.



Hình 2.2. Quy trình biến đổi MFCC

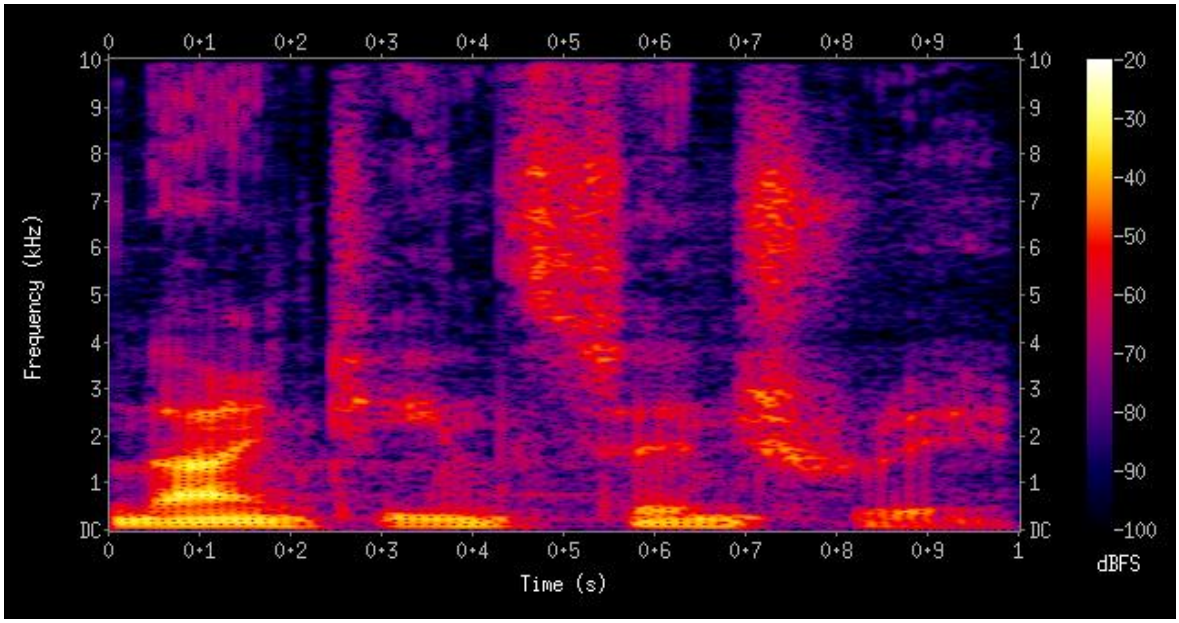
Quá trình tạo ra MFCC:

- A/D Conversion: Âm thanh là dạng tín hiệu liên tục, còn máy tính làm việc với các con số rời rạc. Do đó, ta cần lấy mẫu với một tần số lấy mẫu xác định để chuyển từ dạng tín hiệu liên tục về dạng rời rạc. Ví dụ: sample_rate = 16000 ⇒ trong 1 giây lấy 8000 giá trị.



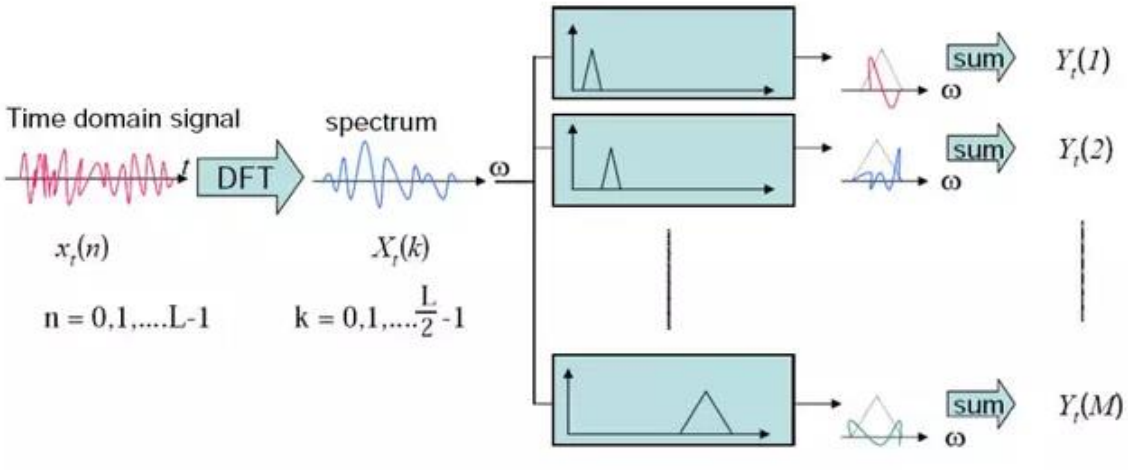
Hình 2.3. Quá trình A/D Conversion

- Pre-emphasis: Kích các tín hiệu của âm thanh ở các tần số thấp và cao để lấy thông tin về âm vị. Do đặc điểm cấu tạo thanh quản và các bộ phận phát nên tiếng nói của chúng ta có đặc điểm: các âm ở tần số thấp có mức năng lượng cao, các âm ở tần số cao lại có mức năng lượng khá thấp. Trong khi đó, các tần số cao này vẫn chứa nhiều thông tin về âm vị.
- Windowing: bao gồm việc cắt dạng sóng âm thanh thành các sliding frames.
- Discrete Fourier Transform - DFT: theo từng frame, ta áp dụng theo công thức: $X[k] = \sum_{n=0}^{N-1} x[n]exp(-j\frac{2\pi}{N}kn)$. Mỗi frame ta thu được 1 list các giá trị độ lớn (magnitude) tương ứng với từng tần số từ 0→N. Áp dụng với tất cả frame thì ta thu đc 1 Spectrogram. Trục x là trục thời gian (tương ứng với thứ tự các frame) trục y là dải tần số từ 0→10000Hz, giá trị magnitude tại từng tần số được thể hiện bằng màu sắc. Qua quan sát spectrogram này, ta nhận thấy các tại các tần số thấp thường có magnitude cao, tần số cao thường có magnitude thấp.



Hình 2.4. Spectrogram

- Mel filterbank: Bình phương các giá trị trong DFT trên ta thu đc DFT power spectrum(phổ công suất). Sau đó áp dụng bộ lọc Mel-scale filter trên từng khoảng tần số. Giá trị output của từng filter là năng lượng dải tần số mà filter đó bao phủ được. Ta thu được Mel-scale power spectrum. Mel filterbank trả về phổ công suất của âm thanh, hay còn gọi là phổ năng lượng.



Hình 2.5. Quy trình Mel filterbank

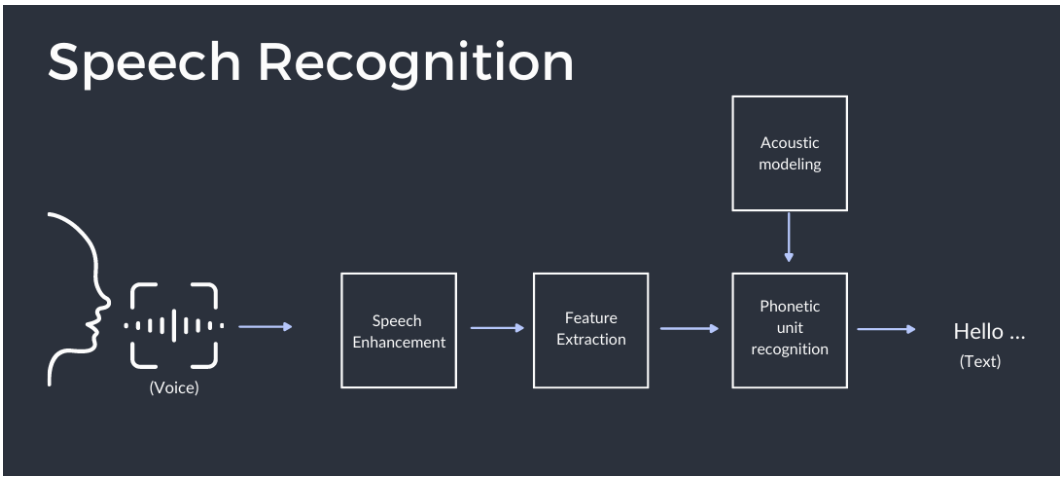
- Log: tính log trên Mel-scale power spectrum để giảm các biến thể âm thanh không đáng kể để nhận dạng giọng nói
- Inverse DFT - IDFT: biến đổi Fourier ngược về miền thời gian ta thu được Cepstrum. Bước này giúp mô hình nhận dạng không bị phụ thuộc vào cao độ

giọng của từng người. Để tính MFCC, ta chỉ cần lấy 12 giá trị đầu tiên trong Cepstrum. Từ đó thu được 12 Cepstrum features.

- MFCC: từ 12 Cepstrum features ta tính thêm power feature 13 theo công thức $Energy = \sum_{t=t1}^{t2} x^2[t]$, sau đó đạo hàm bậc 1 (theo thời gian) với 13 feature đầu tiên bằng công thức $d(t) = \frac{c(t+1)-c(t-1)}{2}$, đạo hàm này chứa thông tin về sự thay đổi từ frame thứ t đến frame thứ t + 1. Tương tự, 13 giá trị cuối của MFCC là sự thay đổi d(t) theo thời gian - đạo hàm của d(t), đồng thời đạo hàm bậc 2 của công thức c(t) là công thức $b(t) = \frac{d(t+1)-d(t-1)}{2}$. Vậy từ 12 cepstrum feature và 1 power feature ta đạo hàm 2 lần và thu được 39 feature. Đây là MFCC feature.

2.4.4. Speech Recognition

Speech Recognition [12] hay nhận dạng giọng nói là một công nghệ cho phép máy tính nhận dạng, hiểu và dịch lời nói của con người thành văn bản. Công nghệ này sử dụng quá trình xử lý ngôn ngữ tự nhiên (natural language processing - NLP) và máy học (Machine Learning - ML) để dịch lời nói của con người.



Hình 2.6. Mô hình cách thức hoạt động của Speech recognition

Cách thức hoạt động của speech recognition:

- Micro chuyển các rung động của giọng nói một người thành tín hiệu điện.
- Máy tính hoặc hệ thống tương tự chuyển tín hiệu đó thành tín hiệu số
- Đơn vị tiền xử lý tăng cường tín hiệu giọng nói trong khi giảm thiểu nhiễu.

- Phần mềm speech recognition phân tích tín hiệu bằng cách sử dụng mô hình âm học (acoustic modeling) để đăng ký các âm vị (phonemes), các đơn vị âm thanh lời nói riêng biệt đại diện và phân biệt từ này với từ khác.
- Các âm vị được xây dựng thành các từ và câu dễ hiểu bằng cách sử dụng mô hình ngôn ngữ (language modeling).

2.4.5. Độ đo Cosine similarity

Độ tương tự cosin [22] là cách đo độ tương tự giữa hai vector khác 0 của một không gian tích vô hướng. Độ tương tự này được định nghĩa bằng giá trị cosine của góc giữa 2 vector, và cũng là tích vô hướng của cùng các vectơ đơn vị để cả hai đều có chiều dài 1.

Cosine của hai vectơ khác không được suy ra bằng cách sử dụng công thức tích vô hướng Euclid [21]:

$$A.B = ||A|| \, ||B|| cos(\theta)$$

Cho hai vectơ chứa các thuộc tính, A và B, độ tương tự cosine, cos(θ), được thể hiện bằng tích vô hướng và độ lớn là

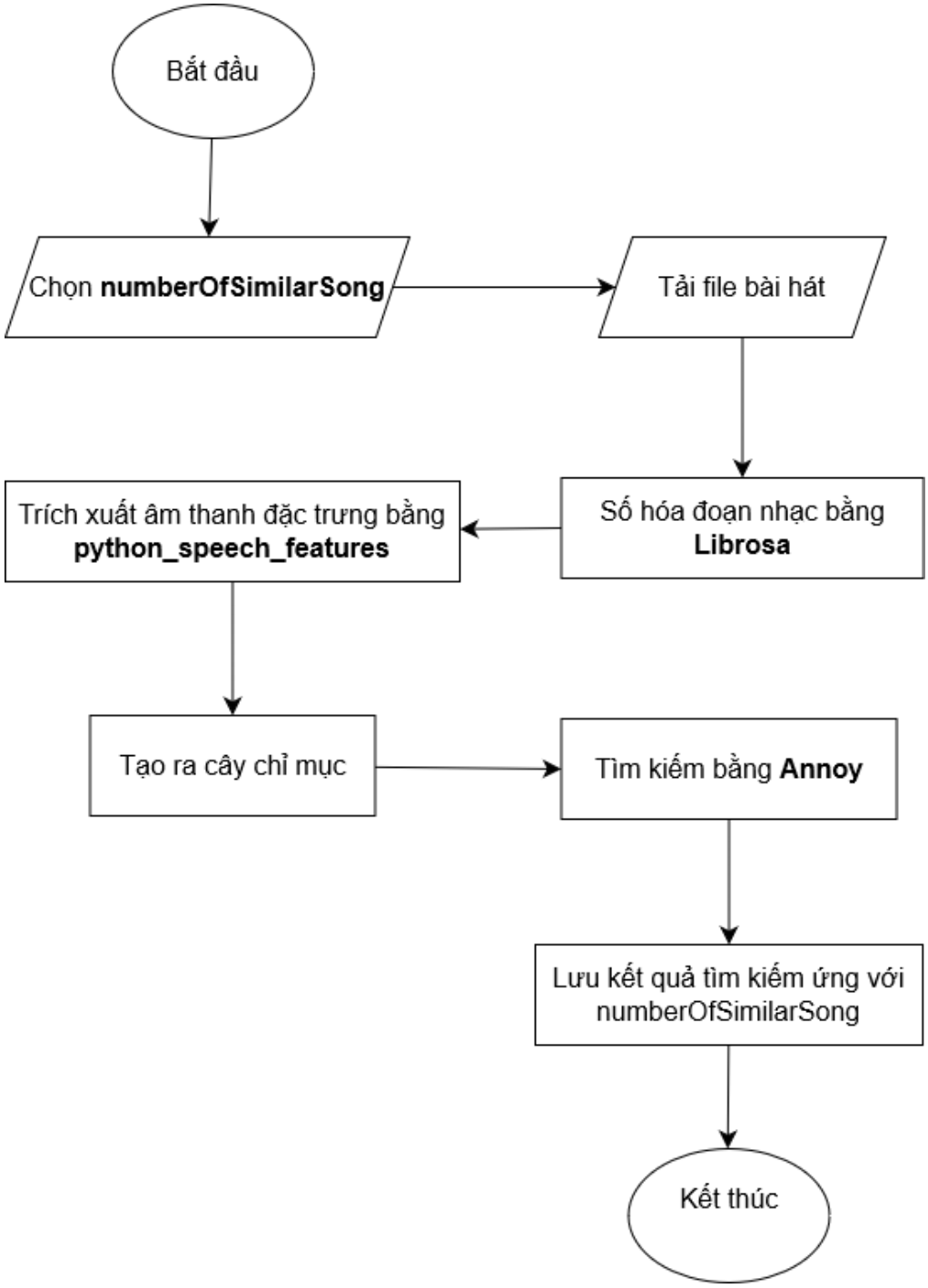
$$similarity = cos(\theta) = \frac{A.B}{||A|| \, ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Với A_i và B_i là thành phần của vector A và B tương ứng

CHƯƠNG 3: PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Phương pháp nghiên cứu

3.1.1. Phương pháp MFCCs



Hình 3.1. Mô hình sử dụng trong đề tài của phương pháp MFCCs

Bảng 3.1 Các tham số sử dụng trong phương pháp MFCC

Tham số	Ý nghĩa	Giá trị sử dụng
y	Giá trị biên độ đã được số hóa	
sr	sampling rate - tốc độ lấy mẫu	11025, 16000
song	Đường dẫn đến tệp bài hát	
nfilt	Độ dài của mỗi MFCC	10, 11, 12
winsteps	Lấy mẫu theo 0.02s cho một MFCC	0.02
feat	Mảng chứa các đặc trưng MFCC trong 0.02s có độ dài nfilt = 10	
i	Vị trí của MFCC	
nbStep	Số MFCC được gộp	10
maxLen	Số chiều của vector	maxLen =nfilt x nbStep
cropFeat	Vector trả về	
features	Mảng lưu các vector	
songs	Mảng lưu lại các bài hát	
dataDirectorySearch	Đường dẫn tới thư viện nhạc	
featuresPk	Đường dẫn lưu tệp vector	
songsPk	Đường dẫn lưu tệp bài hát	
t	Một chỉ mục đọc ghi	
dimensionVector	Số chiều vector	
lenFeatures	Độ dài của mảng features	
vectorNumber	Vector thứ i	
musicAnn	Đường dẫn đến tệp lưu cây chỉ mục	
t.add_item(i, vectorNumber)	Thêm vào i một vectorNumber	
t.build(dimensionVector)	Xây dựng một rừng cây chỉ mục với số chiều vector	
t.save('music.ann')	Lưu lại cây chỉ mục	
result =u.get_nns_by_item(crop_feat, n=5)	Thực hiện tìm kiếm vector crop_feat và trả về 5 chỉ mục có kết quả gần nhất với vector.	
result_songs =[songs[k] for k in result]	Tên các bài hát của kết quả tìm kiếm	
results.append(result_songs)	Đưa tên các bài hát vào một mảng results	

results = np.array(results).flatten()	Đưa mảng results về một chiều	
numberOfSimilarSong	Số bài hát giống với bài hát đưa vào	

Các bước triển khai

Bước 1: Số hóa âm thanh

```
y, sr = librosa.load(song,sr=16000)
```

Hình 3.2. Số hóa âm thanh

Bước 2: Trích xuất âm thanh đặc trưng của âm thanh với thư viện python_speech_features

```
def extractFeature(y, sr = 16000, nfilt = 10, winsteps = 0.02):
    try:
        feat = mfcc(y, sr, nfilt=nfilt, winstep =winsteps)
        return feat
    except:
        raise Exception("Extraction features error")
```

Hình 3.3. Trích xuất âm thanh đặc trưng MFCC

Bước 3: Sau khi trích xuất âm thanh đặc trưng ở bước 2, trong 0.02 giây ta thu được 1 MFCC. Vậy trong 1 giây ta thu được 50 MFCC. Tuy nhiên việc 50 MFCC quá nhỏ để thể hiện đặc trưng của đoạn âm thanh 1s. Vì vậy ta gộp 10 MFCC thành 1 vector 100 chiều ứng với mỗi đoạn âm thanh dài 0.2s

```
def cropFreature(feat, i = 0, nbStep = 10, maxLen = 100):
    cropFeat = np.array(feat[i : i + nbStep]).flatten()
    print(cropFeat.shape)
    cropFeat = np.pad(cropFeat, (0, maxLen - len(cropFeat)),mode='constant')
    return cropFeat
```

Hình 3.4. Gộp MFCC thành 1 vector với mỗi đoạn âm thanh dài 0.2s

Bước 4: Tiền xử lý bài hát trong thư viện Nhạc với hàm extractFeature và cropFeature

```
def find(dataDirectorySearch, features=[], songs=[]):
    for song in tqdm(os.listdir(dataDirectorySearch)):
        song = os.path.join(dataDirectorySearch, song)
        y, sr = librosa.load(song, sr=16000)
        feat = extractFeature(y, sr, opt.nfilt)
        for i in range(0, feat.shape[0], 5):
            features.append(cropFreature(feat, i, nbStep=10, maxLen=10*opt.nfilt))
            songs.append(song)

features = []
songs = []
find(opt.dataDirectorySearch, features, songs)
```

Hình 3.5. Tiền xử lý các bài hát trong thư viện

- Vòng lặp ngoài: Lấy bài hát trong thư viện và số hóa sau đó trích xuất âm thanh đặc trưng âm thanh MFCC của từng bài hát
- Vòng lặp trong: Gộp 10 MFCC trong mảng thành vector 10 x nflit chiều đại diện cho đoạn âm thanh dài 0.2 giây như ở bước 3. Lưu các vector trong mảng feature và lưu mảng các bài hát trong mảng song

Bước 5: Lưu lại các mảng features và songs

```
pickle.dump(features,open(featuresPk, 'wb'))
pickle.dump(songs, open(songsPk, 'wb'))
```

Hình 3.6. Lưu lại các mảng

Bước 6: Tạo cây chỉ mục lưu trữ các vector bằng thư viện annoy

```
t = AnnoyIndex(dimensionVector)
lenFeatures=len(features)
for i in range(lenFeatures):
    vectorNumber = features[i]
    t.add_item(i, vectorNumber)

t.build(opt.dimensionVector)

musicAnn=str(opt.dimensionVector) + '_' + str(opt.nfilt)+'.ann'
t.save(musicAnn)
ticks = time.time()
ticks_end = time.time()-ticks
```

Hình 3.7. Tạo cây chỉ mục

Bước 7: Số hóa và trích xuất đặc trưng âm thanh MFCC của file audio

```
try:
    song = os.path.join(opt.dataDirectory, opt.fileName+'.mp3')
    y, sr = librosa.load(song,sr=16000)
    feat = extractFeature(y)
except:
    song = os.path.join(opt.dataDirectory, opt.fileName+'.wav')
    y, sr = librosa.load(song,sr=16000)
    feat = extractFeature(y)
```

Hình 3.8. Số hóa và trích xuất đặc trưng âm thanh của file audio

Bước 8: Tải lại cây chỉ mục và tìm kiếm những bài hát gần giống nhất

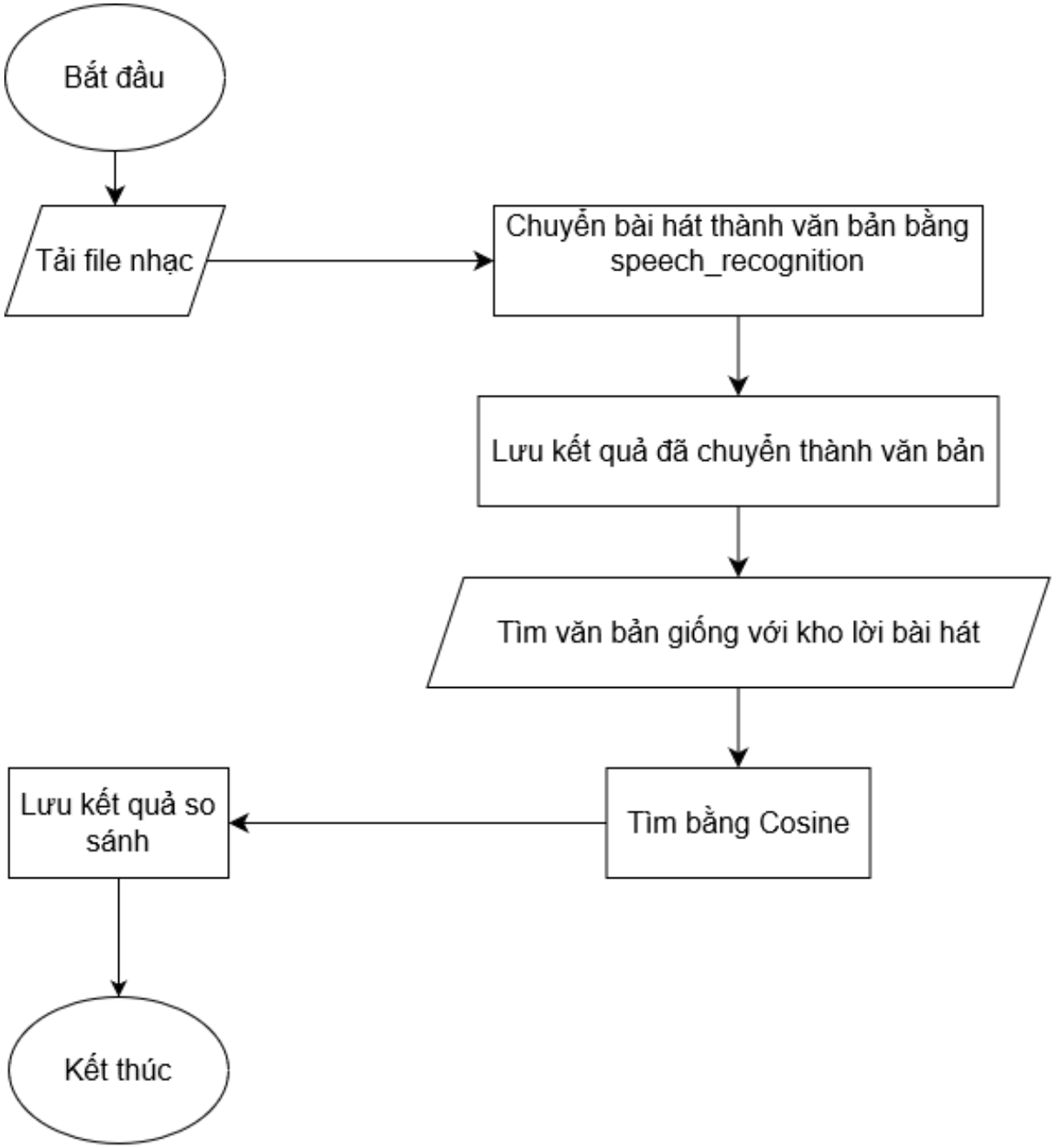
```
results = []

u = AnnoyIndex(opt.dimensionVector)
u.load(musicAnn)
for i in range(0, feat.shape[0], 5):
    cropFeat = cropFreature(feat, i , nbStep=10,maxLen=10*opt.nfilt)
    result = u.get_nns_by_vector(cropFeat, opt.numberOfSimilarSong+5)
    result_songs = [songs[k] for k in result]
    results.append(result_songs)

results = np.array(results).flatten()
```

Hình 3.9. Tìm kiếm bài hát gần giống nhất

3.1.2. Phương pháp Google API



Hình 3.10. Mô hình của phương pháp Google API

Bảng 3.2 Các tham số sử dụng trong phương pháp Google API

Tham số	Ý nghĩa
filePath	Đường dẫn tới file audio
audio = r.record(source)	Trích xuất dữ liệu âm thanh từ tập tin
filenameserult	Đường dẫn tới vị trí lưu file
file.writelines (audioSpeechRecognitionByGoogle(filePath))	file.writelines lưu phần text đã tách từ file audio
a_file	Đường dẫn đến file A
a_vals	các giá trị được thêm vào từ a_file
testSimilarity	hàm so sánh độ giống nhau giữa a_file và b_file
fileDir	đường dẫn tới thư viện lời nhạc
feature	mảng chứa các giá trị Cosine
song	mảng chứa các bài hát
readText =nlp(pathlib.Path(file).read_text(encoding="utf-8"))	đọc nội dung tệp file bằng phương thức read_text của thư viện pathlib
arr = readText.similarity(readFileTest)	so sánh độ tương đồng của file readText và readFileTest

Các bước triển khai

Bước 1: Chuyển file âm thanh thành text bằng thư viện Speech Recognition

```
def audioSpeechRecognitionByGoogle(filePath):
    with sr.AudioFile(filePath) as source:
        audio = r.record(source)

    try:
        temp = str(r.recognize_google(audio, language='vi-VN').lower())
        return temp

    except sr.UnknownValueError:
        print(filePath + " Google Speech Recognition could not understand audio")
```

Hình 3.11. Hàm chuyển file audio thành text

Bước 2: Lưu dữ liệu file text được chuyển vào một file tạm

```
filenameresult_ = opt.fileNameResult + '.txt'
file = open(filenameresult_, 'w+', encoding='utf-8')
file.writelines(audioSpeechRecognitionByGoogle(filePath))
file.close()
```

Hình 3.12. Lưu đoạn text trên vào file tạm

Bước 3: Tìm file lời nhạc giống với file tạm trong thư viện lời bài hát bằng Cosine_similarity của thư viện sklearn (hình 3.19) hay bằng similarity của thư viện Spacy (hình 3.20)

```
a_file = filenameresult_
b_file = ''
a_vals = Counter(a_file)
b_vals = Counter(b_file)
words  = list(a_vals.keys() | b_vals.keys())
a_vect = [a_vals.get(word, 0) for word in words]
len_a  = sum(av*av for av in a_vect) ** 0.5
def testSimilarLyric(fileDir,features=[], song=[]):
    for fileSong in os.listdir(opt.fileLyric):
        f = os.path.join(fileDir, fileSong)
        if os.path.isfile(f):
            b_file = f
            b_vals = Counter(b_file)
            b_vect = [b_vals.get(word, 0) for word in words]
            len_b  = sum(bv*bv for bv in b_vect) ** 0.5
            dot    = sum(av*bv for av,bv in zip(a_vect, b_vect))
            cosine = dot / (len_a * len_b)
            cosine_sim = cosine_similarity([a_vect], [b_vect])
            features.append(cosine)
            song.append(f)
            print("Cosine:", cosine)
            print(f)

    print("max Cosine:", max(features))

features=[]
song=[]
testSimilarLyric(opt.fileLyric,features,song)
```

Hình 3.13. Tìm file lời giống file tạm bằng Cosine của sklearn

```
readText = nlp(pathlib.Path(filenameresult_).read_text(encoding="utf-8"))
def testSimilarLyric(fileDir, features=[], fileSong=[]):
    for filename in os.listdir(fileDir):
        f = os.path.join(fileDir, filename)
        if os.path.isfile(f):
            readFileTest = nlp(pathlib.Path(f).read_text(encoding="utf-8"))
            arr = readText.similarity(readFileTest)
            features.append(arr)
            print(arr)
            fileSong.append(f)
            print(f)

    print("max similar:", max(features))

features = []
fileSong = []


testSimilarLyric(opt.fileLyric,features,fileSong)
```

Hình 3.14. Tìm file lời giống file tạm bằng similarity của Spacy

Bước 4: Sắp xếp độ tương đồng từ cao đến thấp

```
def sortAscending(features):

    lenth = len(features)

    for i in range(0, lenth - 1):
        for j in range(i + 1, lenth):
            if (features[i] < features[j]):
                # Hoán đổi vị trí
                tmp = features[i]
                tmp2 = song[i]
                features[i] = features[j]
                song[i] = song[j]
                features[j] = tmp
                song[j] = tmp2

    #a = np.stack((features, song), axis = 1)
    return features, song
```

Hình 3.15. Hàm sắp xếp độ tương đồng

3.2. Phương tiện nghiên cứu

3.2.1. Công cụ Visual Studio Code

Visual Studio Code [20] hay VS Code là một trong những trình soạn thảo mã nguồn phổ biến được các lập trình viên sử dụng. VS Code hỗ trợ đa dạng các chức năng Debug, Git, Syntax Highlighting. VS Code hỗ trợ nhiều ngôn ngữ lập trình như HTML, CSS, JavaScript, C++, Python,... Và hỗ trợ đa nền tảng (Windows, Mac, Linux) và tích hợp đầy đủ các tính năng và khả năng mở rộng.

3.2.2. Python

Python [19] là một ngôn ngữ lập trình bậc cao sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Ưu điểm dễ đọc, dễ học và dễ nhớ.

3.2.3. Anaconda

Anaconda [14] là bản phân phối các ngôn ngữ lập trình python và R cho tính toán khoa học nhằm mục đích đơn giản hóa việc quản lý và triển khai gói.

3.2.4. Jupyter Notebook

Jupyter Notebook [18] là một nền tảng tính toán khoa học mã nguồn mở cho phép bạn có thể sử dụng để tạo và chia sẻ các tài liệu có chứa code trực tiếp, phương trình, trực quan hóa dữ liệu và văn bản tường thuật. Jupyter Notebook được coi là môi trường tương tác đa ngôn ngữ với môi trường dễ sử dụng, và cho phép người dùng xem kết quả của code in-line(mã inline) mà không cần phụ thuộc vào các thành phần khác của code.

3.2.5. Annoy

Annoy (Approximate Nearest Neighbors Oh Yeah) [16] là một thư viện C ++ với các ràng buộc Python để tìm kiếm các điểm trong không gian gần với một điểm truy vấn nhất định.

3.2.6. Cách cài đặt hệ thống và quy trình chạy

- Tải công cụ Visual Studio Code
- Cài đặt môi trường Anaconda
- Cài đặt ngôn ngữ lập trình Python vô môi trường
- Import các thư viện cần thiết trong quá trình lập trình
- Sau khi lập trình hoàn tất thì tiến hành training trên môi trường dòng lệnh

CHƯƠNG 4: KẾT QUẢ

4.1. Mô tả dữ liệu

4.1.1. Mô tả dữ liệu thực nghiệm

Dữ liệu thực nghiệm trong phương pháp trích xuất âm thanh đặc trưng MFCC bao gồm

- Thư viện 200 bài hát: chọn ngẫu nhiên 45 bài tương ứng với 15 bài mỗi tập dữ liệu dưới 30 giây, từ 30 giây đến 60 giây, trên 60 giây. Còn lại để so sánh đặc trưng.
- Thư viện 84 bài nhạc Cover: Chọn ngẫu nhiên 30 bài tương ứng với 10 bài với mỗi tập dữ liệu dưới 30 giây, từ 30 giây đến 60 giây, trên 60 giây. Còn lại không dùng.

Dữ liệu thực nghiệm trong phương pháp tách lời bằng Google API với thư viện speech recognition bao gồm:

- 100 bài hát đã cắt ra dài 1 phút. Lấy ngẫu nhiên 20 bài để thực nghiệm.
- Thư viện 103 lời bài hát

4.1.2. Mô tả độ đo của thực nghiệm

Tính độ tương đồng văn bản dựa trên độ đo Cosine là phương pháp tương đối đơn giản và cho kết quả với độ chính xác cao. Các văn bản được biểu diễn theo mô hình túi từ (bag-of-words). Trong mô hình này, một văn bản được thể hiện như là một túi các từ của nó, không theo ngữ pháp và thứ tự từ. Mỗi văn bản được được tách ra thành các từ hay cụm từ (n-gram từ), sau đó được bỏ vào trong túi. Mỗi từ hay cụm từ được tính tổng số lần xuất hiện và được tạo thành vector n chiều, trong đó n là số phần tử trong danh sách chung các từ hay cụm từ khác nhau của các văn bản. Sau khi chuyển hai văn bản thành vector là A và B , ta có thể sử dụng độ đo Cosine để tính toán độ tương đồng của các văn bản. Công thức tính độ tương đồng Cosine ở mục 2.4.5. Thuật toán tính độ tương đồng Cosine như sau:

- Đầu vào: 2 văn bản A và B.
- Xử lý: Thực hiện qua các bước sau:

- Tiền xử lý (tách từ, tạo danh sách từ vựng, ...)
- Xây dựng tập từ vựng chung T = {t1, t2....}
- Mô hình hóa văn bản thành vector: Dựa vào T ta tạo vector tần số từ của A và B lần lượt là a và $\vec{b}$ (bằng cách tính TF*IDF)
- Từ 2 vector tần số từ tương ứng với 2 văn bản, tính cos góc giữa 2 vector bằng cách sử dụng công thức tính độ tương đồng Cosine theo ở mục 2.4.5

- Đầu ra: Độ tương đồng giữa 2 văn bản A và B.

4.1.3. Kịch bản

Thực hiện trích xuất âm thanh đặc trưng MFCC và tìm kiếm bài hát bằng thuật toán ANN. Chia làm 2 kịch bản 1,2 để thực nghiệm với tốc độ lấy mẫu là sr = 11025, sr = 16000.

Bảng 4.1. Các thời gian chạy thực nghiệm của hệ thống

Thời gian	Dưới 30s	Từ 30s đến 60s	Từ 60s đến 120s
Số bài hát thực nghiệm	25	25	25

Với từng tốc độ lấy mẫu sr = 11025 và sr = 16000, chia các hệ số bao gồm:

- nflit: độ dài của mỗi MFCC gồm các độ đo: 10, 11, 12.
- dimensionVector: số chiều vector gồm các độ đo 100, 110, 120.
- numberOfSimilarSong: số bài hát gần giống nhất bao gồm 1, 10, 20 .

Kịch bản 3: Phương pháp tách lời bằng Google API với thư viện speech recognition dùng độ đo cosine_similarity của thư viện sklearn và thư viện spacy

4.2. Kết quả nghiên cứu phương pháp MFCCs

Tính độ chính xác:

$$\text{Độ } chính\ xác\ = \frac{Tổng\ số\ bài\ hát\ đúng}{Tổng\ số\ bài\ hát\ thực\ nghiệm} * 100\%$$

Thời gian chạy thực nghiệm bao gồm thời gian tiền xử lý, thời gian xây dựng hệ thống và thời gian tìm kiếm.

```
Thời gian tiền xử lý (giây): 372.828125 seconds
Thời gian tạo cây chỉ mục (giây):  0.0000000 seconds
Thời gian bắt đầu phiên chạy: 2023-05-04 07:09:03.964674
Thời gian kết thúc phiên chạy: 2023-05-04 07:12:11.218786
```

Hình 4.1. Hiển thị thời gian chạy của hệ thống

- Thời gian tiền xử lý bằng hàm time.process_time(): trả về thời gian được tính bằng giây.
- Thời gian bắt đầu phiên chạy được được xử lý bằng hàm datetime.now(): Hàm trả về một datetime đối tượng mới với ngày và dấu thời gian cục bộ hiện tại.
- Thời gian kết thúc phiên chạy được được xử lý bằng hàm datetime.now(): Hàm trả về một datetime đối tượng mới với ngày và dấu thời gian cục bộ hiện tại.

4.2.1. Kết quả kiểm thử kịch bản 1 với tốc độ lấy mẫu sr = 16000

Bảng 4.2. Thời gian tiền xử lý và tạo cây với tốc độ lấy mẫu sr = 16000

Độ dài của mỗi MFCC (nflit)	Số chiều Vector (dimensionVector)	Thời gian tiền xử lý (giây)
10	100	3776.837439
11	110	3941.633481
12	120	4211.208353

Thời gian trung bình tiền xử lý 1 bài hát thấp nhất là 3776 giây với nflit = 10 và cao nhất là 4211 giây với nflit = 12

Bảng 4.3. Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 10 (nfilt = 10)

Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	410.960000	1	52
	403.680000	10	56
	417.323300	20	60

Từ 30s đến 60s	440.256300	1	60
	403.175634	10	60
	424.250000	20	60
Trên 60s	440.630075	1	60
	426.554630	10	64
	410.007500	20	64

Độ chính xác cao nhất 64% qua kết quả tìm kiếm với tập dữ liệu bài hát trên 60 giây - top 20 và thấp nhất là 52% với tập dữ liệu dưới 30 giây - top 1

Thời gian tiền xử lý ngắn nhất với từ 30 đến 60 giây - top 10 là 403.1775634 giây và dài nhất với trên 60 giây - top 1 là 440.630075 giây

Bảng 4.4. Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 11 (nfilt = 11)

Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	440.165231	1	0
	449.883750	10	4
	441.389375	20	4
Từ 30s đến 60s	426.526250	1	0
	436.362425	10	0
	465.036250	20	4
Trên 60s	406.632450	1	0
	425.467750	10	4
	450.200000	20	20

Độ chính xác cao nhất 20% qua kết quả tìm kiếm với tập dữ liệu bài hát trên 60 giây - top 20 và thấp nhất là 0% với tập dữ liệu dưới 30 giây - top 1, từ 30 đến 60 giây - top 1,10 và trên 60 giây - top 1

Thời gian tiền xử lý ngắn nhất với trên 60 giây - top 10 là 406.63245 giây và dài nhất với từ 30 đến 60 giây - top 20 là 440.630075 giây

Bảng 4.5. Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 12 (nfilt = 12)

Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	420.2536000	1	0
	445.0136525	10	4
	484.7537500	20	8
Từ 30s đến 60s	462.2634000	1	0
	486.2421000	10	12
	501.1050000	20	16
Trên 60s	452.7211000	1	0
	471.1120000	10	20
	487.8437500	20	32

Độ chính xác cao nhất 32% qua kết quả tìm kiếm với tập dữ liệu bài hát trên 60 giây - top 20 và thấp nhất là 0% với tập dữ liệu dưới 30 giây - top 1, từ 30 đến 60 giây - top 1 và trên 60 giây - top 1

Thời gian tiền xử lý ngắn nhất với dưới 30 giây - top 1 là 420.2536 giây và dài nhất với từ 30 đến 60 giây - top 20 là 501.105 giây

4.2.2. Kết quả kiểm thử kịch bản 2 với tốc độ lấy mẫu sr = 11025

Bảng 4.6. Thời gian tiền xử lý và tạo cây với tốc độ lấy mẫu sr = 11025

Độ dài của mỗi MFCC (nflit)	Số chiều Vector (dimensionVector)	Thời gian tiền xử lý (giây)
10	100	3763.347811
11	110	3909.454568
12	120	4091.613936

Thời gian trung bình tiền xử lý 1 bài hát thấp nhất là 3763 giây với nflit = 10 và cao nhất là 4091 giây với nflit = 12

Bảng 4.7. Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 10 (nfilt = 10)

Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	402.6750000	1	56
	406.6312500	10	56
	402.2525000	20	60
Từ 30s đến 60s	401.7518500	1	60
	396.5763889	10	60
	454.0486111	20	60
Trên 60s	462.5265850	1	56
	440.6300750	10	56
	426.6856250	20	56

Độ chính xác cao nhất 60% qua kết quả tìm kiếm với tập dữ liệu bài hát từ 30 đến 60 giây - top 1,10, 20 và thấp nhất là 56% với các tập dữ liệu còn lại

Thời gian tiền xử lý ngắn nhất với từ 30 đến 60 giây - top 10 là 396.5763889 giây và dài nhất với trên 60 giây - top 20 là 462.526585 giây

Bảng 4.8.Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 11 (nfilt = 11)

Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	436.0789623	1	0
	450.7714844	10	4
	433.6031250	20	8
Từ 30s đến 60s	420.110035	1	0
	403.953250	10	0
	469.576250	20	4
Trên 60s	403.750000	1	4
	460.253300	10	12
	463.647500	20	20

Độ chính xác cao nhất 20% qua kết quả tìm kiếm với tập dữ liệu bài hát trên 60 giây - top 20 và thấp nhất là 0% với các tập dữ liệu bài hát dưới 30 giây - top 1, từ 30 đến 60 giây - top 1,10.

Thời gian tiền xử lý ngắn nhất với trên 60 giây - top 1 là 403.75 giây và dài nhất với từ 30 đến 60 giây - top 20 là 469.57625 giây

Bảng 4.9. Kết quả độ chính xác của hệ thống tìm kiếm bài hát bằng đoạn nhạc có lời với độ dài của MFCC là 12 (nfilt = 12)

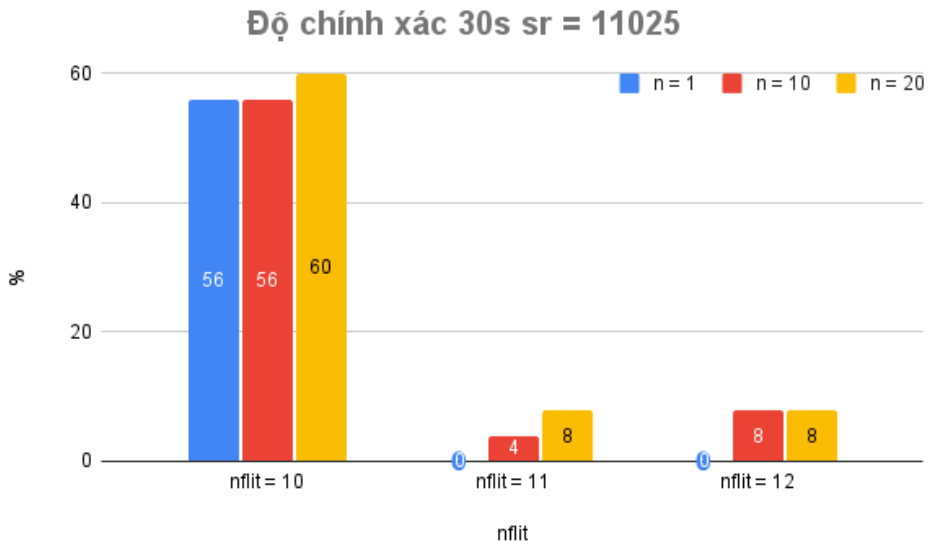
Thời gian bài hát	Thời gian tiền xử lý (giây)	Số bài hát đề xuất (numberOfSimilarSong)	Độ chính xác (%)
Dưới 30s	436.750325	1	0
	455.235105	10	8
	473.078750	20	8
Từ 30s đến 60s	440.263250	1	4
	450.123531	10	12
	506.094375	20	20
Trên 60s	428.421100	1	0
	443.640000	10	24
	458.007500	20	24

Độ chính xác cao nhất 24% qua kết quả tìm kiếm với tập dữ liệu bài hát trên 60 giây - top 20 và thấp nhất là 0% với các tập dữ liệu bài hát dưới 30 giây - top 1, trên 60 giây - top 1.

Thời gian tiền xử lý ngắn nhất với trên 60 giây - top 1 là 428.4211 giây và dài nhất với từ 30 đến 60 giây - top 20 là 506.094375 giây

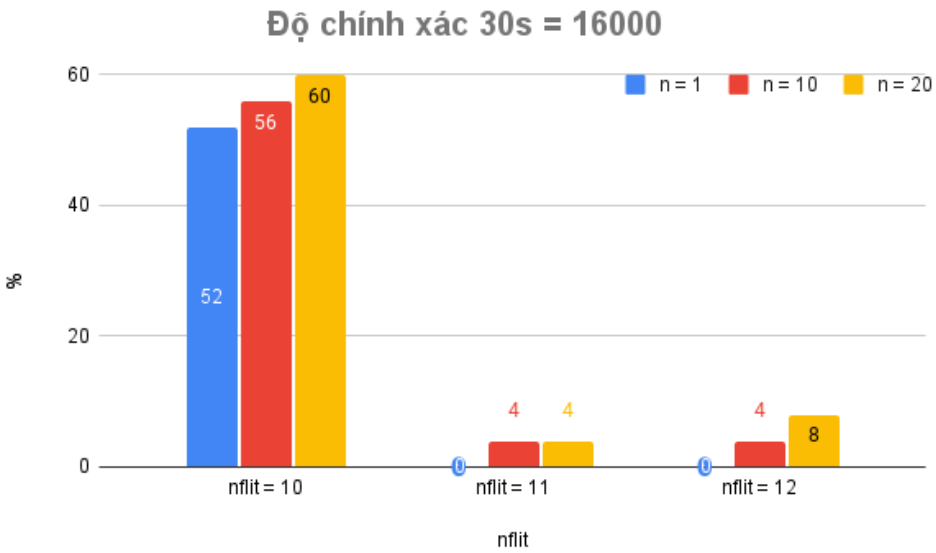
Nhận xét: Với tập dữ liệu trên 60s và nflit = 10 thì độ chính xác cao nhất. Và với tập dữ liệu dưới 30 giây và nflit = 11 thì độ chính xác thấp nhất.

4.2.3. Thống kê kết quả trên biểu đồ



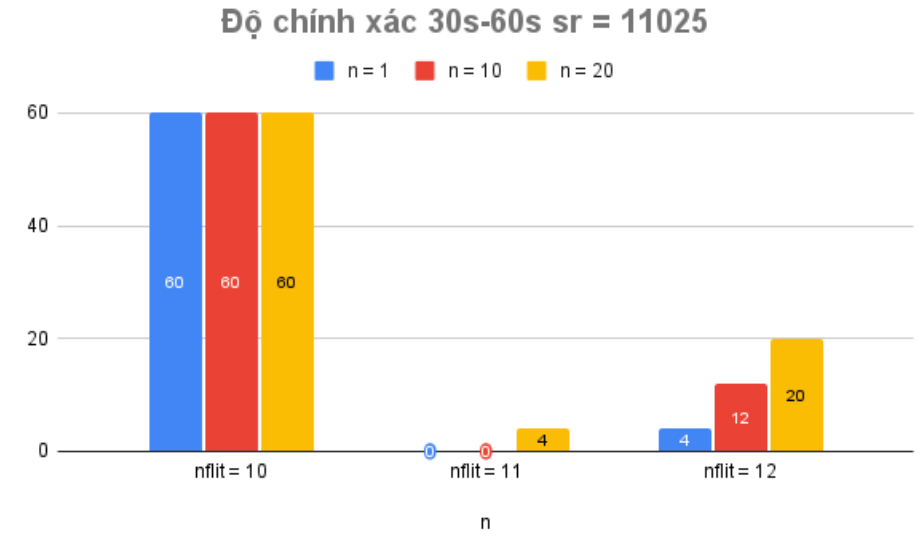
Hình 4.2. Biểu đồ độ chính xác với tập dữ liệu dưới 30s với sr = 11025

Hình 4.2 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát dưới 30 giây và sr = 11025. Kết quả tìm kiếm với top 20 có độ chính xác cao nhất 60% với nflit = 10 và thấp nhất 0 % với top 1 và nflit = 11 và nflit = 12



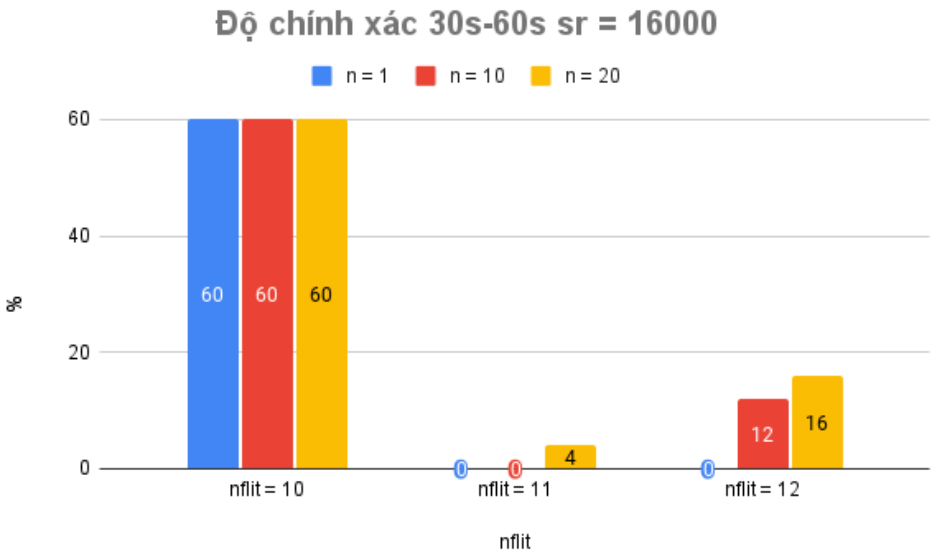
Hình 4.3. Biểu đồ độ chính xác với tập dữ liệu dưới 30 giây với sr = 16000

Hình 4.3 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát dưới 30 giây và sr = 16000. Kết quả tìm kiếm với top 20 có độ chính xác cao nhất 60% với nflit = 10 và thấp nhất 0 % với top 1 và nflit = 11 và nflit = 12



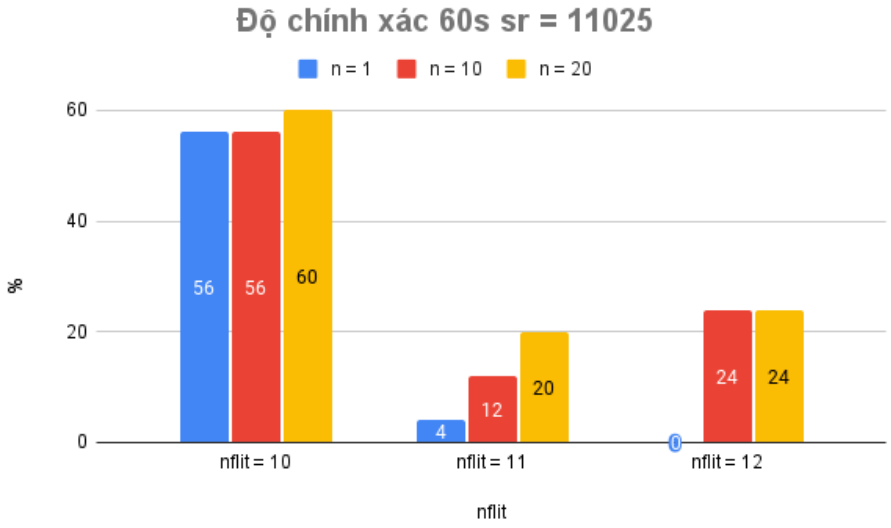
Hình 4.4. Biểu đồ độ chính xác với tập dữ liệu từ 30 đến 60s với sr = 11025

Hình 4.4 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát dưới từ 30 đến 60 giây và sr = 11025. Kết quả tìm kiếm với top 1,10,20 có độ chính xác cao nhất 60% với nflit = 10 và thấp nhất 0 % với top 1,10 và nflit = 11.



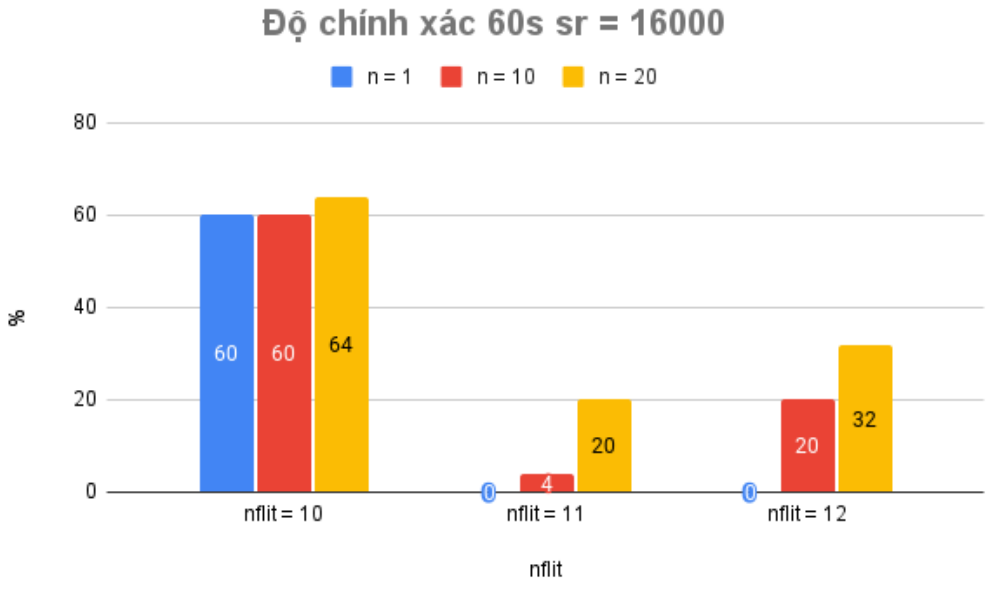
Hình 4.5. Biểu đồ độ chính xác với tập dữ liệu từ 30 đến 60s với sr = 16000

Hình 4.5 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát từ 30 đến 60 giây và sr = 11025. Kết quả tìm kiếm với top 1,10,20 có độ chính xác cao nhất 60% với nflit = 10 và thấp nhất 0 % với top 1,10 và nflit = 11, top 1 nflit = 12.



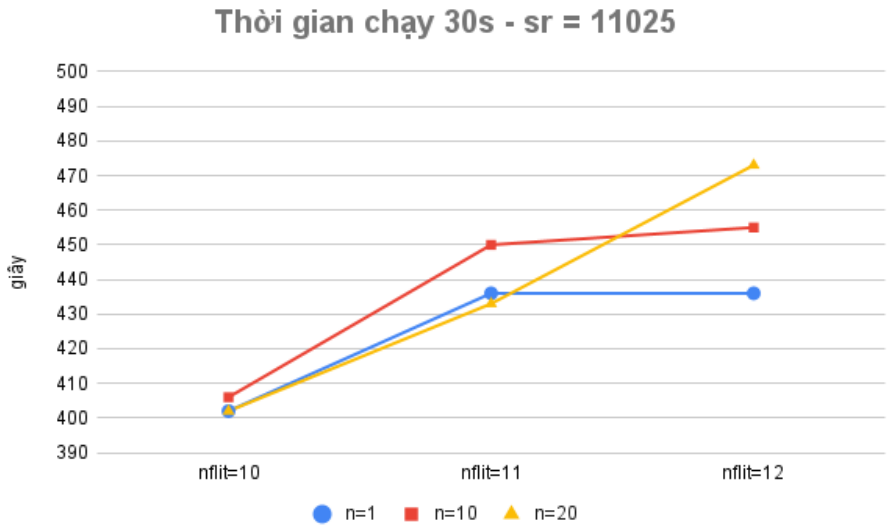
Hình 4.6. Biểu đồ độ chính xác với tập dữ liệu trên 60 giây với sr = 11025

Hình 4.6 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát trên 60 giây và sr = 11025. Kết quả tìm kiếm với top 20 có độ chính xác cao nhất 60% với nflit = 10 và thấp nhất 0 % với top 1 và nflit = 12.



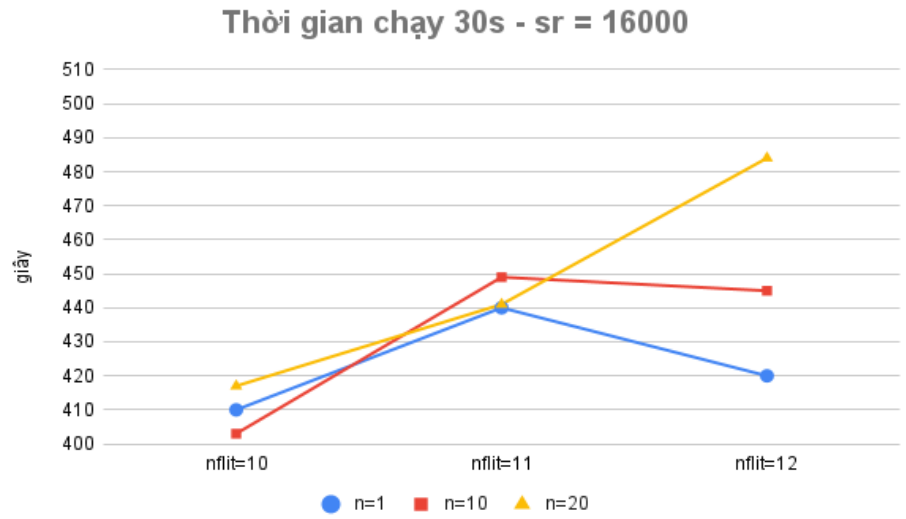
Hình 4.7.Biểu đồ độ chính xác với tập dữ liệu trên 60 giây với sr = 16000

Hình 4.7 là biểu đồ thể hiện độ chính xác với dữ liệu bài hát trên 60 giây và sr = 11025. Kết quả tìm kiếm với top 1,10,20 có độ chính xác cao nhất 64% với nflit = 10 và thấp nhất 0 % với top 1 và nflit = 11,12.



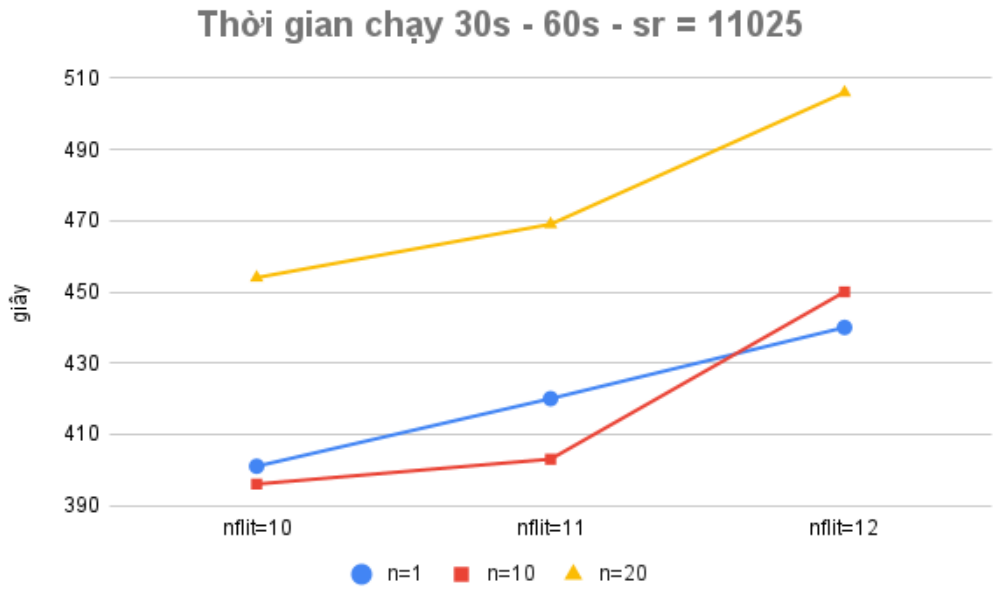
Hình 4.8. Biểu đồ thời gian chạy với tập dữ liệu dưới 30 giây với sr = 11025

Hình 4.8 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát dưới 30 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 402 giây với top 1,20 và nflit = 10. Cao nhất 473 giây với top 20 và nflit = 12.



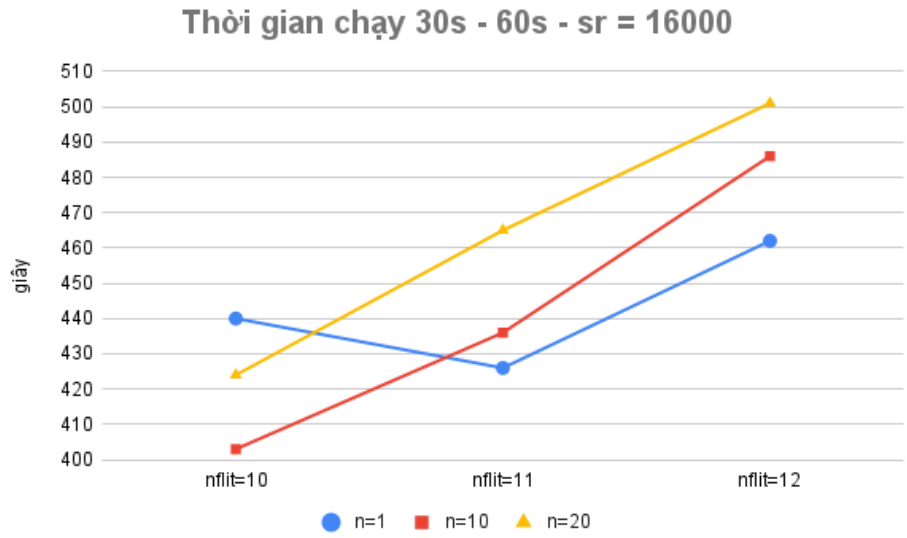
Hình 4.9.Biểu đồ thời gian chạy với tập dữ liệu dưới 30 giây với sr = 16000

Hình 4.9 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát dưới 30 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 403 giây với top 10 và nflit = 10. Cao nhất 484 giây với top 20 và nflit = 12.



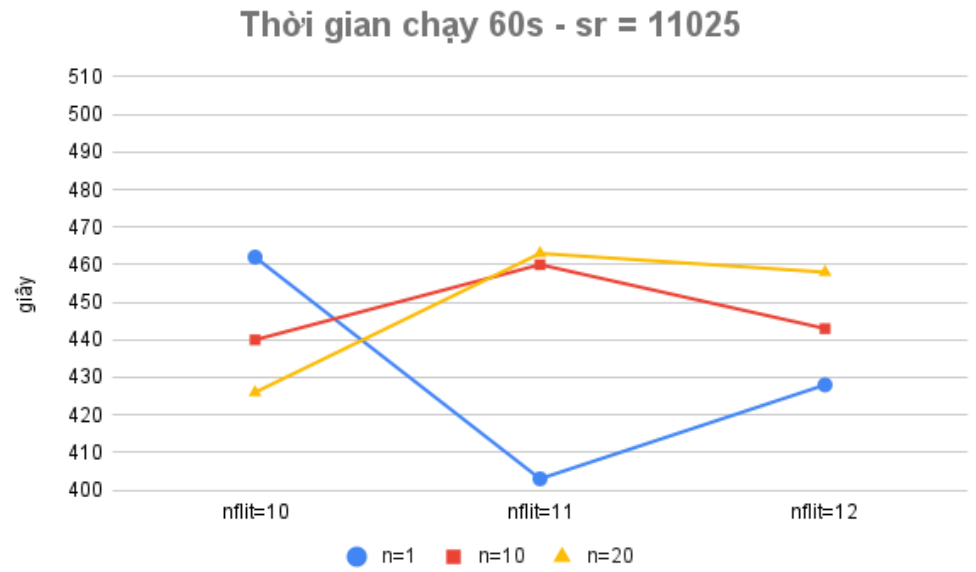
Hình 4.10. Biểu đồ thời gian chạy với tập dữ liệu 30 đến 60s với sr = 11025

Hình 4.10 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát từ 30 đến 60 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 396 giây với top 10 và nflit = 10. Cao nhất 506 giây với top 20 và nflit = 12.



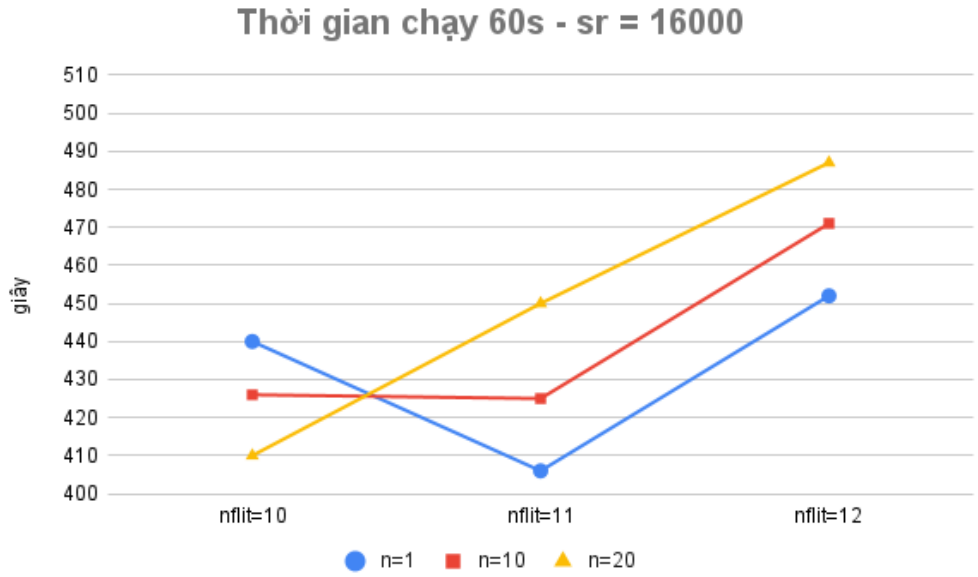
Hình 4.11.Biểu đồ thời gian chạy với tập dữ liệu 30 đến 60s với sr = 16000

Hình 4.11 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát từ 30 đến 60 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 403 giây với top 10 và nflit = 10. Cao nhất 501 giây với top 20 và nflit = 12.



Hình 4.12.Biểu đồ thời gian chạy với tập dữ liệu trên 60 giây với sr = 11025

Hình 4.12 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát trên 60 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 403 giây với top 1 và nflit = 11. Cao nhất 463 giây với top 20 và nflit = 11.



Hình 4.13.Biểu đồ thời gian chạy với tập dữ liệu trên 60 giây với sr = 16000

Hình 4.12 là biểu đồ thể hiện thời gian chạy với dữ liệu bài hát trên 60 giây và sr = 11025. Thời gian tìm kiếm ngắn nhất 406 giây với top 1 và nflit = 11. Cao nhất 487 giây với top 20 và nflit = 12.

Từ kết quả thực nghiệm của 2 tốc độ lấy mẫu sr = 16000 và sr = 11025 ta thấy với tốc độ lấy mẫu là sr = 16000, độ dài MFCC là nfilt = 10 với khoảng thời gian bài hát trên 60 giây và tìm kiếm với top 20 thì đã cho kết quả tìm kiếm chính xác hơn.

4.3. Kết quả nghiên cứu phương pháp Google API

Thực hiện tìm kiếm phân tích đoạn audio bằng Google API với thư viện Speech recognition để hỗ trợ chuyển bài nhạc thành văn bản. Sau khi chuyển được đoạn âm thanh thành văn bản lời bài hát thì chúng ta sẽ tìm kiếm độ tương đồng của văn bản với các lời bài hát đã lưu trong kho 108 lời bài hát bằng Cosine hoặc Spacy Similar. Xuất ra top 20 bài hát giống nhất.

Lời gốc bài hát "Câu hẹn câu thề"

```
Người đi mà chẳng nói một lời
Để anh chơi vơi giữa dòng đời
Lời yêu nay sao quá xa xôi
Lời thương giờ này cũng lặng trôi

Nhận tin em sắp sống bên người
Mà em thương yêu đến cuối đời
Lòng anh như đau thắt em ơi
Làm sao để lệ thôi ngừng rơi.

Hôm nay, em hạnh phúc rồi, em đã bên người, mà lòng em yêu.
Em quên, câu hẹn câu thề, em hứa với người cả đời thương em.

Em ơi! Chữ tình ngắn thôi
Mà sao anh phải kéo theo một đời
Em ơi! Em giờ có đôi
Còn anh thì phải lẻ loi thật rồi

Khi xưa, hứa chẳng cách xa
Gừng cay muối mặn chẳng ham lụa là
Hôm nay, áo hồng gấm hoa
Tình xưa người trả, hết anh thật à?
```

Hình 4.14. Lời gốc bài hát "Câu hẹn câu thề"

Lời bài hát đã được tách từ file audio

```
người đi mà chẳng nói một lời để anh chơi vơi giữa dòng đời
lời yêu sao này quá xa rồi lời thương giờ này cũng làm cho
nhắn tin em sắp sống bên người mà em thương yêu đến cuối đời
lòng anh như đau thắt m làm sao để lễ thôi giữa hôm nay em
hạnh phúc rồi em đã bên người mà lòng em em quên câu hẹn câu
thề em hứa với người cả đời em ơi trữ tình nhân thôi mà sao
anh phải kéo theo một đời em ơi em giờ có đôi còn anh thì
phải lẻ loi thật rồi khi xưa hứa chẳng cách xa gừng cay muối
mặn chẳng hám lụa là hôm nay a hồng gấm hoa tình xưa lời
chào anh thật
```

Hình 4.15. Lời bài hát "Câu hẹn câu thề" đã tách

Độ chính xác với cosine

```
Kết quả:
10  bài hát giống với file audio:D:\LuanVanThucNghiemAPI\CauHenCauThe-
DinhDung-6994741_1.wav
--------------------------------------------------------------------------
--------------------
0.927536231884058D:/Lyric\XesixMasewxNhatnguyen-TuyAm.txt
0.9208077871517533D:/Lyric\RumxNitxMasew-KemDuyen.txt
0.9144204170442609D:/Lyric\PhuongLyxJustaTee-MatTroiCuaEm.txt
0.9131242347147703D:/Lyric\CauHenCauThe.txt
0.9102064996219995D:/Lyric\Hayd-HeadInTheClouds.txt
```

Hình 4.16. Độ chính xác với bài hát "Câu hẹn câu thề" đã tách

Lời gốc bài hát "Ai là người thương em"

```
Người con gái anh từng yêu sao rồi?
Có một mình đi dưới mưa lúc buồn?
Lệ còn rơi khi ngồi xem thước phim buồn?
Ôm thật chặt vào ai khóc như đứa trẻ?

Người con gái anh từng yêu quên rồi
Có những chiều tay nắm tay ngóng đợi
Hoàng hôn xuống ta kề vai nói những lời
Rằng đôi ta sẽ chỉ cần nhau thôi
Hà ha ha ha há ha hà ha
Cô gái anh yêu hay quan tâm anh và nhắc anh bao điều
Em thích hoa hồng và mùa đông được anh ôm phía sau lưng
Em nói bên anh qua bao nơi em cảm thấy rất nhẹ nhàng
Vậy giờ ai là người cho em yên bình?
```

Hình 4.17. Lời gốc bài hát "Ai là người thương em"

Lời bài hát đã được tách

```
người con gái anh từng yêu sao rồi còn một mình
đi dưới mưa lúc bọn lê con dơi khi ngồi xem thước
phim buồn ôm thật chặt vào ai khóc như người con
gái anh từng yêu quên rồi có những chiều tay nắm
tay đời hoàng hôn xuống ta kề vai nỗi những lời
rằng đôi ta sẽ chỉ cần nhau thôi
```

Hình 4.18. Lời bài hát "Ai là người thương em" đã được tách

Độ chính xác với Cosine

```
Kết quả:
20  bài hát giống với file audio:AiLaNguoiThuongEm_2
--------------------------------------------------------------------------------
0.8904662452369615D:/LuanVanThucNghiemAPI/Lyric\SonTung-ChungTaKhongThuocVeNhau.txt
0.8613214599314613D:/LuanVanThucNghiemAPI/Lyric\OnlyC-YeuLaThaThu.txt
0.8531944006707276D:/LuanVanThucNghiemAPI/Lyric\MasewxKhoiVu-NhatThan.txt
0.8291877369127311D:/LuanVanThucNghiemAPI/Lyric\ChiPu-ChiTaBietThoi.txt
0.821401008386543D:/LuanVanThucNghiemAPI/Lyric\QuanAP-AiLaNguoiThuongEm.txt
0.7920037969362131D:/LuanVanThucNghiemAPI/Lyric\LouHoang-ThichHayLaYeu.txt
0.7902751763935048D:/LuanVanThucNghiemAPI/Lyric\OnlyCxLouHoang-DemNgayXaEm.txt
```

Hình 4.19. Độ chính xác lời bài hát "Ai là người thương em" đã tách

Lời gốc của bài hát "Bao giờ lấy chồng"

```
Năm mới lại đến em vẫn lẻ bóng một mình
Mà đâu làm sao vì em đã có gia đình
Mẹ cha chờ em về quê ăn Tết linh đình
Tay xách hành lý, miệng em thì vẫn tươi xinh

Đồng quê ngày nay cũng không gì khác năm xưa
Vẫn đấy chợ phiên người ta chen chúc đẩy đưa
Về đến nhà vui, làm sao kể hết cho vừa
Mẹ em ra đón hỏi ngay chịu lấy chồng chưa?

Nụ cười chợt thoáng vụt tắt trên má hồng
Em nào đâu muốn lấy chồng
Em chỉ muốn ở mãi bên mẹ cha
Mỗi năm tết đến, em mới về quê nhà
Nấu bánh mứt tặng ông bà
Giao thừa quây quần bên cả nhà ta

Đừng ai hỏi em chuyện lấy chồng
Đừng ai hỏi em chuyện lấy chồng
Mùa xuân này em chưa lấy chồng
Em vẫn chưa muốn lấy chồng
```

Hình 4.20. Lời gốc của bài hát "Bao giờ lấy chồng"

Lời bài hát đã được tách:

```
năm mới lại đến em vẫn lẻ bóng một mình mà đau
làm sao vì em đã có gia đình mẹ cha chờ em về quê
em thích lĩnh bình thạnh mỹ bên em thì vẫn quen
nhìn ai cũng khác năm xưa để đi đến nhà vui làm
sao trong sữa mẹ ra đón nhỏ đi lấy chồng chưa
cười chẳng mãi nếu biết rằng chúng ta cùng đến
với em rồi
```

Hình 4.21. Lời bài hát "Bao giờ lấy chồng" đã tách

Độ chính xác với Cosine:

```
Kết quả:
20  bài hát giống với file audio:BaoGioLayChong_1
--------------------------------------------------------------------------------
0.9275362318840580:/LuanVanThucNghiemAPI/Lyric\XesixMasewxNhatnguyen-TuyAm.txt
0.9208077871517533:/LuanVanThucNghiemAPI/Lyric\RumxNitxMasew-KemDuyen.txt
0.9144204170442609:/LuanVanThucNghiemAPI/Lyric\PhuongLyxJustaTee-MatTroiCuaEm.txt
0.9131242347147703:/LuanVanThucNghiemAPI/Lyric\CauHenCauThe.txt
0.9102064996219995:/LuanVanThucNghiemAPI/Lyric\Hayd-HeadInTheClouds.txt
0.9060505152759883:/LuanVanThucNghiemAPI/Lyric\MazieStevexAokiRegard-NewYork.txt
0.9056935758644526:/LuanVanThucNghiemAPI/Lyric\JustaTeexPhuongLy-ThangDien.txt
0.9009131617196757:/LuanVanThucNghiemAPI/Lyric\HoangThuyLinhxDen-GieoQue.txt
0.8977703938568498:/LuanVanThucNghiemAPI/Lyric\MinxDenVauxJustaTee-ViYeuCuDamDau.txt
0.8946481190484873:/LuanVanThucNghiemAPI/Lyric\DuongMotChieu-HuynhTuMagazine-4855398.txt
0.8921094177568789:/LuanVanThucNghiemAPI/Lyric\ViMeAnhBatChiaTay-MiuLe-7503053.txt
0.8913342680270697:/LuanVanThucNghiemAPI/Lyric\MasewxKhoiVu-NhatThan.txt
0.8888932021979545:/LuanVanThucNghiemAPI/Lyric\MasewxPhao-DieuToa.txt
0.8887379098833177:/LuanVanThucNghiemAPI/Lyric\SonTung-ChacAiDoSeVe.txt
0.8862951684868084:/LuanVanThucNghiemAPI/Lyric\MiuLe-MinhYeuTuBaoGio.txt
0.8832352082415980:/LuanVanThucNghiemAPI/Lyric\ChiPu-TalkToMeCoNenDungLai.txt
0.8827119022206268:/LuanVanThucNghiemAPI/Lyric\HoaproxxXesi-VoTinh.txt
0.8767942266815111:/LuanVanThucNghiemAPI/Lyric\DamVinhHung-Hello.txt
0.8764222395791800:/LuanVanThucNghiemAPI/Lyric\MasewxTuanCry-MoiTrau.txt
0.8645353285886334:/LuanVanThucNghiemAPI/Lyric\BichPhuong-BaoGioLayChong.txt
```

Hình 4.22. Độ chính xác với lời bài hát "Bao giờ lấy chồng" đã tách

Bảng 4.10. Kết quả thực nghiệm của phương pháp tách nhạc bằng speech_recognition Google API

Số bài hát thực nghiệm	Số bài hát đúng	Độ chính xác (%)
10	4	40

Kết quả việc tách lời bài hát bằng thư viện speech_recognition chuyển đổi được 46 bài hát trên 100 bài. Số bài còn lại không chuyển đổi thành văn bản được. Từ đó ta thấy việc sử dụng thư viện speech_recognition để tách lời còn có nhiều hạn chế:

- Chuyển đổi từ file nhạc .mp3 sang file .wav mất nhiều thời gian

- Thư viện không chuyển đổi được các bài hát có tốc độ nhanh hay các bài hát quá lớn. Những bài nhạc tiếng Việt có thể chuyển đổi sang nhưng khi kết hợp với những tiếng khác như tiếng Anh thì việc tách lời không đúng.
- Việc tách thiếu lời hoặc chuyển đổi sai dẫn tới kết quả so sánh với thư viện lời bài hát sẽ không chính xác

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Luận văn trên đây đã giới thiệu một số phương pháp tìm kiếm nhạc có lời, việc tìm kiếm nhạc có lời theo nội dung nói chung và nhận dạng giọng nói nói riêng là một vấn đề khó, đòi hỏi kết hợp nhiều phương pháp khác nhau, sử dụng nhiều bộ tham số đặc trưng khác nhau. Trong khuôn khổ luận văn mới chỉ cài đặt thử nghiệm nhận dạng giọng nói sử dụng một đặc trưng MFCC, chưa kết hợp thêm các đặc trưng khác như tần số cơ bản. Sử dụng thư viện Speech Recognition để tách lời bài hát và thử nghiệm độ chính xác với thư viện Spacy hoặc Cosine similar .

Luận văn đã thực hiện được:

- Tìm hiểu các đặc trưng âm thanh và tiếng nói.
- Tìm hiểu phương pháp trích chọn đặc trưng sử dụng MFCC
- Tìm hiểu phương pháp nhận dạng giọng nói Google API
- Học hỏi được ngôn ngữ lập trình Python và các thư viện hỗ trợ
- Biết về thuật toán tìm kiếm lân cận
- Tăng khả năng fix lỗi và tư duy logic

Tuy nhiên, đề tài được xây dựng chưa hoàn chỉnh vẫn còn một số hạn chế:

- Bộ dữ liệu thử nghiệm quá nhỏ. Thư viện 200 bài hát và 100 lời nhạc.
- Phương pháp đã cho kết quả khá tốt nhưng chưa thực sự chính xác có một số kiến thức chưa được học dẫn đến việc xây dựng và triển khai phương pháp gặp khó khăn.
- Chưa tăng được độ chính xác của phương pháp nhận dạng bằng Google API và thư viện Speech recognition.
- Chưa xây dựng website để người dùng tương tác.

2. Hướng phát triển

- Tìm kiếm và thu thập thêm dữ liệu từ nhiều nguồn khác nhau.
- Chạy thực nghiệm thêm với nhiều bộ tham số khác nhau để có những kết quả chính xác hơn.

- Tích hợp thêm việc xây dựng website và tối ưu hóa để người dùng thuận tiện sử dụng và có thể nhìn thấy kết quả trên màn hình.
- Tích hợp thêm chức năng cho người dùng sử dụng hát trực tiếp vào để tìm nhạc bằng micro.
- Tìm hiểu thêm và xây dựng tìm kiếm nhạc theo hướng máy học.

TÀI LIỆU THAM KHẢO

[1] Shazam là gì? [Internet]. [cited 2022 Mar 23] Available from: https://www.thegioididong.com/game-app/shazam-la-gi-cach-tim-bai-hat-bang-shazam-de-dang-nhanh-chong-1413369

[2]Xử lý dữ liệu âm thanh [Internet]. [cited 2022 Jul 5] Available from: https://viblo.asia/p/xu-ly-du-lieu-am-thanh-Qpmlezg95rd

[3]Client-Server [Internet]. [cited] Available from: https://fptcloud.com/client-server/

[4]Âm thanh, các đặc trưng của âm thanh [Internet]. [cited 2023 Mar 28] Available from: https://vi.wikipedia.org/wiki/%C3%82m_thanh

[5]Tần số âm thanh là gì [Internet]. [cited 2022 Jun 12] Available from: https://vi.wikipedia.org/wiki/T%E1%BA%A7n_s%E1%BB%91_%C3%A2m_thanh

[6]Số hóa âm thanh và lợi ích của số hóa âm thanh [Internet]. Available from: https://sohoatailieu.com/so-hoa-am-thanh/

[7]Tần số lấy mẫu, kích thước lấy mẫu [Internet]. [cited 2021 Sep 20] Available from: https://viblo.asia/p/tim-hieu-tan-so-lay-mau-kich-thuoc-mau-va-bit-rate-trong-xu-ly-am-thanh-XL6lA6pr5ek

[8]Approximate Nearest neighbor search ANN [Internet]. [cited 2023] Available from: https://ignite.apache.org/docs/latest/machine-learning/binary-classification/ann

[9]Nearest neighbor search NNS [Internet]. [cited 2023 Feb 18] Available from: https://en.wikipedia.org/wiki/Nearest_neighbor_search

[10]Mel Frequency Cepstral Coefficients [Internet].[cited 2019 Aug 29] Available from:https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9

[11] Định lý lấy mẫu Nyquist–Shannon [Internet]. [cited 2022 Oct 31] Available from: https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_l%C3%BD_l%E1%BA%A5y_m%E1%BA%ABu_Nyquist%E2%80%93Shannon

[12]Difference Speech Recognition and Voice Recognition [Internet]. Available from: https://www.kardome.com/blog-posts/difference-speech-and-voice-recognition

[13] 5 trang web nhận diện bài hát [Internet]. [cited 2023 Feb 27] Available from:

https://www.thegioididong.com/game-app/5-trang-web-nhan-dien-bai-hat-qua-giai-dieu-1279538
[14] Anaconda [Internet]. [cited 2022 Oct 20] Available from: https://vn.got-it.ai/blog/anaconda-la-gi-tim-hieu-nen-tang-khoa-hoc-du-lieu-pho-bien-nhat
[15]Phú T.Q Xây dựng hệ thống tìm kiếm âm thanh theo nội dung dựa trên đặc trưng miền tần số, Đại học dân lập Hải Phòng [Internet]. [cited 2016 Available from: https://lib.hpu.edu.vn/bitstream/handle/123456789/26586/6_PhuThiQuyen_CHCNTTK1.pdf?sequence=1
[16] Annoy [Internet]. [2023 Apr 10] Available from: https://pypi.org/project/annoy/
[17] MFCC [Internet]. [cited 2020 Apr 29] Available from: https://viblo.asia/p/feature-extraction-mfcc-cho-xu-ly-tieng-noi-4dbZN2xmZYM
[18] Jupyter Notebook là gì [Internet]. [cited 2022 Jul 29] Available from: https://200lab.io/blog/jupyter-notebook-la-gi/
[19] Python là gì tại sao nên chọn python [Internet]. [cited 2022 Sep 7] Available from: https://quantrimang.com/hoc/python-la-gi-tai-sao-nen-chon-python-140518
[20] Visual Studio Code là gì [Internet]. [cited 2023 Mar 15] Available from: https://cellphones.com.vn/sforum/visual-studio-code
[21] Tích vô hướng và công thức tính tích vô hướng [Internet]. Available from: https://vi.wikipedia.org/wiki/T%C3%ADch_v%C3%B4_h%C6%B0%E1%BB%9Bng
[22] Độ tương tự Cosine Cosine similarity [Internet]. Available from: https://vi.wikipedia.org/wiki/%C4%90%E1%BB%99_t%C6%B0%C6%A1ng_t%E1%BB%B1_cosin
[23] Tai người nghe được bao nhiêu Hz[Internet]. [cited 2022 Oct 31] Available from: https://nhathuoclongchau.com.vn/bai-viet/giai-dap-tai-nguoi-nghe-duoc-bao-nhieu-hz-la-tot-nhat-65077.html
[24] 9 cách tìm tên bài hát chỉ qua giai điệu nhanh chóng, đơn giản [Internet]. 2021 https://www.dienmayxanh.com/kinh-nghiem-hay/xx-cach-tim-ten-bai-hat-chi-qua-giai-dieu-nhanh-ch-1260043
[25] SciPy Logo [Internet]. Available from: https://scipy.org/
[26] Librosa Logo [Internet]. Available from: https://librosa.org/
[27] Torchaudio [Internet]. Available from: https://pytorch.org/audio/stable/index.html