ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ


BÁO CÁO BÀI TẬP LỚN
Môn học: Các vấn đề hiện đại trong công nghệ thông tin

# ĐỀ TÀI: ỨNG DỤNG TRÍ TUỆ NHÂN TẠO TRONG BÀI TOÁN NHẬN DIỆN GIỌNG NÓI CHO GAME CỜ TƯỚNG

**Giảng viên:** PGS.TS Lê Sỹ Vinh

**Sinh viên:** Nguyễn Anh Đức - 19020004
Phạm Anh Cường - 19020038
Nguuyễn Văn Chiến - 19020002

Tháng 12, năm 2022

## Phân công công việc

### Các đầu việc chính:

- Học về nhận diện giọng nói
- Học cách sử dụng và tinh chỉnh các mô hình nhận diện tiếng nói.
- Xây dựng chuẩn cờ tướng (cùng team VR)
- Kết hợp với team VR để hoàn thiện sản phẩm
- Viết phần 2.3, 3, 4 của báo cáo

### Nguyễn Anh Đức - 19020004 (40%)

- Xây dựng chuẩn cờ tướng
- Xây dựng bô dự liệu dạng văn bản cho lệnh cờ tướng
- Xây dựng mô hình ngôn ngữ cho bộ dữ liệu được xây dựng
- Sử dụng beam search cho CTC để nâng cao hiệu suất
- Tích hợp code nhận diện giọng nói cho bên VR

### Phạm Anh Cường - 19020038 (30%)

- Tìm hiểu về framework wav2vec và wav2vec2
- Xây dựng chuẩn cờ tướng
- Viết phần 1, 2.1, 5.1 của báo cáo

### Nguyễn Văn Chiến - 19020002 (30%)

- Tìm hiểu về thuật toán ctc.
- Xây dựng chuẩn cờ tướng.
- Viết phần 2.2, 5.2 của báo cáo.
- Thuyết trình

# LỜI CẢM ƠN

Đầu tiên, nhóm xin gửi lời cảm ơn đến giảng viên, PGS. TS. Lê Sỹ Vinh đã tạo điều kiện cho nhóm có cơ hội được tìm hiểu và thực hiện đề tài này. Qua quá trình thực hiện bài tập lớn này thì chúng tôi đã học thêm được nhiều kiến thức mới cũng như những kỹ năng bổ ích.

Sau đây là báo cáo bài tập lớn môn Các vấn đề hiện đại trong Công nghệ thông tin, với đề tài là "Ứng dụng trí tuệ nhân tạo trong bài toán nhận diện giọng nói cho game cờ tướng". Tuy đã được rà soát kĩ lưỡng song bản báo cáo này có thể gặp phải một vài thiếu sót, vì vậy nhóm kính mong quý thầy cô có thể thông cảm và góp ý để bản báo cáo này được hoàn thiện tốt hơn. Nhóm chúng tôi xin chân thành cảm ơn.

# Mục lục

# 1 Giới thiệu

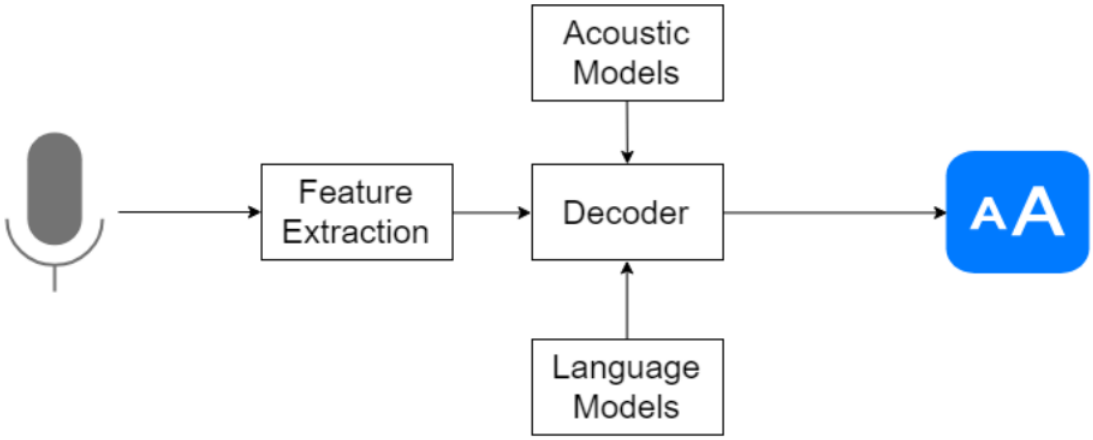
## 1.1 Giới thiệu về Nhận dạng giọng nói

Nhận dạng giọng nói là một bài toán đã xuất hiện và được nghiên cứu từ rất lâu đời, từ những năm 50 của thế kỉ XX.

Về định nghĩa: Nhận dạng giọng nói là công nghệ có thể nhận dạng các từ đã nói, sau đó có thể chuyển đổi thành văn bản.

Về cơ bản, bài toán nhận dạng giọng nói thường gồm 2 giai đoạn chính:

- **Trích xuất đặc trưng (Feature Extraction)**, trích xuất được các đặc trưng từ âm thanh đầu vào và mã hoá chúng về dạng vector.
- **Giải mã (Decoder)**: Ở bước này, ta kết hợp đầu ra từ bước Trích xuất đặc trưng, kết hợp với sự hỗ trợ của **Mô hình ngôn ngữ (Language Models)** và **Mô hình âm thành (Acoustic Models)** để xử lý và thu được đầu ra là văn bản cần thiết.




Sự phát triển của bài toán nhận dạng giọng nói chỉ thực sự bắt đầu chú ý khi ba nhà nghiên cứu của Bell Labs gồm Stephen Balashek, R. Biddulph, và KH Davis đã xây dựng một hệ thống gọi là "Audrey"để nhận dạng chữ số của một người nói vào năm 1952. Kể từ đó, nghiên cứu về nhận dạng giọng nói bắt đầu xuất hiện nhiều dần.

Ngày nay, có thể thấy sử dụng giọng nói đã trở thành một nhu cầu thiết yếu đối với mọi người, từ các ứng dụng trợ lý ảo trên điện thoại thông minh, cho đến các ý tưởng về nhà thông minh,... với độ chính xác đạt ngưỡng rất cao (với tỉ lệ lỗi từ chỉ còn khoảng 2%), Tuy nhiên, để đạt được mức độ chính xác 100% thì giới khoa học vẫn còn một chặng đường dài.

## 1.2 Trí tuệ nhân tạo trong nhận diện giọng nói

Trí tuệ nhân tạo đã mở đầu cho một bước tiến trong việc tự động hoá việc nhận diện giọng nói, bước tiến đầu tiên của sự kết hợp này bắt nguồn từ thời điểm thuật toán Dynamic Time Wrapping ra đời vào đầu những năm 70 của thế kỉ XX. Thuật toán Dynamic Time Wrapping là thuật toán nhằm so sánh độ tương đồng của 2 chuỗi thời gian mà có thể khác bước nhảy. Với sự xuất hiện của thuật toán này, các nhà khoa học đã có thể chuyển đổi âm thanh về dạng tần số và nhận diện tự động được giọng nói với độ chính xác tốt ở thời điểm bấy giờ.

Bước phát triển nhảy vọt của trí tuệ nhân tạo trong nhận diện giọng nói phải kể đến sự xuất hiện của mô hình Markov ẩn (Hidden Markov Model) được đề xuất cuối những năm 1970. Mô hình Markov ẩn là mô hình thống kê trong đó hệ thống được mô hình hóa (quá trình Markov) với các tham số không biết trước. Nhiệm vụ của quá trình Markov này là xác định các tham số ẩn từ các tham số quan sát được. Các tham số của mô hình được suy ra sau đó có thể sử dụng để thực hiện các phân tích kế tiếp.

Tuy nhiên, bước phát triển đột phá nhất phải kể đến sự xuất hiện của học sâu (deep learning) mà cụ thể trong bài toán nhận diện giọng nói là mô hình mạng neural hồi quy (recurrent neural networks) vào khoảng năm 2010. Học sâu đã cải thiện độ chính xác của các mô hình nhận diện giọng nói lên ngưỡng trên 80%, và cho đến thời điểm bây giờ, mô hình sử dụng học sâu vẫn là mô hình tốt nhất cho các tác vụ nhận diện giọng nói. Trong khi các phương pháp truyền thống tỏ ra chậm, độ chính xác còn thấp và không linh hoạt thì học sâu tuy phải huấn luyện trong thời gian lâu nhưng khi áp dụng lại thực thi rất nhanh, độ chính xác cao và có tính linh hoạt khi có thể rút ngắn thời gian hoặc cải tiến mà không ảnh hưởng đến các kết nối liên quan khi ở trong một chương trình lớn.

## 1.3 Giới thiệu về game sử dụng giọng nói

Ngày nay, trò chơi mà kết hợp các yếu tố hỗ trợ đã được nghiên cứu và phát triển ngày càng phổ biến. trên thực tế, giọng nói đã được ứng dụng vào trò chơi, đặc biệt là các trò chơi trực tuyến, nhưng chủ yếu là trên phương diện giao tiếp giữa người với người, còn tương tác giữa người và máy vẫn rất hạn chế. Dẫu vậy, các nhà sản xuất trò chơi đang tin rằng việc sử dụng giọng nói để điều khiển là một hướng đi tiềm năng với tính mới và sự thú vị, thay vì chỉ sử dụng để làm công cụ giao tiếp giữa người với người.

Việc sử dụng giọng nói cũng mang lại những ưu điểm khi có thể giúp người chơi dễ dàng tiếp cận với trò chơi, ngoài ra cũng thể mang những trò chơi này đến với những người không may bị khiếm thị hoặc khuyết tật. Ngoài ra, việc sử dụng được giọng nói để điều khiển có thể giúp người dùng đắm chìm vào trò chơi của mình. Một số loại trò chơi đã có tương tác sử dụng giọng nói có thể kể đến hiện nay như các game nhập vai (ví dụ như Bot Colony, Verbis Virtus), game kinh dị (ví dụ như Phasmophobia),...

## 1.4 Giới thiệu về bài toán của nhóm: Ứng dụng của Trí tuệ nhân tạo cho bài toán nhận diện giọng nói cho game cờ tướng

Ý tưởng về việc sử dụng giọng nói để chơi các môn thể thao trí tuệ đã có từ những năm 1963, khi Raj Reedy từ Stanford AI Lab đã thí nghiệm thành công việc chơi cờ vua sử dụng giọng nói. Hiện nay thì với các môn thể thao trí tuệ như các bộ môn cờ thì nghiên cứu và phát triển các trò chơi đã có nhiều, tuy nhiên vẫn còn đó một môn thể thao trí tuệ phổ biến ở Việt Nam mà gần như chưa được phát triển các chương trình với Tiếng Việt, đó là cờ tướng.

Vì vậy, nhóm chúng tôi đã kết hợp với nhóm làm về thực tế ảo (Virtual Reality - VR) để cùng nhau xây dựng một phần mềm chơi cờ tướng thực tế ảo có sử dụng giọng nói để điều khiển. Trong đó phần game, logic sẽ do bên VR đảm nhận, còn nhóm chúng tôi sẽ tiến hành xây dựng mô hình nhận diện giọng nói tốt nhất có thể để áp dụng vào trò cờ tướng.

Ý tưởng của nhóm là có thể sử dụng các thuật toán nhận diện giọng nói dựa trên Deep Learning để thực hiện công việc này. Tuy nhiên có thể sẽ có một số khó khăn mà mô hình cần phải vượt qua được, ví dụ như cải thiện độ chính xác, nhất là với các nhiệm vụ tiếng Việt (vì trước nay không có nhiều mô hình sử dụng tiếng Việt) và vượt qua được sự khác biệt về phương ngữ.

# 2 Các thuật toán sử dụng

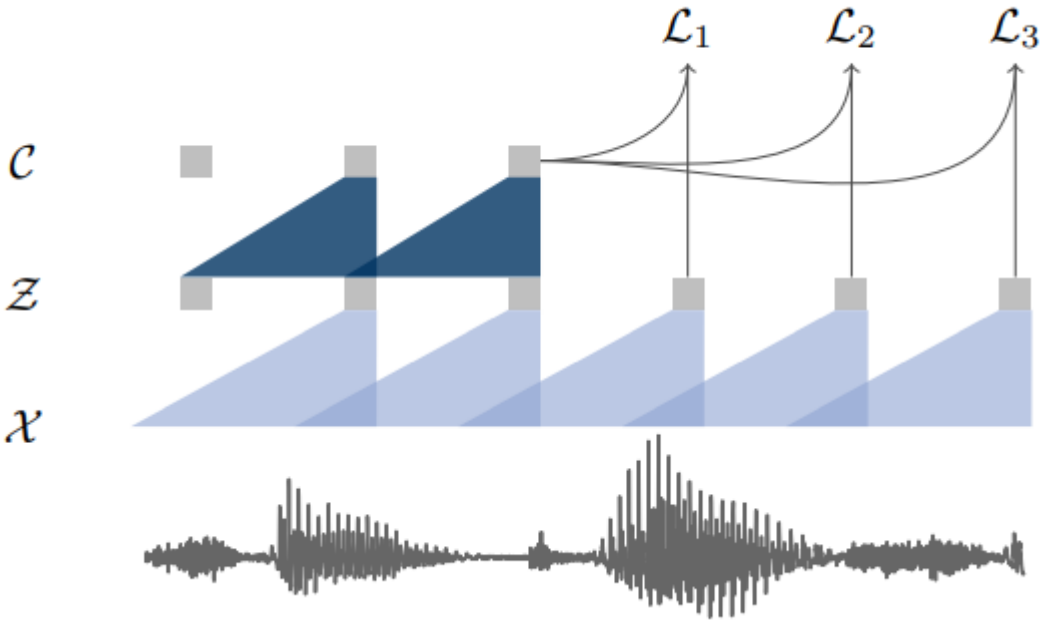
## 2.1 Wav2vec và Wav2vec2

### 2.1.1 Wav2vec

Các mô hình hiện đại để nhận diện giọng nói thường yêu cầu một lượng lớn dữ liệu âm thanh để có thể đạt hiệu suất tốt, và học sâu (Deep Learning) đã nổi lên như một giải pháp hữu ích cho bài toán nhận diện giọng nói mà dữ liệu được gán nhãn khan hiếm.

Dựa trên ý tưởng đó, nhóm nghiên cứu của Facebook AI đã nghiên cứu và phát triển framework wav2vec [5] dựa trên học sâu cho nhiệm vụ nhận diện giọng nói vào năm 2019. Theo nhóm tác giả, họ đã huấn luyện trước (pre-train) một mạng thần kinh tích chập nhiều lớp (multi-layer convolutional neural network) đơn giản. Mô hình này được nhóm tác giả thống kê rằng nó đạt tỷ lệ lỗi từ (WER - word error rate) 2,43% trên tập dữ liệu nov92 và là tốt nhất đến thời điểm phương pháp này được ra mắt.

Về mô hình, mô hình wav2vec phiên bản gốc lấy tín hiệu âm thanh thô làm đầu vào và sau đó áp dụng 2 mạng: bộ mã hoá và mạng ngữ cảnh.




**Bộ mã hoá** Đầu tiên, với mẫu âm thanh thô $x_i \in X$, áp dụng bộ mã hoá $f : X \to Z$ được tham số hoá dưới dạng tích chập 5 lớp. Đầu ra của bộ mã hoá là một biểu diễn đặc trưng tần số thấp (low frequency feature representation) $z_i \in Z$.

**Mạng ngữ cảnh** Sau khi đi qua mạng mã hoá, áp dụng mạng ngữ cảnh $g : Z \to C$ cho đầu ra của bộ mã hoá để trộn các biểu diễn ẩn $z_i \dots z_{i-v}$ thành một tensor ngữ cảnh duy nhất $c_i = g(z_i \dots z_{i-v})$ cho kích thước tiếp nhận (receptive field size) $v$.

Điều quan trọng đối với mô hình này là phải chọn biểu diễn chuẩn hoá bất biến đối với tỉ lệ và offset của đầu vào. Việc này giúp cho quá trình biểu diễn tổng quát tốt hơn.

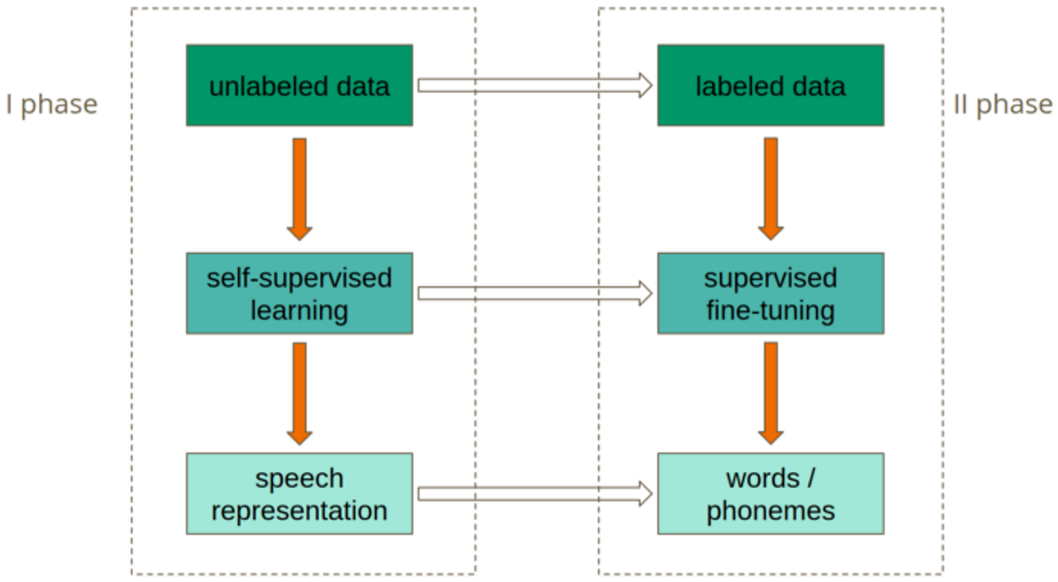
### 2.1.2 Wav2vec2

Wav2vec 2.0 (hay wav2vec2) [6], là phiên bản kế thừa của wav2vec. Mô hình mới này học các đơn vị giọng nói cơ bản được sử dụng để giải quyết một nhiệm vụ tự giám sát. Mô hình được đào tạo để dự đoán đơn vị giọng nói chính xác cho các phần bị che khuất của âm thanh, đồng thời tìm hiểu đơn vị giọng nói (speech unit) nên là gì.

Chỉ với 10 phút giọng nói được phiên âm và 53.000 giờ giọng nói không được gắn nhãn, wav2vec 2.0 cho phép các mô hình nhận dạng giọng nói với tỷ lệ lỗi từ (WER) là 8,6% đối với giọng nói có tiếng ồn và 5,2% đối với giọng nói rõ ràng trên bộ dữ liệu LibriSpeech tiêu chuẩn.

Wav2vec2 cũng được huấn luyện đa ngôn ngữ, và phương pháp của wav2vec2 đã mở ra cơ hội cho các mô hình nhận dạng giọng nói bằng nhiều ngôn ngữ, phương ngữ và miền hơn mà trước đây yêu cầu nhiều dữ liệu âm thanh được sao chép hơn để mang lại độ chính xác có thể chấp nhận được.

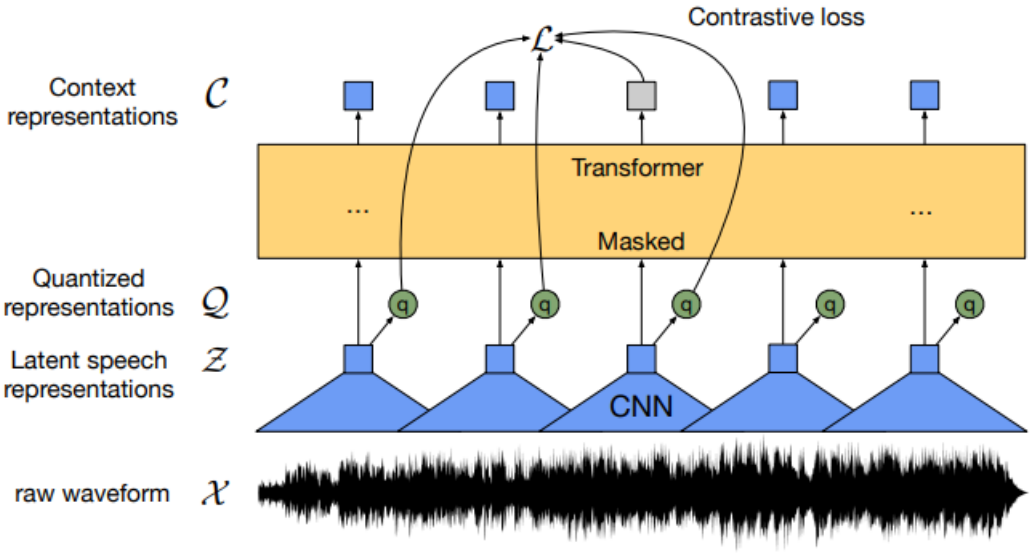
**Ý tưởng chính** Mô hình được huấn luyện qua 2 giai đoạn. Giai đoạn đầu tiên là tự giám sát (self-supervised), được thực hiện bằng cách sử dụng dữ liệu chưa được gắn nhãn (unlabeled data) và nhằm mục đích đạt được khả năng biểu diễn giọng nói tốt nhất có thể. Quá trình này khá tương đồng với quá trình nhúng từ (word embedding). Việc nhúng từ cũng nhằm mục đích đạt được sự thể hiện tốt nhất của ngôn ngữ tự nhiên. Sự khác biệt chính là Wav2Vec 2.0 xử lý âm thanh thay vì văn bản. Giai đoạn đào tạo thứ hai là tinh chỉnh có giám sát (supervised fine-tuning), trong đó dữ liệu được gắn nhãn (labeled data) được sử dụng để huấn luyện mô hình dự đoán các từ hoặc âm vị cụ thể. Âm vị ở đây có thể hiểu là đơn vị âm thanh nhỏ nhất có thể có trong một ngôn ngữ cụ thể, thường được biểu thị bằng một hoặc hai chữ cái. Hình dưới minh hoạ cho 2 giai đoạn của quá trình huấn luyện cũng như tinh chỉnh wav2vec2.




Về mặt kiến trúc, wav2vec2 gồm có 4 phần chính:

- Bộ mã hoá đặc trưng xử lý đầu vào dạng sóng thô $W$ để có được biểu diễn tiềm ẩn (latent representation) - $Z$
- Biểu diễn định lượng $Q$

- Các lớp transformers, dùng để tạo biểu diễn theo ngữ cảnh - $C$
- Phép chiếu tuyến tính thành đầu ra - Y.



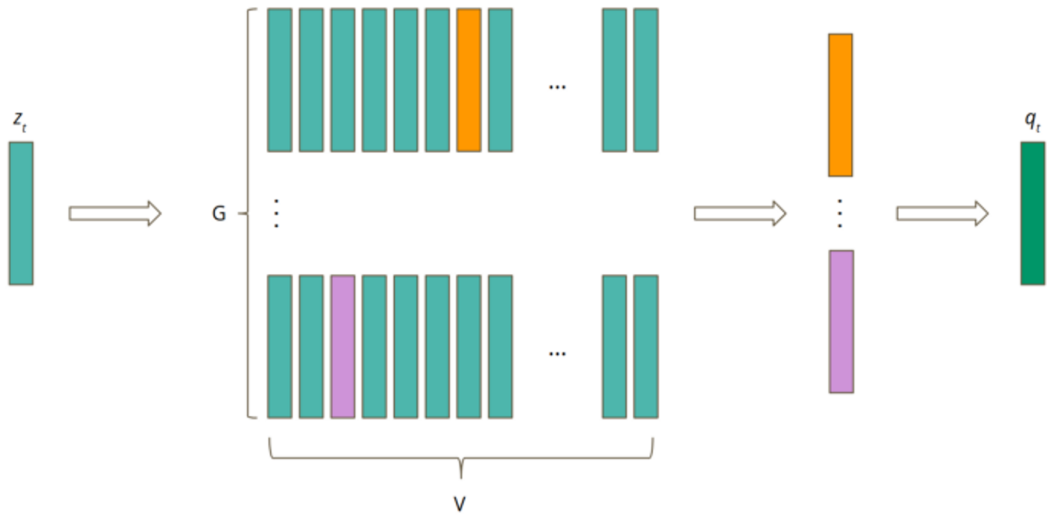

**Bộ mã hoá đặc trưng** Bộ mã hoá gồm một số lớp tích chập với hàm kích hoạt GELU nhằm mã hoá đầu vào dạng sóng thô $W$. Đầu vào này sau khi được đưa vào bộ mã hoá thì được chuẩn hoá về giá trị trung bình bằng 0 và phương sai đơn vị. Về mặt biểu diễn toán học, thì bộ mã hoá là một ánh xạ $f : X \to Z$ biến đầu vào dạng sóng thô $W$ thành các biểu diễn ẩn đầu ra $z_1, ..., z_T$ cho $T$ bước thời gian.

**Biểu diễn ngữ cảnh với Transformers** Đầu ra của bộ mã hoá được đưa vào mạng ngữ cảnh dựa trên kiến trúc Transformer. Ở đây thay vì nhúng vị trí tuyệt đối (fixed positional embeddings) thì các tác giả đã sử dụng một lớp tích chập hoạt động như các nhúng vị trí tương đối (relative positional embeddings). Sau khi thu được đầu ra từ lớp tích chập này thì sử dụng hàm kích hoạt GELU và chuẩn hoá lớp để đưa về các biểu diễn ngữ cảnh. Về mặt biểu diễn toán học thì mạng biểu diễn ngữ cảnh là một ánh xạ $g : Z \to C$ để xây dựng thành các biểu diễn ngữ cảnh đầu ra $c_1, ..., c_T$ mà có thể ghi lại được các thông tin từ chuỗi đã cho (dữ liệu âm thanh $X$).

**Học tương phản trong mạng Transformer biểu diễn ngữ cảnh** Học tương phản là một khái niệm trong đó đầu vào được biến đổi theo hai cách khác nhau. Sau đó, mô hình được đào tạo để nhận ra liệu hai phép biến đổi của đầu vào có còn là cùng một đối tượng hay không. Trong wav2Vec2, các lớp transformers là cách chuyển đổi đầu tiên, cách thứ hai được thực hiện bằng cách lượng tử hóa (quantization) nhằm giúp mô hình hiểu rõ được âm thành đầu vào.

**Module lượng tử hoá** $Q$ Lượng tử hóa là một quá trình chuyển đổi các giá trị từ một không gian liên tục thành một tập hữu hạn các giá trị trong một không gian rời rạc. Đối với huấn luyện tự giám sát, ta rời rạc hoá đầu ra của bộ mã hoá đặc trưng $z$ thành một tập hữu hạn các điểm biểu diễn giọng nói thông qua quá trình lượng tử hoá. Ngoài ra, có một nhận xét là số lượng tất cả các cặp âm vị có thể là hữu hạn. Điều đó có nghĩa

là chúng có thể được biểu diễn một cách hoàn hảo bởi cùng một biểu diễn lời nói tiềm ẩn. Hơn nữa, ta có thể tạo một cuốn sách mã (codebook) chứa tất cả các cặp âm vị có thể có. Sau đó, lượng tử hóa đi xuống để chọn đúng từ mã từ sách mã. Tuy nhiên, bạn có thể tưởng tượng rằng số lượng của tất cả các âm thanh có thể có là rất lớn. Để đào tạo và sử dụng dễ dàng hơn, các tác giả của Wav2Vec2 đã tạo các sách mã, mỗi sách bao gồm $V$ từ mã. Để tạo một biểu diễn lượng tử hóa, nên chọn từ tốt nhất từ mọi sách mã. Sau đó, các vectơ đã chọn được nối và xử lý bằng phép biến đổi tuyến tính để thu được biểu diễn lượng tử hóa.




**Hàm Softmax Gumbel** Để chọn các sách mã rời rạc thì các tác giả của wav2vec2 đã sử dụng công cụ ước tính xuyên suốt (straight-through estimator) và thiết lập $G$ thao tác Gumbel softmax, trong đó $G$ là số sách mã. $V$ là số từ mã của mỗi sách mã. Khi đó xác suất để chọn sách mã thứ $v$ cho nhóm $g$ là:

$$p_{g,v} = \frac{exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} exp(l_{g,v} + n_v)/\tau}$$

Trong đó, $l$ là logit ảnh của ánh xạ từ đầu ra của bộ mã hoá $z$, $\tau$ là một tham số nhiệt độ không âm, $u$ là các mẫu đồng nhất từ phân phối xác suất $\mathcal{U}(0, 1)$ và $n = -log(-log(u))$. Khi đó, sách mã $i$ được chọn theo công thức $i = \text{argmax}_j p_{g,j}$.

**Hàm mục tiêu** Trong quá trình huấn luyện thì có thể thấy mô hình wav2vec2 giải quyết bài toán với 2 hàm loss chính: hàm loss tương phản (contrastive loss) để đánh giá quá trình lượng tử hoá và hàm loss về tính đa dạng (diversity loss) để đánh giá mức độ sử dụng các từ mã từ các sách mã. Cụ thể thì hàm mục tiêu của mô hình sẽ có dạng:

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d$$

Trong đó, $\alpha$ là một tham số điều chỉnh được.

**Hàm loss tương phản** Cho đầu ra của mạng ngữ cảnh $c_t$ ở bước thời gian $t$, mô hình cần xác định biểu diễn tiềm ẩn được lượng tử hoá $q_t$ trong 1 tập $K+1$ các biểu diễn ứng viên $\tilde{q} \in Q_t$. Hàm loss được định nghĩa theo công thức sau:

$$\mathcal{L}_m = -log\frac{exp(sim(c_t, q_t)/\kappa)}{\sum_{\tilde{q}\in Q_t} exp(sim(c_t, q_t)/\kappa)}$$

với sim là độ tương đồng bất kỳ, như các tác giả có gợi ý thì ta có thể sử dụng độ tương đồng cosine.

**Hàm loss về tính đa dạng** Để khuyến khích việc sử dụng một cách cân bằng $V$ từ mã từ $G$ sách mã, hàm loss tính đa dạng sẽ cực đại hoá entropy (maximum entropy) của phân phối softmax trung bình $l$ trên các từ mã sao cho mỗi từ mã $\bar{p}_g$ trong mỗi batch. Cụ thể thì hàm loss này được định nghĩa bởi công thức:

$$\mathcal{L}_d = \frac{1}{GV}\sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV}\sum_{g=1}^{G}\sum_{v=1}^{V} \bar{p}_{g,v} log(\bar{p}_{g,v})$$

## 2.2 CTC

### 2.2.1 Giới thiệu

Ở trong nhận diện giọng nói, chúng ta có một tập dữ liệu dạng âm thanh (audio) và một bảng các transcript tương ứng, nhưng chúng ta không biết được ký tự ở trong transcript sẽ tương ứng với đoạn nào trong 1 audio và điều này tạo ra khó khăn trong việc huấn luyện.

Để giải quyết vấn đề trên chúng ta có thể đưa vào một số nguyên tắc cơ bản ví dụ như "1 câu tương ứng với mười đầu vào", nhưng tốc độ nói cũng như ngữ điệu của mỗi người là khác nhau nên những quy tắc này đều không được chính xác. Một cách khác là chúng ta có thể căn chỉnh từng ký tự vào từng vị trí tương ứng của nó trong đoạn âm thanh, nhưng mà phương pháp này sẽ tốn nhiều thời gian với tập dữ liệu lớn.

Connectionist Temporal Classification (CTC) [1] là phương pháp giúp thể hiện sự phụ thuộc của input và output.

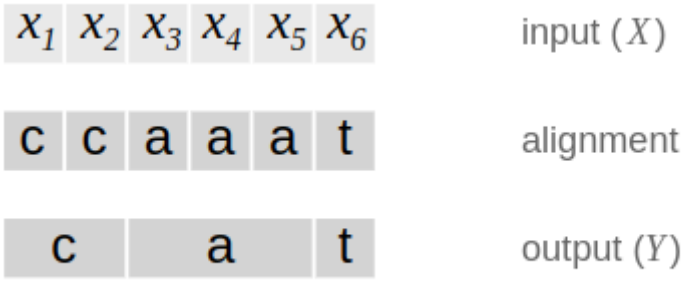
Cụ thể: chúng ta cần ánh xạ đầu vào $X = [x_1, x_2, ..x_T]$ đoạn audio tương ứng sang đầu ra $Y = [y_1, y_2, .., y_U]$ transcript tương ứng.

- Với độ dài của X, và Y có thể thay đổi
- Tỷ lệ giữa độ dài của X và Y có thể thay đổi
- không biết bất kì liên kết chính xác nào giữa tập X và Y

### 2.2.2 Thuật toán

Thuật toán ctc có thể tính được xác suất của mọi tập Y nhận được từ một tập X. Chúng ta sẽ nhìn vào các alignment và cách tính toán hàm loss của thuật toán này.

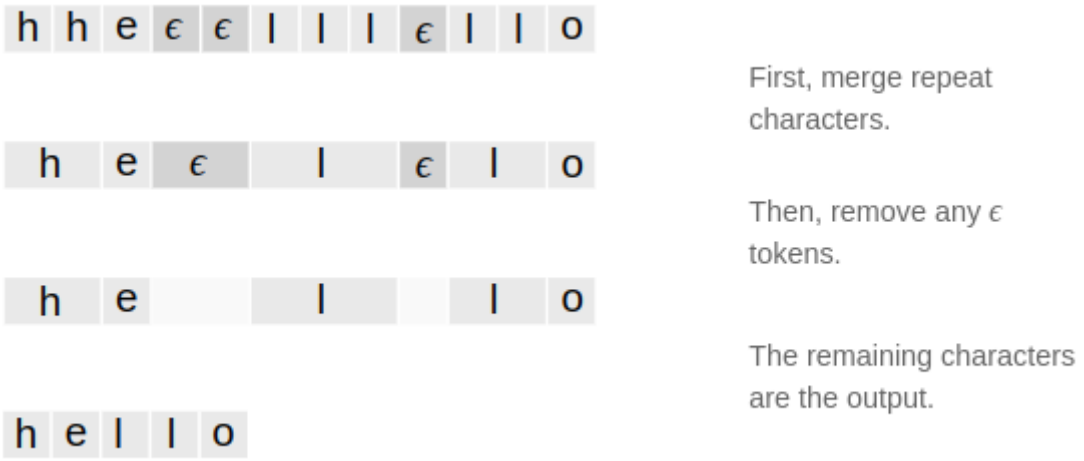
**Alignment** Cách tiếp cận đơn giản giả sử tập X có độ dài là 6 và tập Y = [c, a, t]. Một cách đơn giản để sắp xếp từ X sang Y là gộp tất cả những đoạn gồm các kí tự giống nhau thành 1 kí tự:



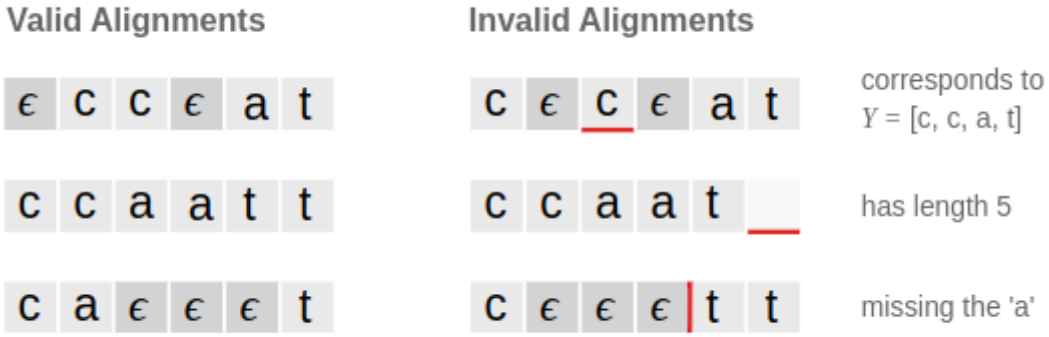

Dễ thấy cách làm trên có một số vấn đề như sau

- Trong nhận diện giọng nói có thể có những khoảng trống (người dùng không nói gì) kéo dài
- Có những từ gồm những ký tự liên tiếp giống như ví dụ như "hello"

Cách để giải quyết vấn này là thêm những token trống vào các output hợp lệ gọi là $\varepsilon$, $\varepsilon$ không tương ứng với bất kì kí tự nào và sẽ được xóa bỏ khỏi output



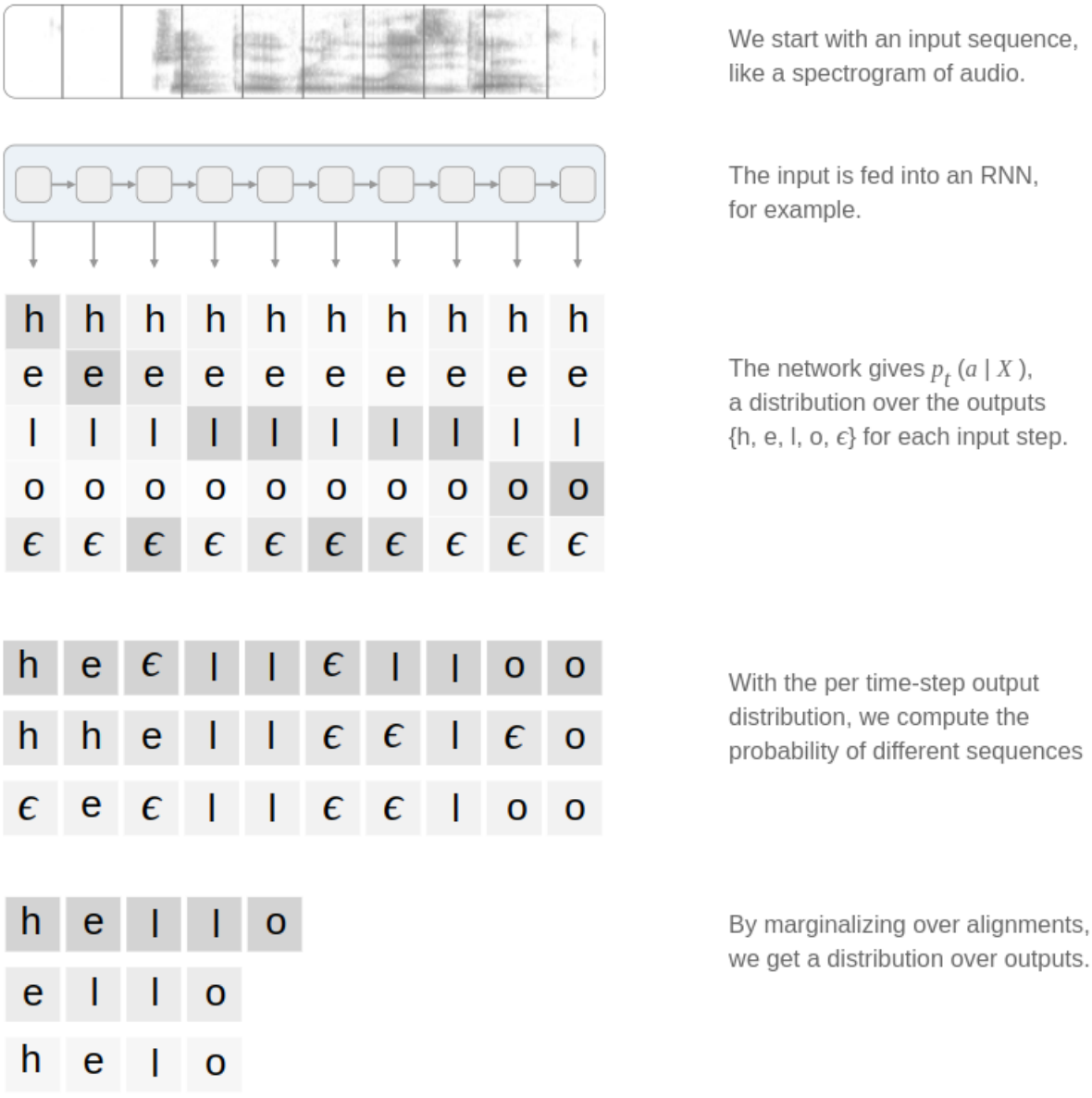

Nếu Y có 2 ký tự giống nhau và ở cạnh nhau thì nó phải có kí tự $\varepsilon$ ở giữa. Ví dụ:



Các đặc tính của CTC alignments

- Quan hệ giữa X và Y là n:1
- Trình tự sắp xếp hợp lệ của X và Y là duy nhất

**Loss Function** CTC alignments cho chúng ta một cách để tính từ xác suất tại mỗi time-step đến xác suất của mỗi output




Cách tính Loss Function của hàm Y, ta chèn ký tự $\varepsilon$ và giữa mới phần tử của Y ta được hàm Z như sau

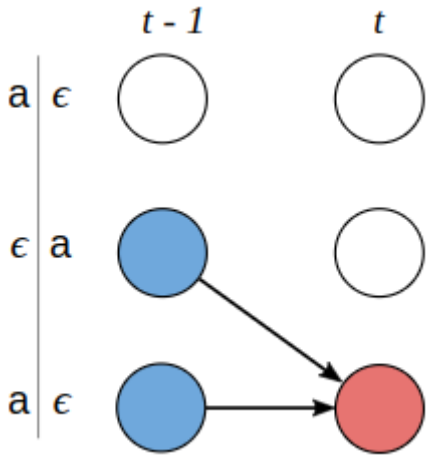
$$Z = [\varepsilon, y_1, \varepsilon, y_2, \varepsilon, .., \varepsilon, y_U, \varepsilon] \tag{1}$$

Ta có $P[s,t]$ là tỷ lệ của $Z_s$ nằm trong input tại bước thứ $t$ Gọi $F[s,t]$ là điểm (tỷ lệ) CTC của xâu con $Z[1:s]$ và tại bước $t$, ta sẽ có các tính mảng $F$ như sau:

TH1: $Z[s] = \varepsilon$

Lúc này mảng mới sẽ được cập nhật từ 2 cách: một là từ chính ký tự $\varepsilon$ ở bước t-1, còn lại là kí tự $Z[s-1]$ vì $Z[s-1] \neq \varepsilon$

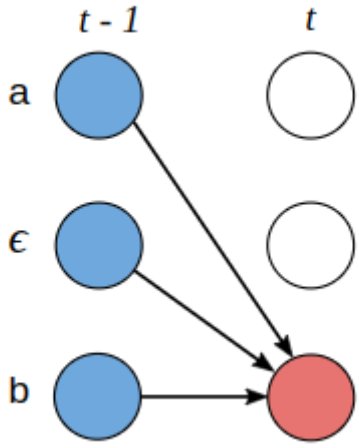
$$F[s,t] = (F[s-1,t-1] + F[s,t-1]) * P[s,t] \tag{2}$$



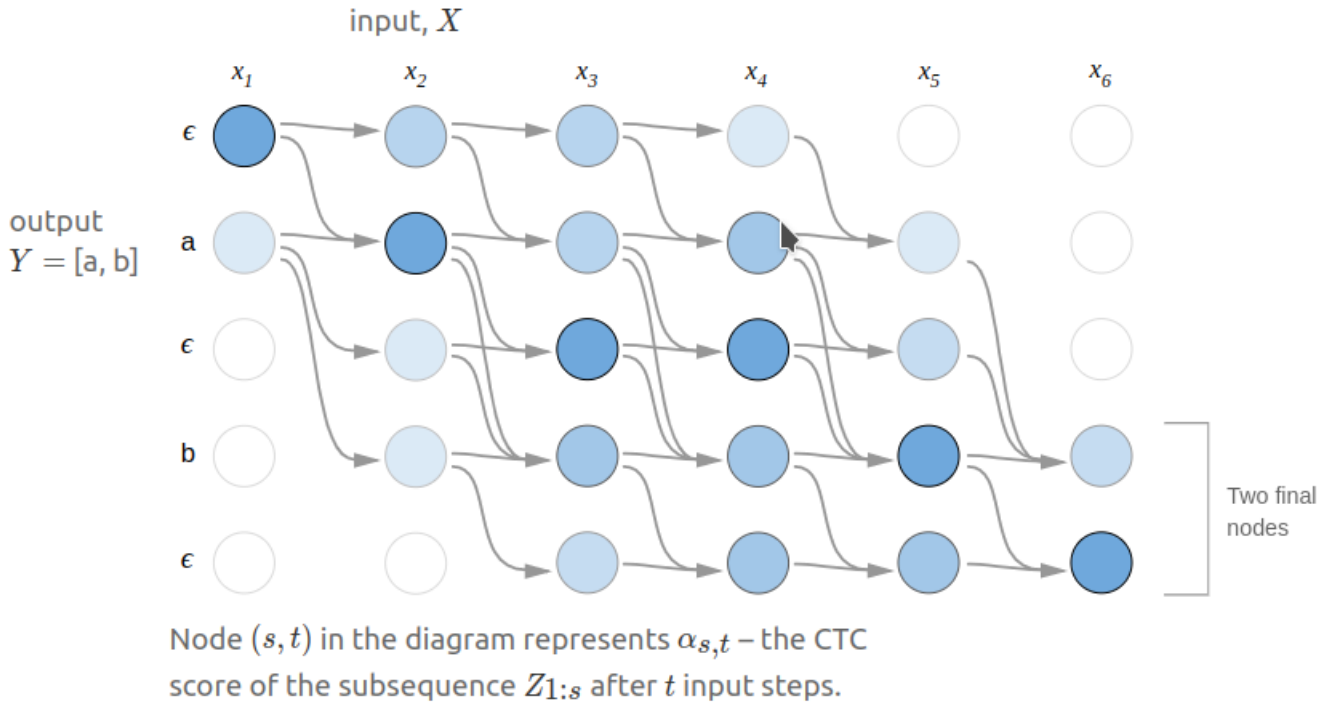

TH2: $Z[s] \neq \varepsilon$

Lúc này phần tử $\varepsilon$ sẽ nằm giữa 2 phần tử riêng biệt nên kết quả sẽ có thể tính từ 3 vị trí của bước t-1

$$F[s,t] = (F[s-2,t-1] + F[s-1,t-1] + F[s,t-1]) * P[s,t] \tag{3}$$




Ví dụ về việc tính toán bằng phương pháp quy hoạch động, mọi alignment hợp lệ đều được biểu diễn trong hình dưới đây




Sau khi có cách tính LossFunction hiệu quả chúng ta cần tính độ dốc(gradient) và huấn luyện mô hình. CTC LossFunction có thể tính toán được tại mỗi time-step vì chỉ gồm các phép tính nhân và cộng điều này khiến cho chúng ta có thể tính toán được độ dốc (gradient) của LossFuntion bằng cách lan truyền ngược

Mỗi tập huấn luyện D. các tham số của model sẽ được điều chỉnh để cực tiểu hóa negative log-likelihood

$$\sum_{(X,Y)\in D} -logP(Y|X) \tag{4}$$

## 2.3 N-gram language model và các kỹ thuật smoothing

### 2.3.1 N-gram language model

Ngôn ngữ tự nhiên là những ngôn ngữ được con người sử dụng trong các giao tiếp hàng ngày: nghe, nói đọc, viết. Mặc dù con người có thể dễ dàng hiểu được và học các ngôn ngữ tự nhiên nhưng việc làm cho máy hiểu được ngôn ngữ tự nhiên không phải là chuyện dễ dàng. Sở dĩ có khó khăn là do ngôn ngữ tự nhiên có các bộ luật, cấu trúc ngữ pháp phong phú hơn nhiều các ngôn ngữ máy tính, hơn nữa để hiểu đúng nội dung các giao tiếp, văn bản trong ngôn ngữ tự nhiên cần phải nắm được ngữ cảnh của nội dung đó. Và như trong [7], mô hình có thể gán xác suất cho một chuỗi các từ được gọi là mô hình ngôn ngữ. Ở đây, chúng tôi sử dụng một mô hình ngôn ngữ thống kê đơn giản là mô hình n-gram.

Đầu tiên, chúng ta cần tính P(w|h) xác suất của một từ khi biết ngữ cảnh ở trước. Ví dụ P("chó"| "tôi có nuôi một con") sẽ được kỳ vọng sẽ lớn hơn P("máy"|"tôi có nuôi một con"). Và ở đây, P(w|h) sẽ được tính từ tần suất xuất hiện của ngữ cảnh và từ đích và công thức tính là:

$$P(w|h) = C(w, h)/C(h) \tag{5}$$

Ở đây C(w, h) là tần suất w xuất hiện sau h, C(h) là tần suất xuất hiệ của h. Với tập dữ liệu lớn, chúng ta có thể tính các tấn suất xuất hiện đó là ước lượng các xác suất một cách tương đối chính xác. Tiếp theo, ta cần tính xác suất của một chuỗi các từ $P(w_1, w_2, ..., w_n)$. Ở đây, ta sẽ sử dụng tính chất chain-rule trong xác suất thông kê và từ đó ta có công thức:

$$\begin{aligned} P(w_1, w_2, ..., w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(wn|w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_{1:k-1}). \end{aligned} \tag{6}$$

Tuy nhiên, khi ước lượng xác suất của một câu dài, ta cần tính tấn suất của một câu dài và việc này tốn rất nhiều thời gian và bộ nhớ có thể tăng theo hàm số mũ. Và ở đây, thay vì tính xác suất của một từ xuất hiện sau toàn bộ ngữ cảnh trước đó ta chỉ tính xác suất của một từ xuất hiện sau ngữ cảnh là một vài từ trước đó. Vì dụ mô hình bigram, ta chỉ ước lượng xác suất $P(w_n|w_{1:n-1})$ bằng chính $P(w_n|w_{n-1})$ bằng cách giả sử hai xác suất này bằng nhau. Giả thiết này được gọi là giả thiết Markov. Và chúng ta có thể tổng quát hóa mô hình bigram thành mô hình trigram hoặc mô hình N-gram và công thức là

$$P(w_k|w_{1:k-1}) \approx P(w_k|w_{k-N+1:k-1}) \tag{7}$$

### 2.3.2 Smoothing

Đôi khi, trong tập train và tập test, phân bố của các từ khác nhau, vì vậy đôi khi các xác suất có thể ra 0 trong tập test dẫn đến việc giảm hiệu suất của mô hình. Vì vậy, các phương pháp làm mịn được đề xuất để khắc phục vấn đề xác suất bằng 0.

**Laplace smoothing (Add one smoothing)** Phương pháp này sẽ công thêm vào số lần xuất hiện của mỗi cụm n-gram lên một đơn vị. Và xác suất của cụm n-gram được tính lại ví dụ trong mô hình bigram bằng công thức:

$$P_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V} \tag{8}$$

Với V ở đây là kích thước của từ vựng hay số lượng từ phân biệt xuất hiện trong dữ liệu.

**Add-k smoothing** Đây là một phiên bản thay thế của phương pháp laplace smoothing hay add one smoothing. Thay vì cộng một đơn vị vào tần suất xuất hiện của các n-gram phương pháp này cộng k đơn vị vào tần suất và tính lại xác suất mới bằng công thức

$$P_{Add-k}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1}) + kV} \tag{9}$$

Và k ở đây thường được lựa chọn bằng cách tối ưu với tập validation.

**Backoff** Những phương pháp trước, sẽ làm mịn một cách công bằng với các từ không xuất hiện cùng ngữ cảnh. Tuy nhiên, chúng ta có thể làm tốt hơn bằng cách tính xác suất của từ xuất hiện sau một ngữ cảnh dài bằng xác suất của từ đó xuất hiện trong một ngữ cảnh ngắn hơn. Ví dụ chúng ta có thể tính $P(w_n|w_{n-2}w_{n-1})$ từ xác suất của $P(w_n|w_{n-1})$ nếu $w_n$ nếu không xuất hiện cùng ngữ cảnh $w_{n-2}w_{n-1}$ và nếu $w_n$ cũng không xuất hiện cùng $w_{n-1}$ thì có thể dùng $P(w_n)$ bằng công thức:

$$P(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n|w_{n-2}w_{n-1}) \tag{10}$$

Tổng của các trọng số $\lambda$ phải bằng 1:

$$\sum_i \lambda_i = 1 \tag{11}$$

Tuy nhiên, trong thực tế rất khó để chọn được trọng số một cách hợp lý vì vậy trọng số thường được tính một cách phụ thuộc vào ngữ cảnh.

**Kneser-Ney Smoothing** Cuối cùng, một trong những kỹ thuật làm mịn được sử dụng nhiều và cho ra hiệu suất tốt là thuật toán nội suy Kneser-Ney. Kneser-Ney được coi là phương pháp hiệu quả nhất để làm mịn do việc sử dụng các chiết khấu tuyệt đối bằng cách trừ đi một giá trị cố định từ điều kiện về tần số bậc thấp của xác suất để bỏ qua n-gram với tần số thấp hơn. Đây là phương pháp hiệu quả với bigram và các mô hình n-gram cao hơn. Đầu tiên, công thức để tính xác suất bigram khi sử dụng Kneser-Ney smoothing là:

$$P_{KN}(w_i|w_{i-1}) = \frac{max(c(w_{i-1}, w_i) - \delta, 0)}{\sum_{w'} c(w_{i-1}, w')} + \lambda_{w_{i-1}} P_{KN}(w_i) \tag{12}$$

$$P_{KN}(w_i) = \frac{|\{w' : c(w', w_i) > 0\}|}{|\{(w', w'') : c(w', w'') > 0\}|} \tag{13}$$

Tham số $\delta$ là hằng số được chọn trước, thường ở giữa 0 và 1.

$$\lambda_{w_{i-1}} = \frac{\delta}{\sum_{w'} c(w_{i-1}, w')} |w' : c(w_{i-1}, w') > 0| \tag{14}$$

Công thức tổng quát cho n-gram là:

$$P_{KN}(w_i|w_{i-n+!}^{i-1}) = \frac{max(c(w_{i-n+1}^{i-1}, w_i) - \delta, 0)}{\sum_{w'} c(w_{i-n+1}^{i-1}, w')} + \delta \frac{|\{w' : c(w_{i-n+1}^{i-1}, w')\}|}{\sum_{w'} c(w_{i-n+1}^{i-1}, w')} P_{KN}(w_i|w_{i-n+1}^{i-1}) \tag{15}$$

# 3 Các thư viện sử dụng

## 3.1 Transformers

Transformers [4] là một thư viện được phát triển bởi huggingface. Transformers cung cấp các API và các công cụ có thể tải và huấn luyện tiếp các mô hình pretrained state of the art. Sử dụng mô hình pretrained có thể giảm chi phí tính toán, tiết kiệm thời gian và tài nguyên cho bạn thay vì huấn luyện lại mô hình từ đầu. Thư viện này hỗ trợ rất nhiều bài toán: xử lý ngôn ngữ tự nhiên, thị giác máy tính, các bài toán về âm thanh và cả multimodal một bài toán khá mới. Transformers cũng hộ trợ rất nhiều framework Pytorch, Tensorflow và JAX. Cụ thể, trong dự án này, nhóm chúng tôi sử dụng thư viện Transformers để triển khai mô hình Wav2Vec2. Khi xây dựng mô hình nhận diện giọng nói sử dụng mô hình Wav2Vec với thư viện Transformers, ta cần sử dụng các lớp Wav2Vec2CTCTokenizer, Wav2Vec2FeatureExtractor, Wav2Vec2ForCTC. Sau đây, chúng tôi sẽ đi vào chi tiết các lớp.

### 3.1.1 Wav2Vec2CTCTokenizer

Wav2Vec2CTCTokenizer là một lớp kế thừa từ lớp PreTrainedTokenizer, được sử dụng để tạo Wav2Vec2CTC tokenizer. Tokenizer này được dùng để decode các phân bố xác suất của các âm tiết của từng khoảng thời gian từ đầu ra của lớp Wav2VecForCTC. Tokenizer này thực chất thực hiện thuật toán tham lam lựa chọn kết quả có xác suất cao nhất ở mỗi bước để đưa ra kết quả cuối cùng.

### 3.1.2 Wav2Vec2FeatureExtractor

Wav2Vec2FeatureExtractor là lớp được sử dùng để tạo Wav2Vec2 feature extractor. Nó được sử dụng để chắc chắn rằng âm thanh đầu vào của bạn trùng khớp với trong cấu hình của mô hình được fine tune hoặc mô hinh pretrain.

### 3.1.3 Wav2Vec2ForCTC

Wav2Vec2ForCTC là một lớp triển khai mô hình Wav2Vec2Model gốc với một mô hình ngôn ngữ ở trên mô hình Wav2Vec2 gốc cho thuật toán Connectionist Temporal Classification (CTC). Nó được sử dụng để nhận đầu vào từ Wav2Vec2FeatureExtractor và cho ra phân bố xác suất của các âm tiết của từng khoảng thời gian và làm đầu vào cho Wav2Vec2CTCTokenizer.

## 3.2 kenlm

Kenlm là một bộ công cụ dùng để xây dựng và sử dụng mô hình ngôn ngữ n-gram một cách hiệu quả. Việc xây dựng mô hình ngôn ngữ sử dụng thuật toán được trình bày ở [3]. Đồng thời, họ sử dụng thuật toán Kneser-ney smoothing được cải tiến bao gồm cả nội suy để tăng hiệu suất của mô hình.Hơn thế, streaming và sorting cho phép thuật toán scale tới các mô hình lớn hơn, cụ thể với một máy tính với 140GB RAM với 2.8 ngày, họ xây dựng được mô hình ngôn ngữ trên bộ dữ liệu 126 tỉ từ. Và nó được cung cấp ở dạng mã nguồn mở như một phần của bộ công cụ Kenlm. Đồng thời, truy vấn tời các mô hình ngôn ngữ cũng được tối ưu về cả thời gian và bộ nhớ sử dụng. Họ xây dựng hai cấu trúc dữ liệu để truy vấn mô hình ngôn ngữ một cách hiệu quả (PROBING, TRIE). Các thuật toán được trình bày ở [2].

## 3.3 pyctcdecode

Pyctcdecode là một thư viện Python triển khai phương pháp decode nhanh và nhiều chức năng cho thuật toán CTC và thực hiện beam search để nâng cao hiệu suất dành cho bài toán nhận diện giọng nói. Thư viện này hỗ trợ mô hình ngôn ngữ n-gram được xây dựng bởi bộ công cụ kenlm được trình bày ở trên. Khi sử dụng, người dùng chỉ cần cung cấp đường dẫn của mô hình ngôn ngữ được export ra và từ điển các chữ cái cho bài toán để xây dựng được 1 decoder và sau đó sử dụng decoder đó để thực hiện beam search và cho ra kết quả.

# 4 Sinh bộ dữ liệu lệnh cờ tướng

Theo như các thực nghiệm trước, việc sử dụng thêm mô hình ngôn ngữ cho đầu ra của mô hình Wav2Vec2ForCTC có thể tăng rất nhiều hiệu suất cho mô hình. Như trong hình 4.1, ta có thể thấy việc sử dụng mô hình ngôn ngữ sẽ tăng hiệu quả mô hình lên rất nhiều.

| | VIVOS | COMMON VOICE VI | VLSP-T1 | VLSP-T2 |
|---|---|---|---|---|
| without LM | 10.77 | 18.34 | 13.33 | 51.45 |
| with 4-grams LM | 6.15 | 11.52 | 9.11 | 40.81 |

Hình 4.1: Lợi ích của việc dùng mô hình ngôn ngữ

Tuy nhiên khi sử dụng mô hình ngôn ngữ được cung cấp bởi chính tác giả và mô hình ngôn ngữ n-gram được xây dựng lại từ bộ dữ liệu văn bản được lấy từ cái bài báo lấy từ repository https://github.com/binhvq/news-corpus, mô hình nhận diện giọng nói vẫn thường đưa ra các lệnh sai. Việc này cũng là điều dễ hiểu, điều đó chứng tỏ phân bố xác suất của các từ của các bộ dữ liệu đó lệch nhiều với phân bố xác suất của các từ của các lệnh cờ tướng. Tuy nhiên, theo nhóm chúng tôi được biết hiện tại chưa có bộ dữ liệu dạng chữ nào cho lệnh cờ tướng. Vì vậy, nhóm chúng tôi xây dựng chuẩn cờ tướng cho bài toán và sinh bộ dữ liệu dạng chữ theo chuẩn đó để có thể xây dựng mô hình ngôn ngữ cho bài toán này. Ở đây, nhóm chúng tôi sinh ra ba loại lệnh trong cờ tướng sau khi thống nhất bao gồm: lệnh di chuyển, lệnh ăn (bắt), lệnh chiếu tướng. Với mỗi loại lệnh, nhóm chúng tôi sử dụng thuật toán quay lui để sinh ra các lệnh cờ tướng theo chuẩn của nhóm chúng tôi thông nhất (có thể chứa các lệnh không hợp lệ nhưng sẽ chứa một phần rất bé). Qua thực nghiệm, chúng tôi thấy kết quả được cải thiện rất nhiều và nhận diện được hầu hết các lệnh khi nhóm chúng tôi sử dụng trên sản phẩm trò chơi cờ tướng.

# 5 Kết quả đạt được

## 5.1 Chuẩn cờ tướng

Trước khi tiến hành cài đặt sản phẩm thì nhóm chúng tôi kết hợp với nhóm VR đã cùng nhau xây dựng một chuẩn cờ tướng chung với các tiêu chí:

- Tương thích với chuẩn cờ tướng của Liên đoàn cờ tướng Việt Nam.
- Tập lệnh gần với logic tự nhiên, dễ dàng điều khiển bằng giọng nói.
- Có thể mở rộng và dễ cài đặt.

Ngoài ra nhóm chúng tôi đã hoàn thành cơ bản việc xây dựng một chuẩn cờ tướng chung, có thể dễ dàng cài đặt cho trò chơi cờ tướng mà hai nhóm cùng làm thêm vào đó chuẩn cờ tướng này cũng có thể áp dụng được cho các game cờ tướng khác. Chuẩn được đăng tại địa chỉ **uet-cacvdhd.github.io/xiangqi-docs**.

Việc xây dựng chuẩn cờ tướng là nền tảng bước đầu để chúng tôi có thể triển khai và hoàn thiện ứng dụng cờ tướng này.

## 5.2 Một số thống kê và kết quả mô hình

Để đánh giá độ chính xác của mô hình nhận diện giọng nói cho game cờ tướng, nhóm chúng tôi có chạy so sánh mô hình wav2vec2 gốc (BASE) và mô hình wav2vec2 gốc kết hợp với mô hình ngôn ngữ được xây trên bộ dữ liệu cờ tướng và beam search (OUR). Để được những đoạn âm thanh để đánh giá nhóm chúng tôi sử dụng dịch vụ text-to-speech của fpt.ai và đồng thời để tăng độ khó cho mô hình chúng tôi sử dụng thêm phương ngữ miền Trung và miễn Nam, những đoạn văn bản dùng để sinh sẽ được sinh ngẫu nhiên 1000 lệnh cờ tướng theo chuẩn của nhóm đề ra. Và kết quả như trong bảng 1, mô hình OUR dự đoán đúng vượt trội so với mô hình BASE gốc.

| Model | Accuracy |
|---|---|
| BASE | 7.5% |
| OUR | 78.8% |

Bảng 1: Hiệu suất của các mô hình thử nghiệm

# 6 Tổng kết

Với chuẩn cờ tướng nhóm chúng tôi và nhóm VR cờ tướng xây dựng nên, nhóm chúng tôi đã phát triển được mô hình chuyển đổi giọng nói cờ tướng sang văn bản có độ chính xác cao. Đồng thời, tích hợp được mô hình nhận diện giọng nói trên vào sản phẩm của nhóm VR và có thể sử dụng một cách mượt mà.

# Tài liệu


[1] Alex Graves **andothers**. "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks". **involume** 2006: **january** 2006, **pages** 369–376. DOI: 10.1145/1143844.1143891.

[2] Kenneth Heafield. "KenLM: Faster and Smaller Language Model Queries". **in** *WMT@EMNLP*: 2011.

[3] Kenneth Heafield **andothers**. "Scalable Modified Kneser-Ney Language Model Estimation". **involume** 2: **august** 2013, **pages** 690–696.

[4] Ashish Vaswani **andothers**. "Attention Is All You Need". **in** *CoRR*: abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[5] Steffen Schneider **andothers**. "wav2vec: Unsupervised Pre-Training for Speech Recognition". **inseptember** 2019: **pages** 3465–3469. DOI: 10.21437/Interspeech.2019-1873.

[6] Alexei Baevski **andothers**. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. **june** 2020.

[7] *Speech and Language Processing (3rd ed. draft) Dan Jurafsky and James H. Martin*. URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf.