TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN TRIỀU HUY - 52000449
TRẦN THỊ NGỌC ÁNH - 52000008

HỆ THỐNG NHẬN DIỆN TIẾNG NÓI DÂN TỘC M'NÔNG

DỰ ÁN CÔNG NGHỆ THÔNG TIN

KỸ THUẬT PHẦN MỀM

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN TRIỀU HUY - 52000449
TRẦN THỊ NGỌC ÁNH - 52000008

HỆ THỐNG NHẬN DIỆN TIẾNG NÓI DÂN TỘC M'NÔNG

DỰ ÁN CÔNG NGHỆ THÔNG TIN

KỸ THUẬT PHẦN MỀM

Người hướng dẫn
PGS. TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng tôi muốn bày tỏ lòng biết ơn sâu sắc đến người hướng dẫn trực tiếp của chúng tôi, Thầy Lê Anh Cường, vì sự giám sát, hướng dẫn và hỗ trợ của Thầy trong môn học Dự án Công nghệ Thông tin. Những kiến thức tiếp thu được từ Thầy không chỉ là tiền đề cho báo cáo này mà còn là hành trang quý báu cho chúng tôi trong tương lai. Tuy nhiên, kiến thức là vô hạn mà thời gian chỉ có hạn, nhóm không thể tránh khỏi những thiếu sót. Rất mong nhận được ý kiến đóng góp từ Thầy để chúng em hoàn thiện hơn kiến thức cho bản thân.

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 01 tháng 01 năm 2024.

Tác giả

(Ký tên và ghi rõ họ tên)

Trần Triều Huy

Trần Thị Ngọc Ánh

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS. TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 01 tháng 01 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Trần Triều Huy

Trần Thị Ngọc Ánh

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

Error! No table of figures entries found.

DANH MỤC CÁC CHỮ VIẾT TẮT

ASR Automatic Speech Reconigtion

ML Machine Learning

WER Word Error Rate

CHƯƠNG 1. GIỚI THIỆU

1.1 Bài toán

Đề tài của nhóm chúng tôi mang tên "Hệ thống nhận diện tiếng nói dân tộc M'Nông". Để thực hiện được việc nhận diện tiếng nói dân tộc, trước tiên phải xây dựng một hệ thống nhận dạng tiếng nói tự động (Automatic Speech Recognition - ASR), đây là hệ thống có khả năng chuyển đổi chuỗi âm thanh thành chuỗi từ ngữ. Việc xây dựng một hệ thống nhận dạng tiếng nói đòi hỏi nhóm phát triển phải am hiểu các kỹ thuật, lý thuyết từ nhiều kiến thức khác nhau như: học máy, trí tuệ nhân tạo, lý thuyết xác suất thống kê, âm học - vật lý,… Trên thế giới đã có nhiều nhóm nghiên cứu phát triển thành công hệ nhận dạng tiếng nói cho các ngôn ngữ lớn như: tiếng Anh, tiếng Trung Quốc, tiếng Nhật,… nhưng giải pháp nhận diện cho tiếng Việt đặc biệt là tiếng nói dân tộc thiểu số tại Việt Nam vẫn còn rất nhiều hạn chế.

1.2 Các nghiên cứu liên quan

Giao tiếp người - máy là một lĩnh vực nghiên cứu lớn và khó khăn nhưng lại có nhiều ứng dụng thực tiễn. Tiếng nói là một phương thức giao tiếp tự nhiên nhất của con người và vì vậy, nghiên cứu để máy tính có thể nhận dạng tiếng nói tự động (ASR) đã trải qua quá trình phát triển hơn 70 năm. Những cố gắng nghiên cứu đầu tiên về ASR đã được thực hiện trong thập niên 50 với ý tưởng chính là dựa trên ngữ âm. Do kỹ thuật xử lý tín hiệu số cũng như khả năng của máy tính còn giới hạn nên các hệ thống nhận dạng khi đó chỉ tập trung khai thác đặc trưng phổ cộng hưởng đối với các nguyên âm của tín hiệu sau khi đi qua các bộ lọc. Trong giai đoạn này có một số hệ thống nổi bật như: hệ thống nhận dạng ký số rời rạc của Bell-lab (1952), bộ nhận dạng 13 âm vị của Đại học College - Anh (1958),…

Trong tập kỷ 60 của thế kỷ trước, một ý tưởng đáng chú ý của tác giả người Nga - Vintsyuk - khi ông đề xuất phương pháp nhận dạng tiếng nói dựa trên quy hoạch động theo thời gian (Dynamic Time Warping - DTW). Đáng tiếc, phải đến những năm 80 thì phương pháp này mới được thế giới biết đến. Cuối những năm

1960, Reddy (Đại học CMU - Mỹ) đã đề xuất ý tưởng đầu tiên về nhận dạng tiếng nói liên tục bằng kỹ thuật đánh dấu đường đi và truy vết lùi tìm kết quả.

Đến những năm 70, nghiên cứu về nhận dạng tiếng nói đã bước đầu thu được những kết quả mong đợi, làm nền tảng cho những nghiên cứu sau này. Trước hết là bài toán nhận dạng từ rời rạc được hiện thực hóa bởi ý tưởng của các nhà khoa học Nga và Nhật. Velichko và Zagoruyko là hai nhà nghiên cứu tiên phong trong việc áp dụng ý tưởng phân lớp mẫu cho ASR. Sakoe và Chiba đề xuất các kỹ thuật sử dụng phương pháp quy hoạch động. Và Itakura trong khi ở Bell-lab đã đưa ra phương pháp mã hóa dự báo tuyến tính (Linear Predictive Coding - LPC) làm tiền đề cho việc áp dụng các tham số LPC vào ASR. Các hệ thống ASR đáng chú ý của giai đoạn này gồm: Harpy và Hearsay-II của Đại học CMU (Mỹ), hệ thống BWIM (BBN),…

Nỗ lực về ASR trong thập kỷ 80 đánh dấu sự dịch chuyển trong phương pháp luận: từ cách tiếp cận đối sách mẫu sang cách tiếp cận sử dụng mô hình thống kê. Ngày nay, hầu hết các hệ thống ASR đều dựa trên mô hình thống kê được phát triển ở thập kỷ này cùng với những cải tiến ở thập kỷ 90. Một trong những phát minh quan trọng nhất giai đoạn này là mô hình Markov ẩn (Hidden Markov Model - HMM). Dù HMM được áp dụng thành công ở một số phòng lab nhưng phải đợi đến vài năm sau đó, mô hình này mới trở nên phổ biến.

Thập niên 90 tiếp tục ghi nhận một số kết quả nghiên cứu mới trong lĩnh vực phân lớp mẫu. Cụ thể hơn, bài toán phân lớp theo mô hình thống kê (dựa trên luật quyết định Bayes), đòi hỏi phương pháp ước lượng các phân bố cho dữ liệu, được chuyển thành bài toán tối ưu, bao gồm phép cực tiểu lỗi phân lớp bằng thực nghiệm. Cuối cùng, các ứng dụng được phát triển trong giai đoạn này gồm: hệ thống trả lời thông tin tự động cho các chuyến bay (Air Travel Information Service - ATIS), hệ thống ghi lại các bản tin phát thanh (Broadcast News Transcription System),…

Đến những năm đầu thế kỷ 21, các nghiên cứu tập trung vào việc nâng cao kết quả nhận diện tiếng nói thông qua một chương trình có tên gọi EARS (Efective

Affordable Reusable Speech-to-Text). Tại thời điểm này, tiếng nói đã không còn bị ràng buộc các điều kiện và có thể thu âm trong môi trường bình thường.

Tại Việt Nam có 2 nhóm nghiên cứu chính về nhận diện tiếng nói liên tục với bộ từ vựng lớn. Nhóm đầu tiên thuộc Viện Công nghệ Thông tin do PGS. Lương Chi Mai đứng đầu, với phương pháp ANN và công cụ CSLU được sử dụng. Nhóm thứ hai thuộc Đại học Khoa học Tự nhiên Tp. HCM do PGS. Vũ Hải Quân đứng đầu với phương pháp HMM và công cụ HTK được sử dụng. Các nghiên cứu tập trung giải quyết bài toán truy vấn thông tin tiếng Việt, nhận dạng tiếng nói, hệ thống giao tiếp người - máy,… Ngoài ra, còn có nghiên cứu của LIG (Laboratoire Informatique de Grenoble) hợp tác với phòng thí nghiệm MICA (Hà Nội) về sự khả chuyển của các mô hình ngữ âm.

1.3 Mục tiêu nghiên cứu

Có thể thấy, lịch sử phát triển của các hệ thống nhận diện tiếng nói từ trong đến ngoài nước là vô cùng phong phú và lâu đời. Tuy nhiên, Việt Nam là một quốc gia với 54 dân tộc anh em cùng sinh sống, bên cạnh tiếng Kinh phổ biến nhất trong cộng đồng người Việt Nam, chúng ta còn rất nhiều tiếng nói khác.

Dân tộc M’Nông là một dân tộc thiểu số ở Việt Nam, với dân số khoảng 127.334 người (theo Tổng điều tra dân số và nhà ở năm 2019), cư trú tại 51 trên tổng số 63 tỉnh, thành phố. Tiếng M’Nông là một ngôn ngữ thuộc ngữ tộc Môn-Khmer của ngữ hệ Nam Á. Tiếng M’Nông được nghiên cứu đầu bởi nhà ngôn ngữ học Richard Phillips vào đầu những năm 1970 và đây là một ngôn ngữ có một hệ thống ngữ âm phức tạp, với nhiều thanh điệu và phụ âm.

Hệ thống nhận diện tiếng nói dân tộc M’Nông là một hệ thống có khả năng nhận diện tiếng nói của người M’Nông thông qua ứng dụng công nghệ thông tin.

Mục tiêu khi chúng tôi thực hiện đề tài là xây dựng một hệ thống nhận diện tiếng nói dân tộc M’Nông có độ chính xác cao, có thể áp dụng vào thực tế và đáp ứng được nhiều yêu cầu của nhiều mục đích sử dụng khác nhau.

Mục tiêu chi tiết:

- Tìm hiểu các khái niệm có liên quan đến hệ thống nhận dạng tiếng nói, làm rõ hơn các yếu tố quan trọng trong việc sử dụng công cụ hỗ trợ.

- Tìm hiểu phương pháp cài đặt công cụ hỗ trợ xây dựng hệ thống nhận diện tiếng nói dân tộc.

- Tìm hiểu, xây dựng mô hình nhận dạng thích hợp cho tiếng M'Nông.

- Xây dựng mô hình, thực nghiệm, từ đó rút ra kết luận và nhận xét.

1.4 Phương pháp nghiên cứu

Nghiên cứu về hệ thống nhận diện tiếng nói dân tộc M'Nông đòi hỏi cần một mô hình kỹ lưỡng và toàn diện. Chính vì thế, cần nghiên cứu và phân tích các tài liệu về lý thuyết nhận diện tiếng nói, đặc điểm ngữ âm của tiếng M'Nông, cũng như các nghiên cứu trước đây về hệ thống nhận diện tiếng nói dân tộc thiểu số. Mục đích nhằm nắm rõ các nguyên lý, phương pháp và thực tiễn của nhận diện tiếng nói, cũng như các đặc điểm của tiếng M'Nông.

Đối tượng được chúng tôi chọn để nghiên cứu là tiếng M'Nông của những người nói giọng huyện Lắk thuộc tỉnh Đắk Lắk. Tất cả các nội dung nghiên cứu trong dự án chỉ giới hạn trong phạm vi điều kiện người nói trong trạng thái bình thường, các vấn đề người nói cố tình cải trang hay giả giọng nói đều nằm ngoài phạm vi nghiên cứu của dự án.

Chúng tôi sử dụng bộ dữ liệu bao gồm các mẫu âm thanh tiếng M'Nông do nhóm thu thập. Bao gồm các câu, từ thường dùng trong đời sống hằng ngày và được thu âm bởi chính người dân địa phương nói tiếng M'Nông. Dữ liệu âm thanh được thu từ hai người nói và với các giọng nói khác nhau (nam, nữ) để hệ thống có thể nhận diện được nhiều giọng nói.

Dữ liệu thu thập cần được xử lý để chuẩn hóa trước khi tiến hành huấn luyện. Trong quá trình xử lý dữ liệu, chúng tôi phân đoạn âm thanh thành các đoạn âm thanh nhỏ hơn, mỗi đoạn tương ứng với một câu hoặc một từ.

Mô hình nhận diện tiếng nói là một thuật toán học máy được sử dụng để phân loại các mẫu âm thanh. Trong đề tài này, chúng tôi sử dụng mô hình Whisper

của OpenAI vì đây là một hệ thống nhận dạng giọng nói tự động mã nguồn mở, thiết kế dễ sử dụng và cho độ chính xác cao.

Trong quá trình huấn luyện mô hình, chúng tôi cung cấp cho mô hình các mẫu âm thanh và nhãn của các mẫu âm thanh đó. Mô hình sẽ học cách phân loại các mẫu âm thanh dựa trên các nhãn đã cung cấp và cho ra sản phẩm là văn bản.

Sau khi huấn luyện, mô hình sẽ được thử nghiệm trên dữ liệu mới để đánh giá độ chính xác và có các điều chỉnh phù hợp.

1.5 Nội dung nghiên cứu

Chương 1: Trình bày tổng quan: giới thiệu về đề tài, bài toán được đề cập, các nghiên cứu có liên quan đến đề tài nhận diện tiếng nói trong và ngoài nước, các mục tiêu nghiên cứu đề ra cũng như phương pháp để tiến hành nghiên cứu.

Chương 2: Trình bày kiến thức nền tảng bao gồm: lý thuyết cơ bản về âm học, ngữ âm học, đặc điểm trong tiếng M’Nông, các kiến thức cơ bản để xây dựng và sử dụng một hệ thống nhận dạng tiếng nói. Cơ sở lý thuyết về mô hình Whisper, bao gồm các thuật toán liên quan, ý nghĩa của Whisper trong một hệ thống nhận diện tiếng nói.

Chương 3: Trình bày phương pháp đề xuất: xây dựng kiến trúc, giải pháp của hệ thống.

Chương 4: Trình bày chi tiết thực nghiệm, thống kê về dữ liệu, các mô hình thực nghiệm là gì, xây dựng bộ huấn luyện, tiến hành huấn luyện, giải thích kết quả huấn luyện.

Chương 5: Nêu lên kết luận, trình bày những kết quả đạt được, những điểm còn hạn chế, cũng như kinh nghiệm sau quá trình thực hiện dự án, từ đó nêu lên các hướng cải thiện và phát triển nghiên cứu.

CHƯƠNG 2. KIẾN THỨC NỀN TẢNG

2.1 Tổng quan về âm học và tiếng nói

2.1.1 Âm học

❖ Khái niệm

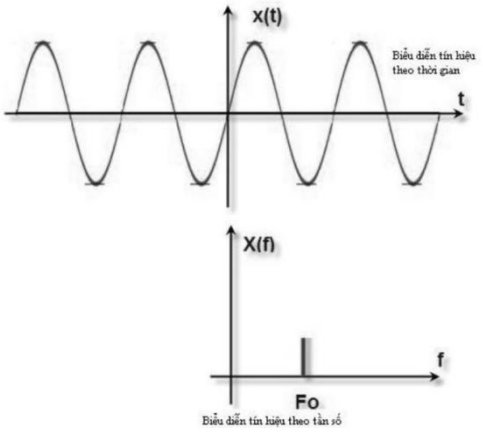
Khi có nguồn phát ra âm thanh (tiếng nói, tiếng nhạc cụ, tiếng động vật kêu,…) ta sẽ nghe và cảm nhận được âm thanh phát ra. Vật tạo ra âm thanh được gọi là nguồn phát âm. Âm thanh chính là sự dao động cơ của các thành phần vật chất trong môi trường lan truyền và khi nó đến tai ta, ta cảm nhận được âm thanh. Trong môi trường không có vật chất tồn tại như chân không sẽ không có âm thanh bởi không có dao động cơ. Trong đời sống xã hội, âm thanh là phương tiện giao tiếp, tuyền đạt thông tin xuất hiện lâu đời nhất trong lịch sử nhân loại. Khi nghiên cứu về âm thanh, người ta thường quan tâm đến 2 đặc điểm: đặc trưng vật lý và đặc trưng sinh học.

❖ Biểu diễn tín hiệu âm thanh trong miền thời gian và tần số

Để biểu diễn âm thanh trong miền thời gian, ta có thể dùng hàm toán học x(t). Trong đó:

- t: thời gian
- x: biên độ biến thiên (ly độ)

Biểu diễn x(t) bằng đồ thị thời gian. Đặt x(t) = A.sin ωot = A.sin 2πFot:



Hình 1. Biểu diễn tín hiệu âm thanh

Phổ tính hiệu: là cách biểu diễn các thành phần cấu tạo nên x(t) theo tần số. Với tín hiệu sin nói trên, đồ thị phổ là một vạch có cao độ A tại điểm có tần số Fo. Ta nói đó là phổ vạch. Trong thực tế, với x(t) bất kỳ, biến thiên, không tuần hoàn, người ta sẽ dùng phân tích Fourier để tính toán phổ tín hiệu. Khi đó ta có phổ liên tục X(Ω).

❖ Các loại âm thanh

Những dao động cơ mà con người nghe được gọi là âm thanh (sound).

Âm thanh có thể biểu diễn theo thời gian và tần số do có thể phân tích một tín hiệu âm thanh thành tổ hợp các thành phần tần số khác nhau (chuỗi Fourier, tích phân Fourier). Đơn giản hơn, một âm thanh có thể là tổ hợp từ nhiều đơn âm có tần số dao động nhất định.

Dải tần số nghe được là từ 20 Hz - 20000 Hz. Siêu âm là âm dao động ngoài 20000 Hz. Hạ âm là âm dao động dưới 20 Hz. Tai người không nghe được siêu âm và hạ âm.

Tiếng nói (voice, speech) là âm thanh phát ra từ miệng con người, được truyền đi trong không khí đến tai người nghe. Dải tần số đủ nghe rõ là từ 300 Hz đến 3500 Hz (dải tần tiêu chuẩn). Dải tần số có chất lượng cao có thể từ 200 Hz đến 7000 Hz (ampli hội trường).

Âm nhạc (music) là âm thanh phát ra từ các nhạc cụ. Dải tần số từ 20 Hz đến 15000 Hz.

Tiếng kêu là âm thanh phát ra từ miệng động vật.

Tiếng động là âm thanh phát ra từ va chạm giữa các đồ vật.

Tiếng ồn (noise) là âm thanh không mong muốn.

Nhìn chung, có 2 loại âm: tuần hoàn (tiếng nói, âm nhạc) và không tuần hoàn (tín hiệu tạp nhiễu).

❖ Đơn vị đo âm thanh

Cảm nhận độ to của âm thanh dựa vào hàm số mũ:

Bel = 10 lg P2/P1

Deccibel = 20 lg I2/I1

2.1.2 Tiếng nói

Là âm thanh phát ra từ miệng người. Nghiên cứu tiếng nói gồm: Bộ máy phát âm của con người. Thụ cảm âm thanh của tai người. Phân loại tiếng nói.

Bộ máy phát âm của con người gồm:

- Phổi đóng vai trò bơm không khí, tạo năng lượng hình thành âm.

- Đôi dây thanh (vocal fold) là hai cơ thịt trong cuống họng, có hai đầu tính vào nhau, còn hai đầu dao động với tần số cơ bản là Fo. Fo của nam giới nằm trong khoảng 100 - 200 Hz, của nữ giới là 300 - 400 Hz, của trẻ em là 500 - 600 Hz.

- Thanh quản và vòm miệng đóng vai trò như hốc và cộng hưởng, tạo ra phân biệt tần số khi tín hiệu dao động từ đôi dây thanh phát ra.

- Miệng đóng vai trò phát tán âm thanh ra bên ngoài.

- Lưỡi thay đổi để tạo ra tần số formant khác nhau.

- Các âm khác nhau là do vị trí tương đối của formants.

Phân loại tiếng nói theo thanh:

- Âm hữu thanh (voiced sounds): là âm khi phát ra có sự dao động của đôi dây thanh, nên nó tuần hoàn với Fo. Vì vậy phổ của nguyên âm là phổ vạch, khoảng cách giữa các vạch bằng chính Fo.

- Âm vô thanh (unvoiced sounds): phát ra khi đôi dây thanh không dao động. Phổ tín hiệu có dạng nhiễu trắng và phổ phân bố đều.

Phân loại tiếng nói:

- Nguyên âm (vowel): là âm phát ra có thể kéo dài. Nguyên âm là âm hữu thanh, tuần hoàn và khá ổn định trong một đoạn thời gian vài chục ms.

- Phụ âm (consonant): là âm chỉ phát ra một nhát, không thể kéo dài. Có phụ âm hữu thanh và phụ âm vô thanh.

Thanh điệu của tiếng Việt tương ứng với các dấu: ngang, huyền, sắc, hỏi, ngã, nặng khi viết. Phân tích cho thấy thanh điệu là sự thay đổi Fo, tần số cơ bản pitch, trong quá trình phát âm các nguyên âm và tai người cảm nhận được. Tiếng Việt có 6 thanh thể hiện sự phong phú và độc đáo, trong khi tiếng Trung Quốc chỉ có 4 thanh.

Giọng bổng (high voiced pitch) hay giọng trầm (low voiced pitch) là Fo cao hay thấp. Như vậy Fo đóng vai trò rất quan trọng trong cảm nhận, thụ cảm âm thanh của con người.

Tiếng bổng hay tiếng trầm tương ứng dải tần số cao hay thấp.

Khi nói, không khí được đẩy từ phổi qua miệng và khoang mũi. Luồng không khí này bị cản trở và thay đổi do hoạt động của lưỡi và môi. Điều này tạo ra các co giãn của không khí, sóng âm, âm thanh. Những âm thanh tạo thành, tương ứng với nguyên âm và phụ âm, thường được gọi là âm vị. Các âm vị được kết hợp với nhau tạo thành từ. Mỗi âm vị được hình thành trong quá trình nói phụ thuộc vào ngữ cảnh của nó, tức là phụ thuộc vào âm vị đứng ngay trước và âm vị đứng ngay sau nó. Tuy nhiên, tiếng nói không chỉ là chuỗi các âm vị tạo thành từ và câu. Có nhiều thành phần của tiếng nói mang thông tin, ví dụ: phong cách nói thể hiện qua âm điệu, cao độ, âm lượng, ngữ điệu. Thông tin này đôi khi được gọi là thông tin phi từ vựng (paralinguistic) của tiếng nói và cũng được sử dụng trong nhiều phương pháp nhằm nâng cao độ chính xác của hệ thống nhận diện.

2.2 Hệ thống ngữ âm tiếng M’Nông

Tiếng M’Nông là một ngôn ngữ thuộc nhóm ngôn ngữ Môn-Khmer, được sử dụng bởi các dân tộc M’Nông ở Việt Nam, Campuchia và Lào. Tiếng M’Nông có hệ thống ngữ âm tương đối phức tạp, với 6 thanh điệu, 44 âm vị, bao gồm 26 phụ âm và 18 nguyên âm được phân loại dựa trên cách phát âm.

Ngoài ra, tiếng M’Nông còn có các phụ âm và nguyên âm biến thể, được tạo thành bằng cách kết hợp các âm vị cơ bản.

Tính chất ngữ âm của tiếng M’Nông: Tiếng M’Nông là một ngôn ngữ có trật tự âm tiết mở, nghĩa là âm tiết bắt buộc phải có nguyên âm. Tiếng M’Nông cũng là một ngôn ngữ có phân chia âm tiết thành âm tiết chính và âm tiết phụ. Âm tiết chính là âm tiết chứa thanh điệu và âm tiết phụ là âm tiết không chứa thanh điệu.

Thanh điệu tiếng M’Nông có thể được phân loại thành 2 nhóm chính: thanh điệu đơn và thanh điệu kép. Thanh điệu đơn là thanh điệu chỉ có một độ cao nhất định. Thanh điệu kép là thanh điệu có hai độ cao.

2.3 Khái quát về công nghệ nhận diện tiếng nói

2.3.1 Khái niệm

ASR (Automatic Speech recognition) là một nhánh của Học máy (Machine Learning – ML). Về cơ bản, thay vì lập trình các quy tắc để chuyển đổi dữ liệu đầu vào (giọng nói) thành đầu ra (văn bản), thì mô hình Học máy được đào tạo bằng cách đưa các tập dữ liệu lớn vào một thuật toán, chẳng hạn như mô hình Whisper. Trải qua quá trình đào tạo, mô hình ngày càng suy luận tốt hơn, và có khả năng nhận dạng tiếng nói của con người.

2.3.2 Cách thức hoạt động

Để chuyển giọng nói sang văn bản, hệ thống phải thực hiện một quá trình gồm nhiều bước phức tạp. Khi nói, ta sẽ tạo ra những rung động trong không khí. Bộ chuyển đổi tín hiệu tương tự sang số (Analog-to-Digital Converter - ADC) chuyển các sóng tương tự (analog) này thành dữ liệu mà máy có thể hiểu được.

Để làm điều này, hệ thống thu thập các mẫu (hoặc số hóa) âm thanh bằng cách đo chính xác sóng âm ở các khoảng thời gian gần nhau, sau đó lọc âm thanh đã được số hoá để loại bỏ tiếng ồn, đôi khi tách chúng thành các dải tần số khác nhau. Nó cũng tinh chỉnh âm thanh đến một mức âm lượng không thay đổi hoặc sắp xếp theo thời gian. Không phải lúc nào con người cũng nói với tốc độ như nhau nên âm thanh phải được điều chỉnh cho phù hợp với tốc độ mà âm thanh mẫu được ghi nhận trong bộ nhớ máy.

Tiếp theo, tín hiệu được chia thành nhiều phần nhỏ (thời gian khoảng vài phần trăm giây, thậm chí là phần ngàn giây trong trường hợp có phụ âm cuối khó phân biệt). Chương trình sau đó đặt những phần âm thanh này vào các âm vị có sẵn trong ngôn ngữ thích hợp.

2.4 Mô hình Whisper

2.4.1 Khái niệm

Mô hình Whisper là một mô hình nhận dạng tiếng nói đa ngôn ngữ và đa nhiệm, được phát triển bởi OpenAI. Mô hình này được huấn luyện trên một tập dữ liệu khổng lồ gồm 680.000 giờ âm thanh tiếng nói, bao gồm 96 ngôn ngữ khác nhau.

Mô hình Whisper hoạt động theo hướng end-to-end, sử dụng kiến trúc mạng nơ-ron nhân tạo Transformer.

Mô hình Whisper đã đạt được kết quả ấn tượng trong các thử nghiệm. Độ chính xác của mô hình trong các tác vụ nhận dạng giọng nói và dịch giọng nói tương đương với các mô hình nhận dạng tiếng nói chuyên biệt cho từng ngôn ngữ.

2.4.2 Cách thức hoạt động

Whisper là một công cụ được tạo ra bởi OpenAI có khả năng hiểu và chuyển đổi ngôn ngữ nói thành văn bản, tương tự như cách Siri hoặc Alexa hoạt động. Cách mà Whisper hoạt động giống như một thông dịch viên. Nó sử dụng một kiến trúc mã hóa-giải mã Transformer. Để đơn giản hóa, hãy tưởng tượng ai đó dịch từ tiếng Anh sang tiếng Việt. Phần 'mã hóa' sẽ hiểu câu tiếng Anh, và phần 'giải mã' sau đó sẽ tạo ra câu tiếng Việt. Trong Whisper, phần 'mã hóa' hiểu ngôn ngữ nói (âm thanh), và phần 'giải mã' tạo ra văn bản.

Để học cách hiểu và chuyển đổi ngôn ngữ nói, Whisper đã được đào tạo bằng cách sử dụng một lượng lớn dữ liệu từ internet - tương đương với việc liên tục lắng nghe suốt hơn 77 năm. Dữ liệu này bao gồm nhiều ngôn ngữ khác nhau và đa nhiệm (bao gồm nhiều loại nhiệm vụ khác nhau, không chỉ là chuyển đổi). Dữ liệu âm thanh mà Whisper học được xử lý theo một cách cụ thể để làm cho hệ thống dễ hiểu hơn. Điều này giống như khi bạn điều chỉnh cài đặt trên TV để làm cho hình ảnh rõ ràng hơn. Âm thanh được 'tái tạo' thành chất lượng chuẩn (16.000 Hz), sau đó nó được biến đổi thành biểu đồ thị trực quan mà hệ thống có thể học. Quá trình biến đổi này được thực hiện trong các phần nhỏ chồng lên nhau để đảm bảo không có phần nào của âm thanh bị bỏ lỡ.

2.4.3 Các thuật toán liên quan

Mô hình Whisper sử dụng các thuật toán sau:

- Kiến trúc mạng nơ-ron nhân tạo Transformer: Kiến trúc Transformer là một kiến trúc mạng nơ-ron nhân tạo được sử dụng phổ biến trong các tác vụ xử lý ngôn ngữ tự nhiên, chẳng hạn như dịch máy, tóm tắt văn bản, và nhận dạng tiếng nói. Kiến trúc Transformer có khả năng xử lý thông tin theo hướng end-to-end, giúp mô hình có thể học hỏi các mối quan hệ phức tạp giữa các từ và các đoạn văn bản.

- Kỹ thuật chuyển đổi attention: Kỹ thuật chuyển đổi attention là một kỹ thuật được sử dụng trong kiến trúc Transformer để giúp mô hình tập trung vào các thông tin quan trọng trong một đoạn văn bản. Kỹ thuật này hoạt động bằng cách tính toán một trọng số cho mỗi từ trong đoạn văn bản, dựa trên mức độ liên quan của từ đó với từ đang được xử lý.

- Kỹ thuật học có giám sát: Kỹ thuật học có giám sát là một kỹ thuật học máy trong đó mô hình được huấn luyện trên một tập dữ liệu có nhãn. Trong trường hợp của mô hình Whisper, tập dữ liệu huấn luyện bao gồm các mẫu âm thanh tiếng nói và nhãn của các mẫu âm thanh đó. Mô hình sẽ học cách phân loại các mẫu âm thanh dựa trên các nhãn đã cung cấp.

Ngoài các thuật toán trên, mô hình Whisper còn sử dụng một số kỹ thuật khác để cải thiện hiệu quả của mô hình:

- Kỹ thuật giảm kích thước: Kỹ thuật giảm kích thước được sử dụng để giảm kích thước của các mảng dữ liệu, giúp mô hình có thể chạy nhanh hơn và hiệu quả hơn.

- Kỹ thuật tăng tốc: Kỹ thuật tăng tốc được sử dụng để tăng tốc độ của mô hình, chẳng hạn như sử dụng các kỹ thuật tính toán song song hoặc sử dụng các phần cứng chuyên dụng.

2.4.4 Ý nghĩa của mô hình Whisper trong nhận diện tiếng nói

Với 680,000 giờ âm thanh được gán nhãn, bộ dữ liệu của Whisper là một trong những bộ dữ liệu lớn nhất từng được tạo ra trong lĩnh vực nhận diện tiếng nói có giám sát.

Mô hình Whisper trong hệ thống nhận diện giọng nói giúp cải thiện độ chính xác của việc nhận dạng giọng nói trong các môi trường có tiếng ồn hoặc khi người

nói nói nhỏ. Điều này có thể được ứng dụng trong nhiều lĩnh vực như: trợ lý ảo, tạo phụ đề, dịch ngôn ngữ, bảo mật,…

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1 Xử lý dữ liệu

Theo xu hướng của các nghiên cứu gần đây, chúng tôi áp dụng một phương pháp tối giản để xử lý dữ liệu. Khác với các hệ thống ASR khác, mô hình Whisper có thể dự đoán bản thô của audio mà không cần bất kỳ tiêu chuẩn hay ràng buộc nào. Chúng tôi phân đoạn âm thanh thành các đoạn âm thanh nhỏ hơn, mỗi đoạn tương ứng với một câu hoặc một từ. Tất cả các đoạn này được huấn luyện và sử dụng làm dữ liệu huấn luyện.

Nhiều bản ghi trên internet thực tế không phải do con người tạo ra mà là kết quả của các hệ thống ASR. Nghiên cứu gần đây đã chỉ ra rằng việc huấn luyện trên các bộ dữ liệu chứa cả dữ liệu được tạo ra bởi con người và máy có thể ảnh hưởng đáng kể đến hiệu suất của các hệ thống dịch. Để tránh việc học "ngôn ngữ bản ghi" không tự nhiên, chúng tôi xây dựng bộ dữ liệu âm thanh từ chính người dân địa phương nói tiếng M’Nông tại huyện Lắk của tỉnh Đắk Lắk, điều này làm tăng cường độ tin cậy cho bộ dữ liệu và chất lượng âm thanh tốt cũng giúp mô hình hoạt động mạnh mẽ hơn.

3.2 Mô hình

Chúng tôi sử dụng một kiến trúc có sẵn là Transformer mã hóa-giải mã vì kiến trúc này đã được xác minh có thể mở rộng và đáng tin cậy. Tất cả âm thanh được lấy mẫu thành 16.000 Hz, và biểu diễn quang phổ Mel với cường độ log 80 channels được tính toán trên cửa sổ 25 mili giây với bước nhảy là 10 mili giây.

Để chuẩn hóa các đặc trưng, đầu vào được tỷ lệ từ -1 đến 1 với giá trị trung bình xấp xỉ 0 trên toàn bộ dữ liệu tiền đào tạo. Bộ mã hóa xử lý biểu diễn đầu vào này với một tầng stem nhỏ bao gồm hai tầng tích chập với chiều rộng bộ lọc là 3 và hàm kích hoạt GELU (Hendrycks & Gimpel, 2016), trong đó tầng tích chập thứ hai có bước nhảy là 2. Phần nhúng có vị trí hình sin sau đó được thêm vào đầu ra của stem, tiếp đó là áp dụng các khối Transformer của bộ mã hóa. Transformer sử dụng các khối pre-activation và áp dụng một lớp chuẩn hóa cuối cho đầu ra của bộ mã

hóa. Bộ giải mã sử dụng phần nhúng được học và biểu diễn thông tin đầu vào-đầu ra được ràng buộc. Bộ mã hóa và bộ giải mã có cùng chiều rộng và số lượng khối Transformer.

Năm 2021, Radford và cộng sự đã nhận thấy trong khi tinh chỉnh một mô hình thị giác máy tính trên bộ dữ liệu ImageNet đã tăng 9.2% độ chính xác khi phân loại đối tượng mà không có bất kỳ cải thiện nào về độ chính xác trung bình khi phân loại các đối tượng giống nhau trên 7 bộ dữ liệu khác. Một mô hình có thể đạt hiệu suất vô cùng tốt khi đào tạo trên một bộ dữ liệu vẫn có thể mắc nhiều lỗi cơ bản khi được đánh giá trên bộ dữ liệu khác. Mô hình Whisper nhóm chúng tôi chọn được đánh giá trên speech-to-text thông thường (không phải trên tiếng M'Nông) nhưng nhóm vẫn chọn vì độ chính xác ổn, có thể chấp nhận và thiết kế dễ tiếp cận.

3.3 Multitask Format

Việc dự đoán từ ngữ nào được nói trong một đoạn âm thanh là vấn đề cơ bản của nhận diện tiếng nói nhưng đó không phải là vấn đề duy nhất. Một hệ thống nhận diện giọng nói đầy đủ tính năng bao gồm nhiều thành phần bổ sung như phát hiện giọng hoạt động, phân loại người nói và chuẩn hóa văn bản. Những thành phần này thường được xử lý riêng lẻ dẫn đến một hệ thống khá phức tạp. Để giảm độ phức tạp này, mô hình Whisper có thể thực hiện toàn bộ quy trình xử lý giọng nói mà không chỉ là phần nhận diện cơ bản.

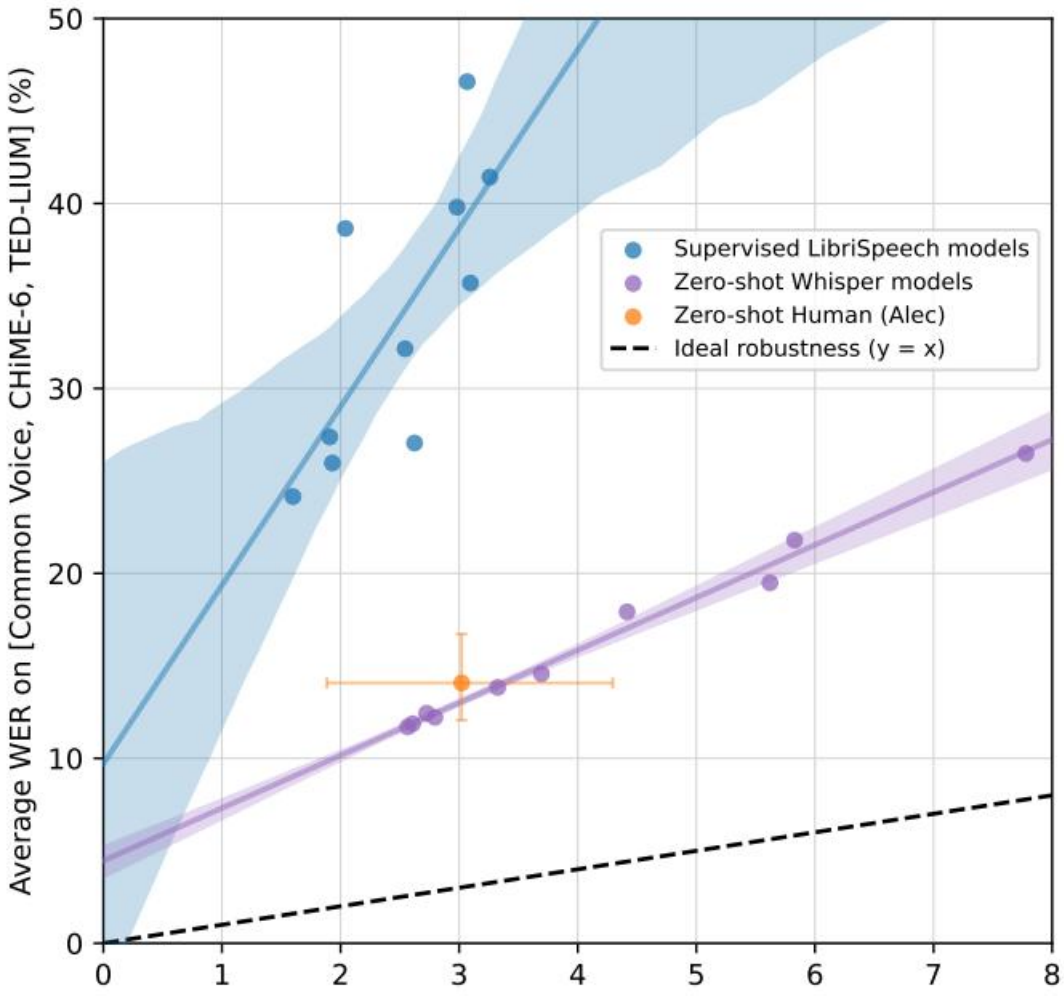
Để làm được điều này, Whisper sử dụng định dạng đơn giản để chỉ định tất cả các nhiệm vụ và thông tin điều kiện dưới dạng một chuỗi các token đầu vào cho bộ giải mã. Whisper chỉ định điểm bắt đầu trước khi dự đoán là một token <start>.

Trước hết, Whisper dự đoán ngôn ngữ đang được nói và biểu diễn bằng một token. Trong trường hợp không có giọng nói được phát hiện, mô hình được huấn luyện để dự đoán token <silence>. Token tiếp theo chỉ định nhiệm vụ "transcription" hoặc "translation" qua token <transcription> hoặc <translation>. Sau đó, mô hình dự đoán liệu có timestamp hay không bằng cách thêm một token <timestamp>. Tại đây, mọi thứ đã được chỉ định đầy đủ và bắt đầu cho ra output.

3.4 Hiệu suất

Các mô hình Whisper được đào tạo trên dữ liệu âm thanh rộng và đa dạng và được đánh giá trong cài đặt zero-shot có thể cho hiệu suất tốt hơn các hệ thống đang tồn tại hiện nay. Chúng ta có thể so sánh các mô hình Whisper với hiệu suất của con người và các mô hình học máy tiêu biểu hiện nay và kiểm tra hiệu suất của chúng.

Các mô hình Whisper zero-shot mạnh mẽ gần với con người. Dù có thể sánh ngang hoặc vượt qua hiệu suất của con người trên tập dev-clean của LibriSpeech, các mô hình LibriSpeech lại mắc khoảng gấp đôi số lỗi so với con người trên các bộ dữ liệu khác. Tuy nhiên trong trường hợp này, độ ổn định ước lượng của các mô hình Whisper zero-shot bao gồm khoảng 95% khoảng tin cậy.



Hình 2. WER trên LibriSpeech dev-clean (%)

Chúng ta xem xét tổng thể hiệu suất trung bình trên nhiều bộ dữ liệu và tính hiệu quả để đo lường sự khác biệt trong hiệu suất giữa một tập dữ liệu tham chiếu và một hoặc nhiều tập dữ liệu bên ngoài. Nhóm nghiên cứu Whisper sử dụng LibriSpeech làm tập dữ liệu tham chiếu do vai trò quan trọng của nó trong nghiên cứu nhận diện giọng nói ngày nay và có nhiều mô hình được phát hành dựa trên LibriSpeech.

Dataset	wav2vec 2.0 Large (no LM)	Whisper Large V2	RER (%)
LibriSpeech Clean	**2.7**	**2.7**	0.0
Artie	24.5	**6.2**	74.7
Common Voice	29.9	**9.0**	69.9
Fleurs En	14.6	**4.4**	69.9
Tedlium	10.5	**4.0**	61.9
CHiME6	65.8	**25.5**	61.2
VoxPopuli En	17.9	**7.3**	59.2
CORAAL	35.6	**16.2**	54.5
AMI IHM	37.0	**16.9**	54.3
Switchboard	28.3	**13.8**	51.2
CallHome	34.8	**17.6**	49.4
WSJ	7.7	**3.9**	49.4
AMI SDM1	67.6	**36.4**	46.2
LibriSpeech Other	6.2	**5.2**	16.1
Average	29.3	**12.8**	55.2

Hình 3. So sánh tính hiệu quả trên nhiều bộ dữ liệu

Cả hai mô hình đều cho hiệu suất trong khoảng 0.1% trên LibriSpeech nhưng mô hình Whisper thực hiện tốt hơn nhiều trên các bộ dữ liệu khác và gây ra ít lỗi hơn trung bình 55.2%. Kết quả được đánh giá dưới dạng tỉ lệ lỗi từ (WER) cho cả hai mô hình.

Mặc dù mô hình Whisper cho một tỉ lệ lỗi WER trên LibriSpeech clean-test khá bình thường là 2.5, tương đương với hiệu suất của một mô hình cơ bản vào giữa

năm 2019, nhưng mô hình Whisper có các đặc tính mạnh mẽ khác biệt so với các mô hình LibriSpeech được giám sát và vượt qua tất cả các mô hình LibriSpeech trên các bộ dữ liệu khác. Ngay cả mô hình Whisper nhỏ nhất, có chỉ 39 triệu tham số và tỷ lệ lỗi WER trên LibriSpeech test-clean là 6.7, cũng vượt trội hơn so với mô hình LibriSpeech tốt nhất khi được đánh giá trên các bộ dữ liệu khác. Khi so sánh với một con người, những mô hình Whisper cho thấy độ chính xác và tính mạnh mẽ của chúng khá tương đồng. Mặc dù hiệu suất rất gần nhau nhưng mô hình Whisper đạt được độ giảm lỗi trung bình là 55.2% khi được đánh giá trên các bộ dữ liệu nhận diện giọng nói khác.

3.5 Hướng tiếp cận

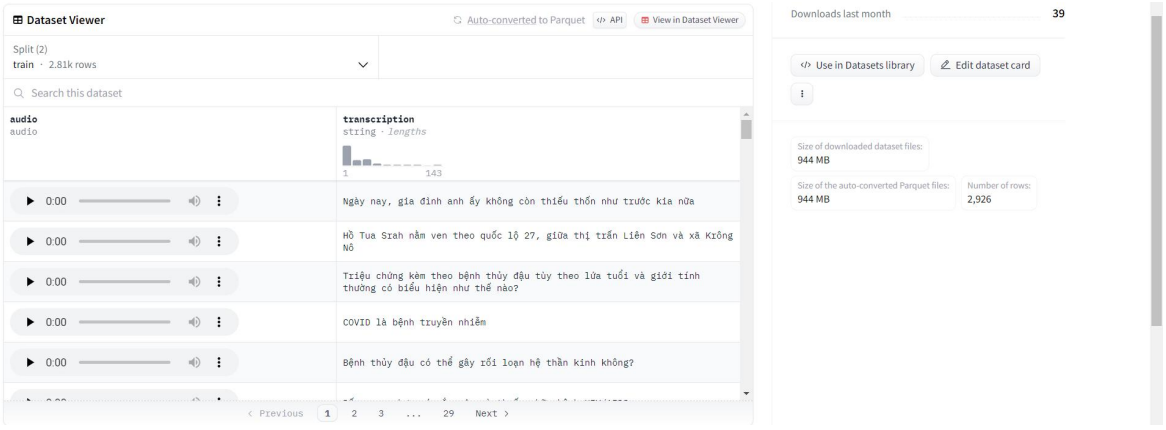
Bài toán được đặt ra là biên dịch tiếng M'Nông ở dạng âm thanh thành văn bản tiếng Việt. Trong ngôn ngữ nói hàng ngày của người dân tộc thiểu số nói chung và dân tộc M'Nông nói riêng, họ sử dụng rất nhiều từ mượn của tiếng nói phổ thông. Nên ta sẽ muốn trong quá trình phiên dịch, mô hình phải vừa phát hiện được tiếng M'Nông lẫn tiếng Việt thông thường. Tiếng Việt và tiếng M'Nông có cùng cấu trúc ngữ pháp giờ đây trở thành một đặc điểm ta có thể tận dụng được để tìm phương pháp xử lý bài toán: tinh chỉnh (fine-tune) lại một mô hình phát hiện tiếng Việt phổ thông.

Nếu ta coi việc biên dịch tiếng M'Nông thành tiếng Việt là đang phiên âm tiếng Việt với các cách phát âm khác nhau của cùng một từ ngữ, hướng đi hiệu quả nhất cho bài toán sẽ là tinh chỉnh lại mô hình phát hiện tiếng nói tự động dựa trên bộ dữ liệu tiếng M'Nông đã thu được. Khi tinh chỉnh, các từ tiếng Việt xuất hiện trong câu sẽ được dịch y nguyên, giúp ta không cần phải huấn luyện lại phần từ vựng tiếng Việt cho mô hình. Với mô hình Whisper trên - đã được huấn luyện cho rất nhiều ngôn ngữ trong đó có tiếng Việt, việc huấn luyện sẽ thuận lợi đáng kể.

CHƯƠNG 4. THỰC NGHIỆM

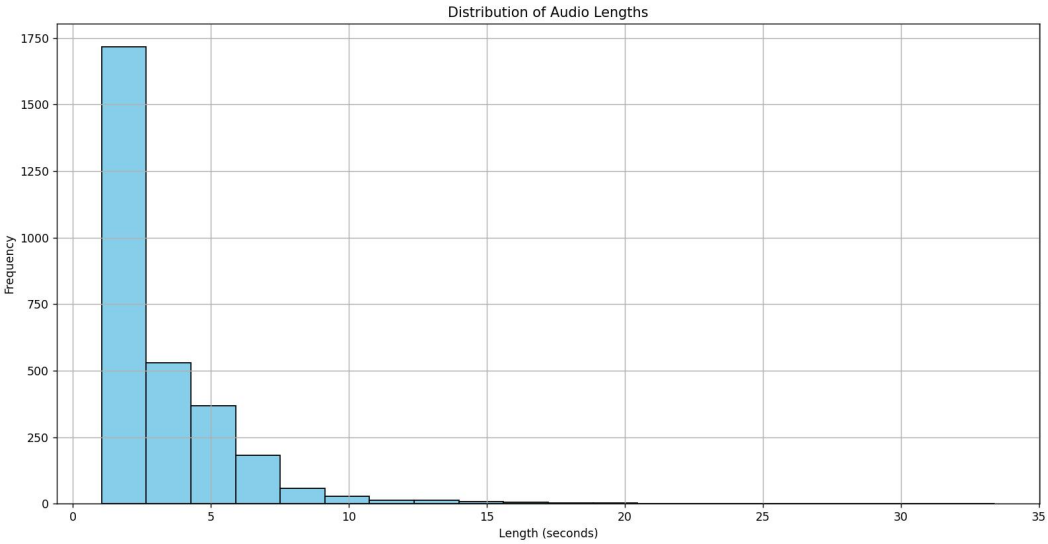
4.1 Dữ liệu thực nghiệm

Dữ liệu được lưu trữ trên nền tảng lưu trữ của Hugging Face. Dữ liệu được chia làm 2 phần: cột "audio" chứa file audio tiếng M'Nông và cột "transcription" chứa bản dịch tiếng Việt của audio tương ứng. Bộ dữ liệu sẽ được chia thành các bộ nhỏ hơn, mang vai trò tương ứng như: "train", "test".



Hình 4. Bộ dữ liệu tiếng M'Nông được đẩy lên Hugging Face

Chi tiết bộ dữ liệu: 2926 file ghi âm, bao gồm 1338 file thu âm câu, 1588 file thu âm từ với tổng thời lượng là 2 giờ 34 phút. Bộ dữ liệu này mới chỉ gồm có 3 giọng đọc, thời lượng còn khá thấp, nhưng cũng vừa đủ để ta thử nghiệm phương án được đưa ra.



Hình 5. Phân bố độ dài của các file âm thanh trong bộ dữ liệu.

4.2 Cài đặt thực nghiệm

Những bước tiến hành cài đặt thực nghiệm sau đây dựa trên "Step-by-step Guide" được cung cấp bởi chính nhóm phát triển mô hình Whisper.

4.2.1 Load Dataset

Đầu tiên, ta tải bộ dữ liệu đã được lưu trên Hugging Face Hub. Sau đó lấy phần "train" và "test" của bộ dữ liệu.

```
[ ]  from datasets import load_dataset, DatasetDict

     dataset = DatasetDict()

     dataset["train"] = load_dataset("legendary2910/MnongAudio", "default", split="train", use_auth_token=True)
     dataset["test"] = load_dataset("legendary2910/MnongAudio", "default", split="test", use_auth_token=True)
```

Hình 6. Load dataset

```
[ ]  print(dataset)

     DatasetDict({
         train: Dataset({
             features: ['audio', 'transcription', 'id'],
             num_rows: 1056
         })
         test: Dataset({
             features: ['audio', 'transcription', 'id'],
             num_rows: 1056
         })
     })
```

Hình 7. Kết quả của load dataset

4.2.2 Chuẩn bị Tokenizer, Feature Extractor và xử lý data

Quy trình ASR có thể được phân rã thành ba thành phần:

1. Một Feature Extractor có tác dụng tiền xử lý dữ liệu thô.
2. Một mô hình thực hiện nhiệm vụ sequence-to-sequence mapping.
3. Một Tokenizer thực hiện xử lý kết quả trả về thành văn bản.

Trong Transformer, mô hình Whisper đi kèm với một bộ feature extractor và một bộ tokenizer lần lượt là WhisperFeatureExtractor và WhisperTokenizer.

4.2.3 Load WhisperFeatureExtractor

Tốc độ của nguồn âm thanh đầu vào phải phù hợp với tốc độ lấy mẫu mà mô hình của chúng ta lựa chọn. Nếu lấy một mẫu âm thanh với tốc độ lấy mẫu là 16kHz và nghe nó với tốc độ lấy mẫu là 8kHz sẽ khiến âm thanh nghe như đang chạy ở nửa tốc độ. Tương tự, truyền âm thanh với tốc độ lấy mẫu không đúng có thể làm sai lệch mô hình ASR. Bộ feature extractor của Whisper cần đầu vào âm thanh với tốc độ lấy mẫu là 16kHz, vì vậy chúng ta cần tinh chỉnh đầu vào với dữ liệu theo giá trị này.

Bộ feature extractor của Whisper thực hiện hai hoạt động. Trước hết, nó chia độ dài của các mẫu âm thanh sao cho tất cả các mẫu có chiều dài đầu vào là 30 giây. Các mẫu ngắn hơn 30 giây được thêm độ dài bằng cách thêm các số không vào cuối

chuỗi (số không trong tín hiệu âm thanh tương ứng với không có tín hiệu hoặc im lặng). Các mẫu dài hơn 30 giây được cắt bớt thành 30 giây.

Hoạt động thứ hai mà bộ feature extractor của Whisper thực hiện là chuyển đổi các mảng âm thanh đã được chia độ dài thành các biểu đồ tần số log-Mel.

```
[ ]  from transformers import WhisperTokenizer

     tokenizer = WhisperTokenizer.from_pretrained("openai/whisper-small", language="Vietnamese", task="transcribe")
```

Hình 8. Load WhisperFeatureExtractor

4.2.4 Load WhisperTokenizer

Mô hình Whisper xuất ra các token văn bản chỉ định chỉ số của văn bản dự đoán trong từ điển các mục từ vựng. Tokenizer ánh xạ một chuỗi các token văn bản thành chuỗi văn bản thực tế (ví dụ: [1169, 3797, 3332] -> "the cat sat").

Trong trước đây, khi sử dụng các mô hình chỉ có trình mã hóa cho ASR, chúng ta giải mã bằng cách sử dụng Connectionist Temporal Classification (CTC). Ở đây, chúng ta cần đào tạo một tokenizer CTC cho mỗi bộ dữ liệu chúng ta sử dụng. Một trong những ưu điểm của việc sử dụng kiến trúc mã hóa-giải mã là chúng ta có thể trực tiếp tận dụng tokenizer từ mô hình được đào tạo trước.

Tokenizer của Whisper được đào tạo trên các bản ghi chép cho 96 ngôn ngữ được đào tạo trước. Do đó, nó có một byte-pair mở rộ rất phù hợp cho hầu hết các ứng dụng ASR đa ngôn ngữ.

```
[ ]  from transformers import WhisperTokenizer

     tokenizer = WhisperTokenizer.from_pretrained("openai/whisper-small", language="Vietnamese", task="transcribe")
```

Hình 9. Load WhisperTokenizer

4.2.5 Kết hợp để tạo ra WhisperProcessor

Để đơn giản hóa việc sử dụng bộ feature extractor và tokenizer, chúng ta có thể kết hợp cả hai vào một lớp duy nhất là WhisperProcessor. Đối tượng này kế thừa từ WhisperFeatureExtractor và WhisperProcessor và có thể được sử dụng trên đầu vào âm thanh và dự đoán của mô hình khi cần thiết.

```
[ ]  from transformers import WhisperProcessor

     processor = WhisperProcessor.from_pretrained("openai/whisper-small", language="Vietnamese", task="transcribe")
```

Hình 10. Tạo WhisperProcessor

4.2.6 Chuẩn bị dữ liệu

Bây giờ hãy in ra ví dụ đầu tiên trong bộ dữ liệu để xem dữ liệu ở dạng như thế nào.

```
[ ]  print(dataset["train"][0])

     {'audio': {'path': 'audio_w65.wav', 'array': array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
            6.10351562e-05, 1.83105469e-04, 2.13623047e-04]), 'sampling_rate': 48000}, 'transcription': 'Cầm'}
```

Hình 11. In data đầu tiên

Có thể thấy rằng chúng ta có một mảng âm thanh đầu vào 1 chiều và bản dịch tương ứng. Chúng ta đã nói về tầm quan trọng của tốc độ lấy mẫu và thực tế là chúng ta cần phải điều chỉnh tốc độ lấy mẫu của âm thanh với tốc độ của mô hình Whisper (16kHz). Vì âm thanh đầu vào của chúng ta được lấy mẫu ở 48kHz, cần giảm mẫu xuống 16kHz trước khi truyền vào bộ feature extractor của Whisper.

Chúng ta sẽ đặt tốc độ lấy mẫu của đầu vào âm thanh đúng bằng cách sử dụng phương thức cast_column của bộ dữ liệu. Thao tác này không thay đổi âm thanh ngay tại chỗ, mà thay vào đó báo hiệu cho bộ dữ liệu để resample các mẫu âm thanh trực tiếp lần đầu tiên chúng được tải.

```
[ ]  from datasets import Audio

     dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
```

Hình 12. Điều chỉnh tốc độ lấy mẫu

Việc tải lại mẫu âm thanh sẽ resample nó về tốc độ chúng ta mong muốn.

```
print(dataset["train"][0])

{'audio': {'path': 'audio_w65.wav', 'array': array([ 6.36646291e-12,  1.72803993e-11, -1.81898940e-11, ...,
       -1.58221950e-03, -7.20107753e-04,  7.26098660e-05]), 'sampling_rate': 16000}, 'transcription': 'Cầm'}
```

Hình 13. Resample mẫu âm thanh về tốc độ chuẩn

Có thể thấy rằng tốc độ lấy mẫu đã được giảm xuống còn 16kHz, giá trị mảng cũng khác nhau.

Bây giờ chúng ta có thể viết một hàm để chuẩn bị dữ liệu sẵn sàng cho mô hình:

1. Tải và resample dữ liệu âm thanh bằng cách gọi batch[“audio”]

2. Sử dụng bộ feature extractor để tính các đặc trưng đầu vào log-Mel spectrogram từ mảng âm thanh 1 chiều.

3. Mã hóa transcriptions thành các nhãn thông qua tokenizer.

```
[ ]  def prepare_dataset(batch):
         # load and resample audio data from 48 to 16kHz
         audio = batch["audio"]

         # compute log-Mel input features from input audio array
         batch["input_features"] = feature_extractor(audio["array"], sampling_rate=audio["sampling_rate"]).input_features[0]

         # encode target text to label ids
         batch["labels"] = tokenizer(batch["transcription"]).input_ids
         return batch
```

Hình 14. Hàm chuẩn bị dữ liệu

Áp dụng hàm chuẩn bị dữ liệu cho tất cả các ví dụ đào tạo bằng cách sử dụng phương thức .map của bộ dữ liệu.

```
[ ]  dataset = dataset.map(prepare_dataset, remove_columns=dataset.column_names["train"], num_proc=4)
```

Hình 15. Áp dụng hàm chuẩn bị dữ liệu

Đến đây chúng ta đã chuẩn bị đầy đủ dữ liệu cho quá trình huấn luyện.

4.2.7 Huấn luyện và đánh giá

Để Trainer thực hiện các công việc liên quan đến huấn luyện, chúng ta cần:

1. Xác định data collator: Lấy dữ liệu đã được xử lý trước và chuẩn bị các tensor PyTorch.

2. Đo độ chính xác: Cần một hàm compute_metrics để xử lý tính toán các chỉ số lỗi từ vựng (WER) để đánh giá mô hình.

3. Tải một pre-trained chechkpoint và cấu hình nó cho quá trình huấn luyện.

4. Xác định đối số được sử dụng bởi Trainer trong khi lên lịch huấn luyện.

4.2.8 Định nghĩa Data Collator

Bộ thu thập dữ liệu cho một mô hình nói chung sequence-to-sequence xử lý input_features và nhãn một cách độc lập: input_features phải được xử lý bởi bộ feature extractor và gán nhãn bởi tokenizer.

Các input_features đã được chia độ dài thành 30 giây và chuyển đổi thành log-Mel spectrogram có kích thước cố định, vì vậy tất cả những gì chúng ta cần làm là chuyển đổi chúng thành các tensor PyTorch. Chúng ta thực hiện điều này bằng cách sử dụng phương thức .pad của bộ feature extractor với return_tensors=pt.

Ngược lại, các nhãn không được chia độ dài. Đầu tiên, chúng ta chia độ dài các chuỗi đến độ dài tối đa trong lô bằng cách sử dụng phương thức .pad của tokenizer. Các token sau đó được thay thế bằng -100 để những token này không được tính vào khi tính toán mất mát. Sau đó, chúng ta cắt bỏ token bắt đầu của bản trích từ đầu của chuỗi nhãn vì chúng ta sẽ thêm nó vào sau này trong quá trình huấn luyện.

Chúng ta có thể tận dụng WhisperProcessor đã định nghĩa trước đó để thực hiện cả hai thao tác feature extractor và tokenizer. Sau đó khởi tạo bộ thu thập dữ liệu vừa định nghĩa.

```
[ ]  import torch

     from dataclasses import dataclass
     from typing import Any, Dict, List, Union

     @dataclass
     class DataCollatorSpeechSeq2SeqWithPadding:
         processor: Any

         def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
             # split inputs and labels since they have to be of different lengths and need different padding methods
             # first treat the audio inputs by simply returning torch tensors
             input_features = [{"input_features": feature["input_features"]} for feature in features]
             batch = self.processor.feature_extractor.pad(input_features, return_tensors="pt")

             # get the tokenized label sequences
             label_features = [{"input_ids": feature["labels"]} for feature in features]
             # pad the labels to max length
             labels_batch = self.processor.tokenizer.pad(label_features, return_tensors="pt")

             # replace padding with -100 to ignore loss correctly
             labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -100)

             # if bos token is appended in previous tokenization step,
             # cut bos token here as it's append later anyways
             if (labels[:, 0] == self.processor.tokenizer.bos_token_id).all().cpu().item():
                 labels = labels[:, 1:]

             batch["labels"] = labels

             return batch
     data_collator = DataCollatorSpeechSeq2SeqWithPadding(processor=processor)
```

Hình 16. Định nghĩa và khởi tạo Data Collator

4.2.9 Số liệu đánh giá

Tiếp theo, cần xác định chỉ số đánh giá mà chúng ta sẽ sử dụng trên tập đánh giá của mình. Ta sẽ sử dụng chỉ số Word Error Rate (WER), chỉ số 'de-facto' để đánh giá các hệ thống ASR.

```
[ ]  import evaluate

     metric = evaluate.load("wer")
```

Hình 17. Xác định WER

Sau đó, chúng ta chỉ cần định nghĩa một hàm lấy dự đoán của mô hình và trả về chỉ số WER. Hàm này được gọi là compute_metrics, đầu tiên thay thế -100 bằng pad_token_id trong label_ids (hoàn ngược bước chúng ta áp dụng trong bộ thu thập dữ liệu để bỏ qua đúng các token). Sau đó, nó giải mã predicted và label ids thành chuỗi. Cuối cùng, nó tính toán chỉ số WER giữa các dự đoán và nhãn tham chiếu.

```
[ ]  def compute_metrics(pred):
         pred_ids = pred.predictions
         label_ids = pred.label_ids

         # replace -100 with the pad_token_id
         label_ids[label_ids == -100] = tokenizer.pad_token_id

         # we do not want to group tokens when computing the metrics
         pred_str = tokenizer.batch_decode(pred_ids, skip_special_tokens=True)
         label_str = tokenizer.batch_decode(label_ids, skip_special_tokens=True)

         wer = 100 * metric.compute(predictions=pred_str, references=label_str)

         return {"wer": wer}
```

Hình 18. Hàm trả về WER

4.2.10 Tải pre-trained checkpoint

```
[ ]  from transformers import WhisperForConditionalGeneration

     model = WhisperForConditionalGeneration.from_pretrained("openai/whisper-small")
```

Hình 19. Tải một pre-trained checkpoint

Mô hình Whisper có khả năng dự đoán những ngôn ngữ khác nhau xuất hiện trong câu được huấn luyện. Điều này khiến cho việc huấn luyện và dự đoán gặp khó khăn khi những từ mới của tiếng M'Nông xuất hiện có thể được mô hình dự đoán thành một ngôn ngữ khác. Nên ra ép mô hình phải sử dụng tiếng Việt và tác vụ phiên âm bằng việc cấu hình lại mô hình.

```
model.generation_config.language = "<|vi|>"
model.generation_config.task = "transcribe"
```

Hình 20. Cấu hình lại mô hình

4.2.11 Định nghĩa Traing Arguments

Trong bước cuối cùng, chúng ta định nghĩa tất cả các tham số liên quan đến quá trình huấn luyện. Một phần của các tham số được giải thích dưới đây:

- `output_dir`: thư mục cục bộ để lưu trữ trọng số của mô hình. Đây cũng sẽ là tên kho chứa trên Hugging Face Hub.

- `generation_max_length`: số lượng tối đa các token để tạo ra theo cách phát sinh tự động trong quá trình đánh giá.

- `save_steps`: trong quá trình huấn luyện, các điểm kiểm tra trung gian sẽ được lưu và tải lên bất đồng bộ lên Hub mỗi save_steps.

- `eval_steps`: trong quá trình huấn luyện, đánh giá các điểm kiểm tra trung gian sẽ được thực hiện mỗi eval_steps.

- `report_to`: nơi lưu trữ nhật ký huấn luyện. Các nền tảng được hỗ trợ là "azure_ml", "comet_ml", "mlflow", "neptune", "tensorboard" và "wandb". Chọn nền tảng ưa thích của bạn hoặc để là "tensorboard" để đăng nhập vào Hub.

```
[ ] from transformers import Seq2SeqTrainingArguments

    training_args = Seq2SeqTrainingArguments(
        output_dir="legendary2910/Mnong-ASR",  # change to a repo name of your choice
        per_device_train_batch_size=16,
        gradient_accumulation_steps=1,  # increase by 2x for every 2x decrease in batch size
        learning_rate=1e-5,
        warmup_steps=500,
        max_steps=4000,
        gradient_checkpointing=True,
        fp16=True,
        evaluation_strategy="steps",
        per_device_eval_batch_size=8,
        predict_with_generate=True,
        generation_max_length=225,
        save_steps=1000,
        eval_steps=1000,
        logging_steps=25,
        report_to=["tensorboard"],
        load_best_model_at_end=True,
        metric_for_best_model="wer",
        greater_is_better=False,
        push_to_hub=True
    )
```

Hình 21. ĐỊnh nghĩa các tham số liên quan

Chúng ta có thể chuyển các đối số huấn luyện cho Trainer cùng với mô hình, bộ dữ liệu, bộ thu thập dữ liệu và hàm compute_metrics.

```
[ ]  from transformers import Seq2SeqTrainer

     trainer = Seq2SeqTrainer(
         args=training_args,
         model=model,
         train_dataset=dataset["train"],
         eval_dataset=dataset["test"],
         data_collator=data_collator,
         compute_metrics=compute_metrics,
         tokenizer=processor.feature_extractor
     )
```

Hình 22. Chuyển các đối số huấn luyện cho Trainer

Và với thao tác vừa rồi, mô hình đã sẵn sàng để huấn luyện.

4.2.12 Huấn luyện mô hình

Với tổng gần 3000 dữ liệu thu được, nhóm sử dụng dữ liệu trên để huấn luyện và trích ra 8% tổng dữ liệu (khoảng 230 dữ liệu) cho việc đánh giá thông qua phương pháp WER (sẽ đề cập trong phần tiếp theo).

Việc huấn luyện được thực hiện trên Google Colab trong tổng thời gian 3 tiếng, với các thông số kỹ thuật như sau:

- CPU: Intel(R) Xeon(R) CPU @ 2.00GHz.
- GPU: Tesla V100-SXM2-16GB.
- RAM: 52GB.

Sau khi huấn luyện xong, mô hình đã được huấn luyện sẽ được đẩy lên Huggingface Hub cho việc sử dụng sau này dưới tên “legendary2910/MNong-ASR”.

```
[ ]  trainer.train()
```

Hình 23. Huấn luyện mô hình

Output:

```
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:460: UserWarning: torch.utils.c
  warnings.warn(
`use_cache = True` is incompatible with gradient checkpointing. Setting `use_cache = False`...
[4000/4000 3:02:20, Epoch 23/24]
```

Step	Training Loss	Validation Loss	Wer
1000	0.293000	1.003265	73.689052
2000	0.009100	1.049089	66.053358
3000	0.000500	1.066912	59.429623
4000	0.000400	1.074851	59.521619

Hình 24. Kết quả huấn luyện

```
kwargs = {
    #"dataset_tags": "legendary2910/MnongAudio",
    "dataset": "MnongAudio",  # a 'pretty' name for the training dataset
    "dataset_args": "config: vi, split: test",
    "language": "vi",
    "model_name": "Whisper Small Mnong",  # a 'pretty' name for your model
    "finetuned_from": "openai/whisper-small",
    "tasks": "automatic-speech-recognition",
    "tags": "hf-asr-leaderboard",
}
trainer.push_to_hub(**kwargs)
```

```
tokenizer.push_to_hub(repo_id="legendary2910/Mnong-ASR")
```

Hình 25. Đưa kết quả lên Hugging Face Hub

4.3 Đánh giá

Trong ASR, metric thường được sử dụng cho bài toán này là Word Error Rate (WER). Nó so sánh predicted output và target transcript, giữa từ với từ (hoặc giữa ký tự với ký tự) để tìm ra sự khác biệt giữa 2 câu.

Sự khác biệt giữa 2 câu có thể có từ xuất hiện trong transcript nhưng không có trong prediction (được tính như là Deletion), 1 từ không có trong transcript nhưng lại được thêm vào prediction (coi như là Insertion), hoặc 1 từ được thay thế giữa prediction và transcript (xem như là Substitution).

Công thức tính metric khá là dễ hiểu và sử dụng. Nó tính tỉ lệ phần số từ sự khác biệt giữa transcript và prediction.

$$\text{Word Error Rate} = \frac{\text{Inserted + Deleted + Substituted}}{\text{Total words in transcript}}$$

$$= \frac{1 + 1 + 1}{6}$$

$$= 0.5$$

Hình 26. Công thức tính WER

Hiện tại có thể thấy ở hình 24, kết quả huấn luyện mô hình còn chưa tốt với WER khá cao. Nguyên nhân có thể đến từ việc dữ liệu dù đa dạng, nhưng không có quá nhiều giọng đọc cho cùng một câu, và tất cả các dữ liệu đều khá khác nhau.

4.4 Thực nghiệm và kết quả

Vì mô hình đã được lưu lại trên Huggingface Hub, ta có thể tạo một ví dụ đơn giản cho việc sử dụng mô hình trong tương lai như sau. Bằng cách này, ta có thể sử dụng mô hình để thực hiện phiên dịch những file âm thanh trong bộ dữ liệu sử dụng để đánh giá. Qua đó có thể quan sát được kết quả ra sao trong quá trình huấn luyện.

```
[ ]  import torch
     from transformers import AutoModelForSpeechSeq2Seq, AutoProcessor, pipeline
     from datasets import load_dataset


     device = "cuda:0" if torch.cuda.is_available() else "cpu"
     torch_dtype = torch.float16 if torch.cuda.is_available() else torch.float32

     model_id = "legendary2910/Mnong-ASR"

     demo_model = AutoModelForSpeechSeq2Seq.from_pretrained(
         model_id, torch_dtype=torch_dtype, low_cpu_mem_usage=True, use_safetensors=True
     )
     demo_model.to(device)

     demo_processor = AutoProcessor.from_pretrained(model_id)

     pipe = pipeline(
         "automatic-speech-recognition",
         model=demo_model,
         tokenizer=demo_processor.tokenizer,
         feature_extractor=demo_processor.feature_extractor,
         max_new_tokens=128,
         chunk_length_s=30,
         batch_size=16,
         return_timestamps=True,
         torch_dtype=torch_dtype,
         device=device,
     )
```

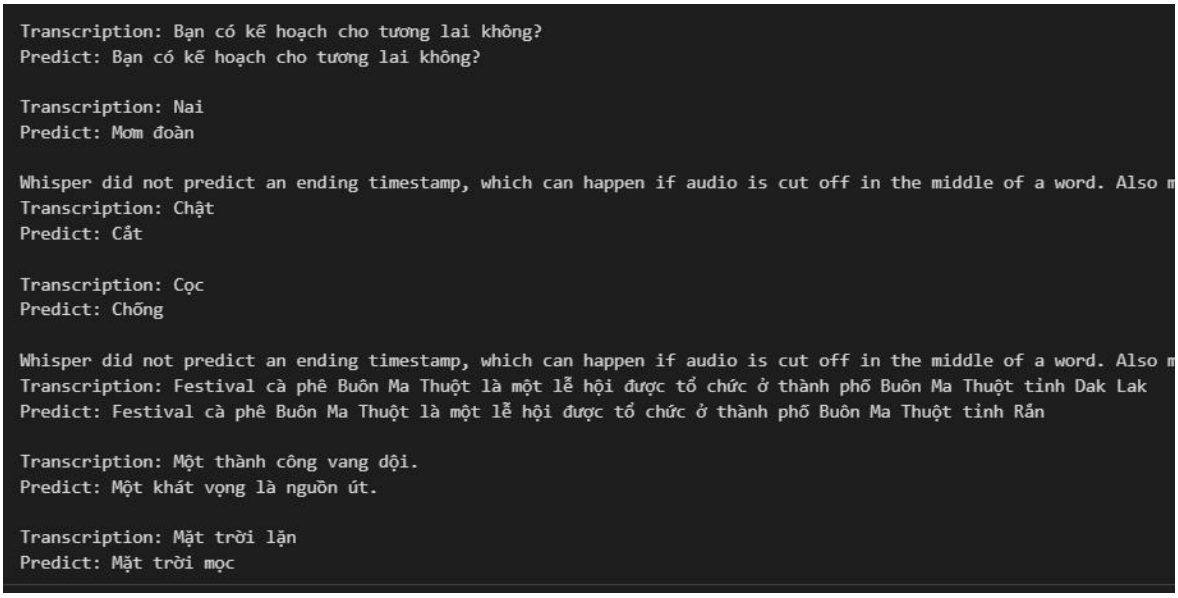
Hình 25. Ví dụ đơn giản sử dụng mô hình đã qua huấn luyện.

Hình 26. Kết quả khi thực hiện kiểm tra trên bộ dữ liệu sử dụng để kiểm định.

Ở phần chụp lại kết quả trên, phần “Transcription” hiển thị cho phần dịch đúng tương ứng với file âm thanh, phần “Predict” hiển thị kết quả mà mô hình dự

đoán ý nghĩa của câu nói có trong file âm thanh. Ta có thể thấy mô hình dự đoán những từ ta đã sử dụng trong quá trình huấn luyện với độ chính xác cao.

Vì sự hạn chế của bộ dữ liệu, vẫn có những từ ngữ mà mô hình chưa được học xuất hiện trong bộ dữ liệu đánh giá, khiến cho kết quả dự đoán bị sai đi hoặc sai hoàn toàn.



Hình 27. Kết quả khi thực hiện kiểm tra trên bộ dữ liệu sử dụng để kiểm định.

CHƯƠNG 5. ĐÁNH GIÁ VÀ KẾT LUẬN

Đây là một đề tài mới lạ đối với nhóm, yêu cầu nhiều sự tâm huyết cũng như thời gian dành cho đề tài. Việc phiên dịch tiếng nói của những dân tộc ít người có thể giúp cho công tác giáo dục tiếng nói và trong sinh hoạt hàng ngày của người dân tộc M'Nông nói riêng và người dân tộc thiểu số nói chung trở nên thuận tiện hơn. Việc chưa hề có bất kì dự án nào tương tự cũng gây nên không ít khó khăn cho nhóm khi thực hiện đề tài.

Qua việc huấn luyện trên, ta có thể thấy được bất cập của bộ dữ liệu ảnh hưởng đến chất lượng của kết quả ra sao. Bộ dữ liệu chỉ có sự tham gia của 3 giọng đọc, số lượng dữ liệu dạng từ vựng lại chiếm quá nhiều trong bộ dữ liệu; những câu dài được sử dụng để phiên dịch sang tiếng M'Nông cho quá trình thu âm lại quá phức tạp cho một ngôn ngữ thô sơ như tiếng M'Nông, làm cho việc phiên dịch trở nên khó khăn. Những lý do trên đã khiến cho kết quả của mô hình không được như mong đợi.

Ngoài ra, về mặt kỹ thuật cũng xảy ra rất nhiều vấn đề trong quá trình thực hiện. Từ xử lý việc mô hình tự động phát hiện đa ngôn ngữ, cho đến cấu hình lại mô hình, lưu trữ mô hình sau khi huấn luyện,…

Tuy có nhiều khó khăn như thế, xong kết quả có được cũng có một vài dấu hiệu khả quan, như việc mô hình đã có thể dịch được khá sát những từ vựng đã được sử dụng trong quá trình huấn luyện. Đây sẽ là nền tảng để nhóm khắc phục những nhược điểm trên nhằm cải thiện kết quả, cũng như hy vọng vào việc đề tài này có thể đem lại nhiều những giá trị thực tiễn hơn để phục vụ cộng đồng, phục vụ đất nước.

TÀI LIỆU THAM KHẢO

Tiếng Việt

…

Tiếng Anh

[1] Alcorn, M. A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.S., and Nguyen, A. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.4845–4854, 2019.

[2] Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019.

[3] Seide, F., Li, G., Chen, X., and Yu, D. Feature engineering in context dependent deep neural networks for conversational speech transcription. In 2011 IEEE Workshop on Automatic Speech Recognition & Understanding, pp.24–29. IEEE, 2011.

[4] Towards Datascience. Foundations of NLP Explained BLEU Score and WER Metrics