ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: Công nghệ thông tin Việt – Nhật

ĐỀ TÀI:

XÂY DỰNG ỨNG DỤNG HỖ TRỢ VIẾT LỜI NHẠC RAP

Người hướng dẫn: **TS. ĐẶNG HOÀI PHƯƠNG**
Sinh viên thực hiện: **NGUYỄN NGỌC BẢO NHÂN**
Số thẻ sinh viên: **102210043**
Lớp**:** **21TCLC_Nhat1**

Đà Nẵng, 06/2025

概要

課題名: ラップの歌詞作成を支援するアプリケーションの構築
実施する学生: Nguyen Ngoc Bao Nhan (グエン・ゴック・バオ・ニャン)
学生証番号: 102210043　　クラス: 21TCLC_Nhat1

ベトナムではラップの人気が高まっており、特に若い愛好家の間でラップの作詞の需要が高まっている。しかし、ラップの歌詞を書くには、音韻、韻の踏み方、言語構造、感情表現など、高度な専門知識が必要だ。初心者はしばしば、韻を踏む単語を見つけること、テーマに沿って内容を構築すること、曲全体を通して一貫した感情的トーンを維持することに苦労する。

論文「ラップの作詞をサポートするアプリケーションの構築」は、これらの課題に直接取り組むために開発された。核となる解決策は、ベトナム語のGPT-2モデル（NlpHUST/gpt2-vietnamese）に基づく人工知能（AI）システムを開発し、ラップの作詞プロセスにおいてユーザーを支援することである。このモデルは文脈を考慮した次の文章を生成することができ、トピックと韻律の両方の要件に正確に一致する単語候補を提供するように微調整されている。

このシステムは、ビジネスロジックとAIモデルの統合のためにSpring Bootで開発されたバックエンド(BE)と、直感的でユーザーフレンドリーなインターフェイスを提供するためにReactJSで構築されたフロントエンド(FE)の2つの主要コンポーネントで構成されるWebアプリケーションとして実装されています。

期待される成果は、ユーザー、特に初心者が、テーマに関連し、リズム的に正しいベトナム語ラップの歌詞を簡単に作成できるようにするプラットフォームであり、それによって、より創造的で、効果的で、持続可能なラップ音楽への関与を可能にする。

TÓM TẮT

Tên đề tài: Xây dựng ứng dụng hỗ trợ viết lời nhạc rap
Sinh viên thực hiện: Nguyễn Ngọc Bảo Nhân
Số thẻ SV: 102210043 Lớp: 21TCLC_Nhat1

Trong bối cảnh nhạc rap ngày càng phát triển mạnh mẽ tại Việt Nam, nhu cầu sáng tác lời rap – đặc biệt từ các bạn trẻ yêu thích thể loại này – cũng gia tăng đáng kể. Tuy nhiên, quá trình viết lời rap lại đòi hỏi kiến thức chuyên môn cao về ngữ âm, kỹ thuật gieo vần, tư duy cấu trúc ngôn ngữ và khả năng biểu đạt cảm xúc. Những người mới bắt đầu thường gặp khó khăn trong việc tìm từ ngữ hợp vần, xây dựng nội dung theo chủ đề và duy trì mạch cảm xúc xuyên suốt bài rap.

Đồ án "**Xây dựng ứng dụng hỗ trợ viết lời nhạc rap**" được thực hiện nhằm giải quyết trực tiếp những thách thức nói trên. Giải pháp cốt lõi là phát triển một hệ thống trí tuệ nhân tạo (AI) ứng dụng mô hình **GPT-2 tiếng Việt** (NlpHUST/gpt2-vietnamese) để hỗ trợ người dùng trong quá trình sáng tác lời rap. Mô hình này có khả năng sinh câu tiếp theo theo ngữ cảnh, đồng thời được tinh chỉnh để đưa ra các gợi ý từ ngữ phù hợp với chủ đề và đảm bảo vần điệu chính xác.

Hệ thống được triển khai dưới dạng một ứng dụng web, bao gồm hai thành phần chính: **BE sử dụng Spring Boot** để xử lý logic nghiệp vụ và kết nối với mô hình AI, và **FE phát triển bằng ReactJS** để cung cấp giao diện trực quan, thân thiện với người dùng.

Kết quả kỳ vọng là một nền tảng hỗ trợ người dùng – đặc biệt là người mới – có thể dễ dàng sáng tác lời rap tiếng Việt đúng vần, đúng chủ đề, từ đó tiếp cận với nhạc rap một cách sáng tạo, hiệu quả và bền vững hơn.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Ngọc Bảo Nhân Số thẻ sinh viên: 102210043

Lớp: 21TCLC_Nhat1 Khoa: Công nghệ thông tin

Ngành: Công nghệ thông tin Việt - Nhật

1. *Tên đề tài đồ án:* Xây dựng ứng dụng hỗ trợ viết lời nhạc rap
2. *Đề tài thuộc diện:* ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
   - Mô hình tiền huấn luyện GPT-2 tiếng Việt (NlpHUST/gpt2-vietnamese)
4. *Nội dung các phần thuyết minh và tính toán:*
   Chương 1: Cơ sở lý thuyết
   Chương 2: Phân tích và thiết kế hệ thống
   Chương 3: Triển khai hệ thống
   Kết luận và hướng phát triển
5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*
   - Sơ đồ kiến trúc hệ thống
   - Lưu đồ xử lý dữ liệu và huấn luyện mô hình
   - Biểu đồ đánh giá chất lượng đầu ra của mô hình
   - Giao diện minh họa hệ thống web
6. *Họ tên người hướng dẫn:* TS. Đặng Hoài Phương
7. *Ngày giao nhiệm vụ đồ án:* *……/……/2025*
8. *Ngày hoàn thành đồ án:* *……/……/2025*

Đà Nẵng, ngày tháng 06 năm 2025

Trưởng Bộ môn …….. **Người hướng dẫn**

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ trí tuệ nhân tạo (AI) ngày càng phát triển mạnh mẽ và được ứng dụng rộng rãi trong nhiều lĩnh vực, việc đưa AI vào hỗ trợ sáng tác nội dung – đặc biệt là trong lĩnh vực âm nhạc – đang mở ra những hướng đi mới đầy tiềm năng. Thể loại nhạc rap, với đặc trưng ngôn ngữ phức tạp và đòi hỏi cao về khả năng gieo vần, đang ngày càng trở nên phổ biến tại Việt Nam. Tuy nhiên, quá trình sáng tác lời rap vẫn là một thách thức lớn đối với người mới bắt đầu.

Đề tài "Xây dựng ứng dụng hỗ trợ viết lời nhạc rap" được lựa chọn nhằm đáp ứng nhu cầu thực tiễn trong việc tạo ra một công cụ hỗ trợ hiệu quả cho người dùng trong việc sáng tác lời rap tiếng Việt. Đề tài ứng dụng mô hình ngôn ngữ GPT-2 đã được tiền huấn luyện trên tiếng Việt (NlpHUST/gpt2-vietnamese), kết hợp với các công nghệ phát triển phần mềm hiện đại như Spring Boot, ReactJS và MySQL để triển khai hệ thống dưới dạng ứng dụng web thân thiện và dễ sử dụng.

Thông qua quá trình nghiên cứu, phát triển và triển khai hệ thống, đồ án không chỉ hướng đến việc hiện thực hóa một công cụ hữu ích cho cộng đồng yêu thích rap mà còn góp phần ứng dụng hiệu quả AI vào lĩnh vực sáng tạo ngôn ngữ, đặc biệt là đối với tiếng Việt – một ngôn ngữ có cấu trúc phức tạp.

Em xin chân thành cảm ơn TS. Đặng Hoài Phương – giảng viên hướng dẫn – đã tận tình hỗ trợ và định hướng chuyên môn trong suốt quá trình thực hiện đồ án. Đồng thời, em cũng gửi lời cảm ơn đến các thầy cô trong Khoa Công nghệ Thông tin, gia đình và bạn bè đã luôn đồng hành, động viên và tạo điều kiện thuận lợi để em hoàn thành tốt đề tài này.

Tuy đã có nhiều nỗ lực, nhưng với giới hạn về thời gian và kinh nghiệm, đồ án khó tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ quý thầy cô và hội đồng phản biện để có thể hoàn thiện hơn trong tương lai.

CAM ĐOAN

Tôi xin cam đoan rằng toàn bộ nội dung trong báo cáo đồ án tốt nghiệp với đề tài "**Xây dựng ứng dụng hỗ trợ viết lời nhạc rap**" là kết quả nghiên cứu và thực hiện của cá nhân tôi dưới sự hướng dẫn của TS. Đặng Hoài Phương.

Tôi khẳng định không sao chép hoặc sử dụng bất kỳ phần nội dung nào từ các công trình khác mà không trích dẫn đầy đủ theo quy định. Các số liệu, kết quả và thông tin trình bày trong báo cáo là trung thực, được thực hiện nghiêm túc, tuân thủ các nguyên tắc về liêm chính học thuật theo quy định của Trường Đại học Bách khoa – Đại học Đà Nẵng.

Tôi hoàn toàn chịu trách nhiệm về tính trung thực và toàn vẹn của nội dung trong đồ án này.

Sinh viên thực hiện

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

CHỮ VIẾT TẮT:

Chữ viết tắt	**Diễn giải**
AI	Artificial Intelligence (Trí tuệ nhân tạo)
NLP	Natural Language Processing (Xử lý ngôn ngữ tự nhiên)
GPT-2	Generative Pre-trained Transformer 2 (Mô hình sinh ngôn ngữ tiền huấn luyện thế hệ thứ hai)
API	Application Programming Interface (Giao diện lập trình ứng dụng)
UI	User Interface (Giao diện người dùng)
UX	User Experience (Trải nghiệm người dùng)
GPU	Graphics Processing Unit (Bộ xử lý đồ họa)
Colab	Google Colaboratory (Nền tảng notebook cho Python có hỗ trợ GPU)
JSON	JavaScript Object Notation (Định dạng dữ liệu trao đổi phổ biến)
DB	Database (Cơ sở dữ liệu)
REST	Representational State Transfer (Kiến trúc giao tiếp API phổ biến)

MỞ ĐẦU

Tổng quan

Trong những năm gần đây, nhạc rap đã trở thành một trong những thể loại âm nhạc có sức lan tỏa mạnh mẽ trong giới trẻ Việt Nam. Với khả năng truyền tải thông điệp sâu sắc, tính sáng tạo trong ngôn ngữ và sự tự do trong biểu đạt, rap đã khẳng định được vị trí riêng trong đời sống âm nhạc đại chúng. Tuy nhiên, để sáng tác được lời nhạc rap chất lượng không phải là điều đơn giản. Quá trình này đòi hỏi người viết phải có kiến thức về ngôn ngữ, kỹ năng gieo vần, tư duy logic trong cấu trúc câu, và khả năng truyền tải cảm xúc.

Sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và đặc biệt là các mô hình ngôn ngữ như GPT (Generative Pre-trained Transformer) đã mở ra hướng ứng dụng mới trong lĩnh vực sáng tác ngôn ngữ. Việc kết hợp giữa AI và sáng tác nhạc rap có tiềm năng hỗ trợ người dùng, đặc biệt là người mới bắt đầu, vượt qua các rào cản kỹ thuật và sáng tạo. Đây cũng chính là nền tảng để thực hiện đồ án "Xây dựng ứng dụng hỗ trợ viết lời nhạc rap".

Mục đích

Đề tài được thực hiện nhằm xây dựng một hệ thống có khả năng hỗ trợ người dùng viết lời nhạc rap tiếng Việt thông qua việc gợi ý từ hợp vần và sinh nội dung theo chủ đề đầu vào. Hệ thống được triển khai dưới dạng ứng dụng web, tích hợp mô hình ngôn ngữ GPT-2 tiếng Việt nhằm tối ưu trải nghiệm sáng tác cho người dùng.

Mục tiêu đề tài

- Ứng dụng mô hình GPT-2 tiếng Việt để hỗ trợ sinh lời rap theo chủ đề và từ khóa vần yêu cầu.
- Xây dựng hệ thống backend bằng Spring Boot kết nối mô hình AI qua API và lưu dữ liệu đầu vào/đầu ra.
- Phát triển giao diện web bằng ReactJS cho phép người dùng tương tác dễ dàng với hệ thống.
- Đánh giá chất lượng đầu ra của mô hình và hiệu quả sử dụng hệ thống.

Phạm vi và đối tượng nghiên cứu

- Phạm vi:
  - Tập trung vào sinh lời nhạc rap tiếng Việt.
  - Chỉ hỗ trợ sinh các câu rap ngắn, đơn câu hoặc đoạn đơn, không thực hiện phân tích âm nhạc hay phối nhạc.

– Đối tượng sử dụng:
  - Người mới bắt đầu viết rap
  - Người yêu thích rap cần công cụ gợi ý sáng tạo
  - Người làm nội dung, truyền thông sáng tạo quan tâm đến ứng dụng AI trong ngôn ngữ

Phương pháp nghiên cứu

– Phân tích tài liệu lý thuyết về mô hình ngôn ngữ, đặc trưng lời rap, các kỹ thuật NLP.
– Thu thập và xử lý dữ liệu lời rap tiếng Việt từ các nguồn công khai.
– Tinh chỉnh mô hình GPT-2 tiếng Việt để sinh lời theo chủ đề và đúng vần.
– Xây dựng hệ thống web và kết nối backend – frontend – AI qua RESTful API.
– Kiểm thử và đánh giá chất lượng đầu ra bằng phương pháp định tính và định lượng.

Cấu trúc của đồ án tốt nghiệp

Đồ án bao gồm các nội dung chính sau:

Cơ sở lý thuyết – Trình bày những lý thuyết cơ sở được áp dụng trong đề tài.

Phân tích và thiết kế hệ thống – Trình bày bản phân tích, thiết kế để xây dựng hệ thống và luồng hoạt động của hệ thống.

Triển khai hệ thống – Trình bày cách thức cài đặt, sử dụng các công nghệ vào phát triển chương trình và kết quả chương trình đạt được.

Kết luận và hướng phát triển – Tổng kết những kết quả, bài học đã đạt được qua quá trình phát triển hệ thống, những hạn chế và phương hướng phát triển dự án trong tương lai.

Chương 1: CƠ SỞ LÝ THUYẾT

1.1 Mô tả bài toán

Trong bối cảnh âm nhạc Việt Nam hiện đại, nhạc Rap đã trở thành một hiện tượng văn hóa, thu hút đông đảo nghệ sĩ và giới trẻ. Sự phổ biến của thể loại này không chỉ đến từ giai điệu cuốn hút mà còn bởi tính chất dễ tiếp cận, khuyến khích sự sáng tạo cá nhân. Khán giả đại chúng có thể dễ dàng tìm thấy các track nhạc (beat) được chia sẻ rộng rãi, từ đó tự do thể hiện cảm xúc và câu chuyện của mình qua những lời rap.

Tuy nhiên, dù có vẻ dễ dàng, việc sáng tác một bài Rap hoàn chỉnh và chuyên nghiệp lại không hề đơn giản. Những nguyên tắc cơ bản và nâng cao trong việc xây dựng một bài nhạc Rap vẫn còn là rào cản lớn đối với nhiều người mới bắt đầu tại Việt Nam. Các tài liệu chính thống, chuyên sâu về lý thuyết sáng tác Rap bằng tiếng Việt còn khá hạn chế và chưa được phổ biến rộng rãi. Điều này dẫn đến việc người học thường phải tự mày mò, hoặc tìm kiếm thông tin từ các nguồn không chính thức, thiếu tính hệ thống.

Để hoàn thiện một bài Rap chất lượng, người sáng tác cần nắm vững nhiều yếu tố quan trọng như:

- Lựa chọn chủ đề (topic): Đảm bảo nội dung có ý nghĩa, mới mẻ và phù hợp với thông điệp muốn truyền tải.
- Vần điệu (rhyme scheme): Áp dụng các kỹ thuật gieo vần đa dạng, linh hoạt để tạo nên sự mượt mà và hấp dẫn cho lời ca.
- Flow (nhịp điệu và cách nhấn nhá): Khả năng sắp xếp từ ngữ, kết hợp với beat để tạo nên một dòng chảy âm thanh độc đáo, thể hiện cá tính của rapper.
- Tính cá nhân hóa (individuality): Khắc họa dấu ấn riêng của người viết thông qua cách sử dụng từ ngữ, lối tư duy và phong cách trình bày.
- Cấu trúc bài hát (song structure): Sắp xếp các phần như verse, chorus, bridge một cách hợp lý để bài hát có tính liền mạch và phát triển.

Đặc biệt, việc sử dụng linh hoạt các từ ngữ để vừa truyền tải đúng ý nghĩa, vừa đảm bảo tính vần điệu và tạo nên sự uyển chuyển, mượt mà cho câu rap là một thử thách lớn. Nó đòi hỏi người viết không chỉ có vốn từ phong phú mà còn phải có khả năng cảm âm, nhạy bén trong việc phối hợp các âm tiết, thanh điệu để lời rap không bị gượng ép, thiếu tự nhiên.

Nhận thức được nhu cầu này, nhiều công cụ ứng dụng Trí tuệ Nhân tạo (AI) đã ra đời trên thế giới nhằm hỗ trợ quá trình sáng tác như DeepBeat, These Lyrics Do Not

Exist, Boomy, hay gần đây nhất là Suno với khả năng tạo ra bài hát hoàn chỉnh từ văn bản. Những công cụ này đã thể hiện tiềm năng lớn trong việc tự động hóa và hỗ trợ sáng tạo.

Mặc dù có những bước tiến vượt bậc, hầu hết các công cụ AI hiện có đều gặp phải một vấn đề chung nghiêm trọng: chưa được tối ưu cho ngôn ngữ tiếng Việt. Sự phức tạp của ngữ pháp, hệ thống thanh điệu, và đặc biệt là tính chất đa dạng của vần điệu trong tiếng Việt (vần đơn, vần đôi, vần hỗn hợp, vần xuyên tâm, vần chéo,...) khiến các mô hình AI hiện tại gặp khó khăn trong việc tạo ra những câu rap tự nhiên, đúng ngữ nghĩa, và tuân thủ các quy tắc vần một cách chính xác. Kết quả là, những sản phẩm được tạo ra thường gượng gạo, thiếu tính nghệ thuật, hoặc không thể hiện được "chất" riêng của Rap tiếng Việt.

Do đó, một công cụ ứng dụng AI chuyên biệt, có khả năng hỗ trợ người mới tiếp cận với nhạc Rap bằng ngôn ngữ tiếng Việt là vô cùng cấp thiết. Công cụ này cần có khả năng:

- Tạo ra các câu rap đơn giản: Giúp người dùng dễ dàng bắt đầu mà không bị choáng ngợp bởi độ phức tạp.
- Đảm bảo tính chính xác về vần điệu: Đây là yếu tố then chốt để tạo nên sự mượt mà và tính nhạc cho lời rap.
- Hiểu và xử lý tốt ngữ pháp, từ vựng tiếng Việt: Để sản phẩm cuối cùng tự nhiên và có ý nghĩa.

Việc phát triển một công cụ như vậy sẽ không chỉ hạ thấp rào cản cho những người đam mê Rap tại Việt Nam, giúp họ dễ dàng thử sức và phát triển khả năng sáng tạo, mà còn góp phần thúc đẩy sự phát triển của thể loại Rap tiếng Việt nói chung trong cộng đồng.

Trước khi đi sâu vào mô hình chính mà tôi sẽ trình bày trong đồ án này, chúng ta hãy cùng tìm hiểu hai mô hình nổi bật trong lĩnh vực sinh lời nhạc rap. cần lưu ý rằng, mặc dù các mô hình được đề cập sau đây đã có thể được cải tiến và khắc phục những hạn chế, phát huy ưu điểm ở thời điểm hiện tại, nhưng vì những hạn chế trong việc tiếp cận tài liệu mới nhất, tôi sẽ tập trung phân tích dựa trên các báo cáo có sẵn tại thời điểm nghiên cứu.

1.2 Mô hình DeepBeat

1.2.1 Nguồn gốc và tác giả

Mô hình DeepBeat [1] được phát triển bởi một nhóm nghiên cứu tài năng gồm 5 tác giả: Eric Malmi (Aalto University và HIIT, Espoo, Phần Lan), Pyry Takala (Aalto University, Espoo, Phần Lan), Hannu Toivonen (University of Helsinki và HIIT, Helsinki, Phần Lan), Tapani Raiko (Aalto University, Espoo, Phần Lan), và Aristides

Gionis (Aalto University và HIIT, Espoo, Phần Lan). Công trình của họ, với tựa đề "DopeLearning: A Computational Approach to Rap Lyrics", được công bố vào tháng 6 năm 2016 trên arXiv (Cornell University), thể hiện một bước tiến quan trọng trong lĩnh vực ứng dụng Trí tuệ Nhân tạo vào sáng tạo âm nhạc. Mục tiêu của DeepBeat là giải quyết thách thức cốt lõi trong việc sáng tác lời nhạc rap – một quá trình đòi hỏi sự kết hợp phức tạp giữa khả năng xây dựng nội dung có ý nghĩa (truyền tải câu chuyện, thông điệp) và kỹ năng lyrical điêu luyện (áp dụng vần điệu, flow phức tạp). Cụ thể, bài toán mà mô hình này hướng tới là tự động hóa quá trình tạo lời rap bằng cách dự đoán dòng lời rap tiếp theo phù hợp nhất dựa trên ngữ cảnh của các dòng đã có. Điều này giúp người dùng, đặc biệt là những người chưa có nhiều kinh nghiệm, có thể tạo ra lời rap có vần điệu và ý nghĩa một cách hiệu quả hơn.

1.2.2 Ý tưởng thiết kế

DeepBeat dựa trên một hướng tiếp cận độc đáo, đó là **truy xuất thông tin (Information Retrieval - IR)**, thay vì nỗ lực sinh ra từng từ một như các mô hình ngôn ngữ truyền thống. Điều này có nghĩa là, mô hình sẽ không tự tạo ra lời mới hoàn toàn, mà sẽ tìm kiếm và lựa chọn dòng rap phù hợp nhất từ một kho dữ liệu khổng lồ các lời rap đã có sẵn. Cách tiếp cận này đảm bảo tính tự nhiên, phong phú về từ vựng và cấu trúc, đồng thời tận dụng được các mẫu vần điệu đã được chứng minh là hiệu quả từ những rapper thực thụ. Để thực hiện việc lựa chọn thông minh này, mô hình kết hợp hai kỹ thuật học máy mạnh mẽ: thuật toán RankSVM [1] để xếp hạng mức độ phù hợp của các dòng ứng cử viên, và một mô hình mạng nơ-ron sâu (Deep Neural Network) với cấu trúc đặc biệt để nắm bắt ngữ nghĩa phức tạp.

1.2.3 Cách hoạt động

Quá trình hoạt động của DeepBeat bắt đầu bằng việc chuẩn bị dữ liệu huấn luyện. Mô hình được trang bị một tập dữ liệu rất lớn, bao gồm hơn nửa triệu dòng lời nhạc từ 10.980 bài hát của 104 nghệ sĩ rap nói tiếng Anh nổi tiếng. Dữ liệu này trải qua quá trình tiền xử lý kỹ lưỡng: loại bỏ các ký tự không phải ASCII và các dòng chỉ có một từ, đưa các từ về dạng gốc (stemming) và viết thường (lower-casing). Để đảm bảo tính đồng nhất, các dòng quá dài (hơn 13 từ) được loại bỏ và các dòng ngắn hơn được thêm phần đệm (padding) để có độ dài cố định. Ngoài ra, các dòng lặp lại trong một bài hát và các bài hát mang tính chất "intro", "outro", "skit", "interlude" cũng được loại bỏ vì chúng thường chứa lời nói thông thường hơn là lời rap.

Khi người dùng muốn tạo lời rap, mô hình sẽ xác định bài toán dự đoán dòng tiếp theo. Người dùng sẽ cung cấp một hoặc một vài dòng rap đã có, được gọi là "truy vấn" (ký hiệu $B = (s_1, \ldots, s_m)$). Đồng thời, mô hình có sẵn một tập hợp lớn các "dòng ứng cử viên" (ký hiệu $C = \{l_1, \ldots, l_k\}$) từ kho dữ liệu đã được tiền xử lý. Mục tiêu của DeepBeat

là tìm ra dòng rap trong tập ứng cử viên (C) mà nó tin là phù hợp nhất để nối tiếp các dòng rap đã cho (B).

Để đánh giá mức độ phù hợp giữa các dòng rap đã có và một dòng ứng cử viên, mô hình tiến hành trích xuất các đặc trưng từ cả hai phía. Các đặc trưng này được phân loại thành ba nhóm chính, cung cấp cái nhìn toàn diện về mối quan hệ giữa các dòng:

(i) Nhóm thứ nhất là, **Đặc trưng về vần điệu (Rhyming features)**. DeepBeat đánh giá vần điệu thông qua ba chỉ số. Đặc trưng "vần cuối dòng" (EndRhyme) đo lường số lượng nguyên âm trùng khớp ở cuối dòng hiện tại (s_m) và dòng ứng cử viên (l_i), bỏ qua các phụ âm và khoảng trắng.

Dòng	*Phiên âm*
pay for	pei fɔ:r
stay warm	stei wɔ:rm

Đặc trưng "vần cuối dòng-1" (EndRhyme-1) tương tự nhưng so sánh với dòng trước đó (s_{m-1}) nhằm phát hiện các kiểu vần xen kẽ phổ biến (ví dụ: "abab"). Cuối cùng, đặc trưng "vần khác" (Other Rhyme) tính trung bình số lượng nguyên âm trùng khớp dài nhất trên mỗi từ giữa dòng ứng cử viên và dòng cuối cùng của truy vấn (s_m). Đặc trưng này quan trọng vì nó nắm bắt các loại vần không nằm ở cuối dòng (vần nội bộ, vần đa âm tiết), và để thực hiện được, lời rap được chuyển đổi sang dạng phiên âm ngữ âm (phonetic transcription).

(ii) Nhóm đặc trưng thứ hai là, **Đặc trưng về cấu trúc (Structural Similarity)**. Trong nhóm này, "độ dài dòng" (LineLength) là đặc trưng chính, đo lường sự khác biệt về độ dài ký tự giữa dòng ứng cử viên và dòng cuối cùng của truy vấn. Các dòng rap thường có độ dài tương đối giống nhau do cần được đọc trong cùng một nhịp nhạc (một bar/measure nhạc), nên đặc trưng này giúp duy trì sự đồng đều trong cấu trúc.

(iii) Nhóm đặc trưng cuối cùng và phức tạp nhất là, **Đặc trưng về ngữ nghĩa (Semantic Similarity)**, giúp mô hình hiểu được ý nghĩa của lời rap. Ba phương pháp được sử dụng bao gồm:

- **"Túi từ"** (Bag-of-Words - BOW) biểu diễn các dòng rap dưới dạng tập hợp các từ mà không quan tâm đến thứ tự, sau đó tính toán độ tương đồng Jaccard [1] giữa "túi từ" của dòng ứng cử viên và "túi từ" của các dòng trước đó (thường là 5 dòng gần nhất để có ngữ cảnh tốt hơn).
- **"Phân tích ngữ nghĩa tiềm ẩn"** (Latent Semantic Analysis - LSA) giúp mô hình hiểu được mối quan hệ ngữ nghĩa giữa các từ (từ đồng nghĩa, đa nghĩa) mà BOW không làm được; nó biến các dòng thành các vector trong

một không gian ngữ nghĩa chiều thấp, sau đó tính độ tương đồng cosin giữa các vector này.

- Mạnh mẽ nhất là **"Mô hình ngôn ngữ mạng nơ-ron" (Neural Network - NN5)**, sử dụng một mạng nơ-ron sâu để học cách biểu diễn các từ thành các vector (word embeddings), sau đó kết hợp chúng để tạo thành biểu diễn vector cho cả dòng, và cuối cùng là cho nhiều dòng liên tiếp. Mạng nơ-ron này được huấn luyện để dự đoán liệu một dòng có phải là dòng tiếp theo phù hợp hay không, mang lại khả năng hiểu ngữ nghĩa vượt trội.

Sau khi trích xuất tất cả các đặc trưng cho từng cặp (dòng đã có, dòng ứng cử viên), mô hình tiến hành xếp hạng các dòng ứng cử viên bằng cách sử dụng thuật toán RankSVM (Rank Support Vector Machine). RankSVM là một kỹ thuật học máy giám sát được thiết kế đặc biệt cho các bài toán xếp hạng. Không giống như các bộ phân loại truyền thống chỉ đơn thuần phân loại các mục thành các lớp riêng biệt, RankSVM học cách sắp xếp một danh sách các mục theo một thứ tự ưu tiên nhất định.

Trong ngữ cảnh của DeepBeat, RankSVM được huấn luyện để so sánh các cặp dòng lời rap: một dòng là "dòng tiếp theo đúng" (từ dữ liệu gốc) và một dòng là "dòng không đúng" (được chọn ngẫu nhiên). Mục tiêu của thuật toán là học một hàm trọng số để đảm bảo rằng dòng "đúng" luôn được xếp hạng cao hơn dòng "không đúng". Cụ thể, RankSVM sẽ học các trọng số cho từng đặc trưng đã trích xuất (vần điệu, độ dài, ngữ nghĩa). Các trọng số này phản ánh mức độ quan trọng của mỗi đặc trưng trong việc xác định sự phù hợp của một dòng lời rap. Mô hình được huấn luyện dựa trên nguyên lý tối thiểu hóa sai số trong việc xếp hạng các cặp, thay vì tối thiểu hóa sai số phân loại. Với các trọng số đã học, mô hình tính toán một "điểm phù hợp" (relevance score) cho mỗi dòng ứng cử viên. Điểm này là tổng có trọng số của tất cả các đặc trưng, thể hiện mức độ liên quan và phù hợp của dòng ứng cử viên đó với ngữ cảnh đã cho.

Cuối cùng, trong giai đoạn tạo lời rap, dòng ứng cử viên nào có điểm phù hợp cao nhất sẽ được chọn làm dòng tiếp theo. Quá trình này được lặp đi lặp lại: dòng mới được chọn sẽ được thêm vào "truy vấn" (các dòng đã có), và mô hình lại tiếp tục tìm kiếm dòng tiếp theo, cho đến khi đạt được độ dài bài hát mong muốn. Trong suốt quá trình này, thuật toán cũng kiểm tra để đảm bảo rằng dòng được chọn không chỉ phù hợp về ngữ nghĩa mà còn có vần điệu tốt với các dòng trước đó. Tóm lại, DeepBeat hoạt động như một "DJ" thông minh, có khả năng lựa chọn và kết hợp các dòng rap có sẵn từ một thư viện khổng lồ, dựa trên các tiêu chí về vần điệu, cấu trúc và ngữ nghĩa, để tạo ra một

bài rap mới, liền mạch và có ý nghĩa. Mô hình này đã được triển khai dưới dạng công cụ trực tuyến DeepBeat.

1.2.4 Các hạn chế

Mặc dù mô hình DeepBeat thể hiện một bước tiến đáng kể trong việc hỗ trợ sáng tác lời rap tiếng Anh, nhưng khi xét đến việc áp dụng cho bài toán viết rap tiếng Việt, nó bộc lộ nhiều hạn chế đáng kể. Những hạn chế này không được nêu trực tiếp trong nghiên cứu (vì nghiên cứu tập trung vào tiếng Anh), nhưng có thể suy luận logic dựa trên kiến trúc của mô hình và đặc thù của ngôn ngữ tiếng Việt.

(i) Hạn chế về ngôn ngữ và dữ liệu huấn luyện

Nghiên cứu khẳng định rằng DeepBeat được huấn luyện trên một tập dữ liệu rất lớn gồm hơn nửa triệu dòng lời từ 10.980 bài hát của 104 nghệ sĩ rap nói tiếng Anh nổi tiếng. Điều này có nghĩa là toàn bộ kho kiến thức mà mô hình học được, từ cách gieo vần, cấu trúc câu, ngữ nghĩa, cho đến phong cách biểu đạt, đều dựa trên đặc điểm của tiếng Anh.

- **Tính chất đa âm của tiếng Việt**: Tiếng Việt là một ngôn ngữ đơn lập, nghĩa là mỗi từ thường có ý nghĩa riêng và không thay đổi hình thái (ví dụ: "đi" luôn là "đi", không biến đổi thành "goes" hay "went"). ngoài ra, tiếng Việt có hệ thống thanh điệu phức tạp (6 thanh dấu), mỗi thanh có thể thay đổi hoàn toàn nghĩa của từ. ngược lại, tiếng Anh là ngôn ngữ biến đổi hình thái từ, ví dụ động từ "go" nghĩa là “đi” có thể biến thành "goes", "went", "gone" tùy theo thì hoặc chủ ngữ. sự khác biệt cơ bản này khiến các phương pháp xử lý ngôn ngữ tự nhiên (NLP) được tối ưu cho tiếng Anh không thể hoạt động hiệu quả hoặc thậm chí là không thể hoạt động với tiếng Việt nếu không có sự điều chỉnh lớn và dữ liệu tương ứng.
- **Vấn đề vần điệu và thanh điệu**: Nghiên cứu mô tả cách DeepBeat tính toán vần điệu bằng cách so sánh số lượng nguyên âm trùng khớp và sử dụng phiên âm ngữ âm. Tuy nhiên, hệ thống vần trong tiếng Việt phức tạp hơn nhiều. Không chỉ dựa vào việc trùng khớp âm cuối, vần trong tiếng Việt còn phải tuân thủ các quy tắc về vần luật (vần bằng, vần trắc), tương hợp dấu thanh, và cả cách ngắt nhịp, gieo vần chéo, vần đôi... Một "vần" trong tiếng Anh có thể chỉ là sự tương đồng về âm cuối, nhưng trong tiếng Việt, sự "vần" còn phải đi kèm với sự hài hòa về thanh điệu để câu rap không bị "ngang", "phô". Mô hình được huấn luyện trên tiếng Anh sẽ

không thể nhận diện hay tạo ra các vần điệu chuẩn và tự nhiên theo ngữ âm tiếng Việt.

(ii) Hạn chế trong trích xuất đặc trưng ngữ nghĩa

Mặc dù DeepBeat sử dụng các phương pháp mạnh mẽ như Túi từ (Bag-of-Words - BOW), Phân tích ngữ nghĩa tiềm ẩn (Latent Semantic Analysis - LSA) và đặc biệt là Mô hình Ngôn ngữ Mạng Nơ-ron (Neural Network - NN5) để nắm bắt ngữ nghĩa, các phương pháp này đều có hạn chế khi chuyển đổi sang tiếng Việt:

- **BOW và LSA:** Các phương pháp này phụ thuộc rất nhiều vào cấu trúc từ vựng và mối quan hệ ngữ nghĩa học được từ tập dữ liệu. Với sự khác biệt lớn về cấu trúc ngữ pháp và từ vựng giữa tiếng Anh và tiếng Việt, các biểu diễn BOW hay không gian LSA được xây dựng từ tiếng Anh sẽ không có ý nghĩa hoặc không hiệu quả khi áp dụng cho tiếng Việt. Các vấn đề như từ đồng âm (cùng âm nhưng khác nghĩa), từ đa nghĩa (một từ nhiều nghĩa), và cách dùng từ láy trong tiếng Việt là những thách thức lớn mà các đặc trưng này, vốn học từ tiếng Anh, không thể giải quyết chính xác.
- **Neural Network (NN5) và Word Embeddings**: Mô hình NN5 sử dụng "Word Embeddings" (biểu diễn từ thành vector) để nắm bắt ngữ nghĩa. Các Word Embeddings này được huấn luyện trên một lượng lớn dữ liệu tiếng Anh. Để DeepBeat có thể hoạt động với tiếng Việt, cần phải có một tập hợp Word Embeddings tương ứng đã được huấn luyện riêng cho tiếng Việt, phản ánh đúng mối quan hệ ngữ nghĩa trong tiếng Việt. Nếu không, các biểu diễn này sẽ không mang ý nghĩa ngữ nghĩa nào trong tiếng Việt, làm mất đi khả năng hiểu và tạo lời rap có ngữ cảnh của mô hình.

(iii) Hạn chế về sự phù hợp văn hóa và phong cách

Đây cũng chính là hạn chế lớn nhất đối với hầu hết các mô hình hiện tại:

- **Văn hóa và ngữ cảnh**: Lời rap không chỉ là sự kết hợp từ ngữ mà còn phản ánh văn hóa, ngữ cảnh xã hội, và các lối chơi chữ đặc trưng của từng ngôn ngữ. Các lời rap tiếng Anh có thể chứa các thành ngữ, tiếng lóng, hay lối nói ví von mà tiếng Việt không có hoặc có cách thể hiện hoàn toàn khác. Mô hình DeepBeat, được huấn luyện để hiểu và tái tạo phong cách rap tiếng Anh, sẽ không thể tạo ra những lời rap tiếng Việt có "chất", tự nhiên và phù hợp với văn hóa rap Việt hiện đại.
- **Tính cá nhân hóa (Individuality)**: Nghiên cứu có đề cập đến khả năng của mô hình trong việc tạo ra lời rap có vần điệu và ý nghĩa,

nhưng việc tạo ra "tính cá nhân hóa" hay "dấu ấn riêng" của người nghệ sĩ là một thách thức lớn đối với bất kỳ mô hình AI nào, đặc biệt khi chuyển đổi ngôn ngữ. Lối flow, cách nhấn nhá, và phong cách viết đặc trưng của rapper tiếng Việt sẽ không được phản ánh nếu mô hình chỉ học từ dữ liệu tiếng Anh.

(iv) Yêu cầu về dữ liệu và huấn luyện lại

Để DeepBeat có thể giải quyết bài toán viết rap tiếng Việt, cần có những nỗ lực đáng kể:

- **Xây dựng tập dữ liệu lớn tiếng việt**: Đây là yêu cầu tiên quyết. Cần thu thập một lượng lớn lời rap tiếng Việt đã được sáng tác bởi các nghệ sĩ Việt Nam để huấn luyện lại mô hình từ đầu. Tập dữ liệu này phải đủ lớn và đa dạng để mô hình học được các quy tắc vần điệu, ngữ pháp, và phong cách của rap tiếng Việt.
- **Điều chỉnh kiến trúc mô hình**: Một số đặc trưng hoặc cách xử lý ngữ âm, ngữ pháp có thể cần được điều chỉnh hoặc thiết kế lại hoàn toàn để phù hợp với đặc thù của tiếng Việt (ví dụ: cách xử lý thanh điệu, từ láy, từ ghép).
- **Huấn luyện lại từ đầu**: Gần như toàn bộ mô hình cần được huấn luyện lại với dữ liệu tiếng Việt. Quá trình này sẽ tốn kém về tài nguyên tính toán và thời gian.

Tóm lại, trong khi ý tưởng và kiến trúc của DeepBeat là tiên tiến, việc áp dụng trực tiếp nó vào bài toán viết rap tiếng Việt mà không có sự thay đổi và huấn luyện lại đáng kể với dữ liệu tiếng Việt sẽ gặp phải những hạn chế nghiêm trọng về khả năng tạo vần, hiểu ngữ nghĩa, và thể hiện phong cách phù hợp với ngôn ngữ và văn hóa rap Việt.

1.3 Mô hình LyricJam

1.3.1 Nguồn gốc và tác giả

Mô hình LyricJam [2] được phát triển bởi nhóm tác giả Olga Vechtomova, Gaurav Sahu và Dhruv Kumar từ Đại học Waterloo. Công trình của họ, với tựa đề "LyricJam: A system for generating lyrics for live instrumental music", được công bố vào tháng 6 năm 2021, giới thiệu một hệ thống thú vị có khả năng tạo lời nhạc (lyrics) theo thời gian thực (real-time) dựa trên một luồng âm nhạc không lời (instrumental music) đang được chơi trực tiếp. Mục tiêu chính của LyricJam là hỗ trợ các nhạc sĩ, nghệ sĩ trong quá trình sáng tác lời, đặc biệt là những người thường bắt đầu với giai điệu và để cảm xúc từ âm nhạc dẫn dắt việc chọn chủ đề, từ ngữ cho lời bài hát.

1.3.2 Ý tưởng thiết kế

Để đạt được mục tiêu này, LyricJam giải quyết bài toán phức tạp là làm thế nào để kết nối không gian tiềm ẩn của âm thanh (latent space of audio) và văn bản (text representation). Nghĩa là, làm sao để mô hình có thể "hiểu" được cảm xúc, chủ đề từ âm nhạc đang phát và sinh ra những dòng lời phù hợp với cảm xúc và ngữ cảnh đó. Nghiên cứu đề xuất hai cách tiếp cận mới lạ để "căn chỉnh" (align) các không gian tiềm ẩn này, cho phép hệ thống tạo ra các dòng lời mới lạ khớp với nhạc cụ đang chơi trực tiếp. Một cách tiếp cận dựa trên sự căn chỉnh đối nghịch (adversarial alignment) của các biểu diễn tiềm ẩn của âm thanh và lời bài hát, trong khi cách khác học cách chuyển giao cấu trúc (transfer the topology) từ không gian tiềm ẩn của âm nhạc sang không gian tiềm ẩn của lời bài hát.

1.3.3 Cách hoạt động

Quá trình hoạt động của LyricJam có thể được chia thành các bước chính sau:

(i) Thu thập và xử lý dữ liệu huấn luyện

LyricJam cần một lượng lớn dữ liệu để học cách liên kết âm nhạc và lời bài hát. Dữ liệu này bao gồm các cặp (bài hát không lời, lời bài hát tương ứng). Dữ liệu âm thanh và văn bản sẽ được xử lý riêng biệt:

- Dữ liệu Âm thanh: Các bản nhạc không lời được đưa qua một mô hình học sâu để trích xuất các đặc trưng âm thanh quan trọng, biểu diễn chúng dưới dạng vector trong một không gian tiềm ẩn. Các đặc trưng này có thể bao gồm nhịp điệu, hòa âm, âm sắc, hoặc cảm xúc âm nhạc. Cụ thể hơn, nghiên cứu sử dụng VGGish (một biến thể của mô hình VGG được thiết kế cho âm thanh) để tạo ra các biểu diễn nhúng (embeddings) 128 chiều từ mỗi đoạn âm thanh 0.96 giây. Các embedding này được làm mượt (smoothed) theo thời gian để phản ánh sự thay đổi dần dần của âm nhạc.
- Dữ liệu Lời bài hát: Các dòng lời được mã hóa thành các biểu diễn nhúng câu (sentence embeddings), nắm bắt ý nghĩa ngữ nghĩa của chúng trong một không gian tiềm ẩn khác. Nghiên cứu sử dụng một phiên bản của Sentence-BERT (một mô hình BERT được tinh chỉnh để tạo ra sentence embeddings có ý nghĩa), cụ thể là mô hình "all-mpnet-base-v2" của Sentence-BERT để tạo ra các embedding 768 chiều. Mô hình này được huấn luyện trên một tập dữ liệu lớn bao gồm lời bài hát, các cặp paraphrase (câu đồng nghĩa), và các câu có ý nghĩa liên quan để đảm bảo các câu có ý nghĩa tương tự sẽ có vector gần nhau.

Mục tiêu ở bước này là có được hai loại biểu diễn vector riêng biệt nhưng có khả năng tương quan với nhau: một cho âm thanh và một cho văn bản.

(ii) Căn chỉnh không gian tiềm ẩn (Aligning Latent Spaces)

Đây là bước cốt lõi và đổi mới của LyricJam, nơi mô hình học cách "dịch" từ ngôn ngữ âm nhạc sang ngôn ngữ lời bài hát. Nghiên cứu đề xuất hai phương pháp chính để căn chỉnh:

- **Căn chỉnh đối nghịch (Adversarial Alignment)**: Phương pháp này sử dụng một kiến trúc tương tự Mạng đối nghịch sáng tạo (Generative Adversarial Network - GAN). Sẽ có hai thành phần chính:
    - Bộ Tạo (Generator - G): Học cách chuyển đổi biểu diễn vector của âm thanh (audio embedding) sang một biểu diễn vector trong không gian tiềm ẩn của văn bản (audio-based lyric embedding).
    - Bộ Phân Biệt (Discriminator - D): Cố gắng phân biệt xem một biểu diễn vector trong không gian văn bản là do Bộ Tạo tạo ra từ âm thanh, hay là một biểu diễn "thật" của lời bài hát (từ dữ liệu lời bài hát gốc).
    - Mục tiêu của Bộ Tạo là tạo ra các biểu diễn lời bài hát "giả" mà Bộ Phân Biệt không thể phân biệt được với lời bài hát "thật". Quá trình này buộc Bộ Tạo phải học cách ánh xạ âm thanh sang lời bài hát một cách có ý nghĩa ngữ nghĩa, tạo ra sự liên kết giữa hai không gian tiềm ẩn.
- **Chuyển giao cấu trúc (Topology Transfer)**: Phương pháp này tập trung vào việc đảm bảo rằng cấu trúc hay mối quan hệ tương đối giữa các đoạn nhạc sẽ được giữ nguyên khi chúng được "chuyển đổi" thành lời bài hát. hãy hình dung không gian tiềm ẩn (latent space) như một bản đồ. trên bản đồ âm nhạc, những bài hát buồn sẽ nằm gần nhau, những bài hát vui sẽ nằm gần nhau, và bài hát buồn sẽ xa bài hát vui. Mục tiêu của "chuyển giao cấu trúc" là khi chúng ta ánh xạ (map) các điểm từ bản đồ âm nhạc sang bản đồ lời bài hát, thì mối quan hệ tương đối giữa chúng vẫn phải được bảo toàn. Để đạt được điều này, các tác giả sử dụng một kỹ thuật đặc biệt gọi là Hàm lỗi Triplet (Triplet Loss) trong quá trình huấn luyện mô hình. Hàm lỗi Triplet hoạt động như sau:
    - Nó sẽ chọn ba điểm (hay còn gọi là "bộ ba" - triplet): Một điểm "gốc" (anchor - ví dụ: biểu diễn âm thanh của một đoạn nhạc buồn); một điểm "dương" (positive - ví dụ: biểu diễn âm thanh của một đoạn nhạc buồn khác, tương tự điểm gốc); một điểm

"âm" (negative - ví dụ: biểu diễn âm thanh của một đoạn nhạc vui, rất khác điểm gốc).

- Mục tiêu của Hàm lỗi Triplet là ép mô hình học cách biến đổi sao cho: Khoảng cách giữa biểu diễn lời bài hát của điểm "gốc" và điểm "dương" phải nhỏ. Và khoảng cách giữa biểu diễn lời bài hát của điểm "gốc" và điểm "âm" phải lớn, lớn hơn một ngưỡng nhất định.

Nhờ có Hàm lỗi Triplet, mô hình được "ép" phải học một phép ánh xạ mà không chỉ chuyển đổi đơn thuần mà còn giữ được cấu trúc, sự phân bố cảm xúc và chủ đề từ không gian âm nhạc sang không gian lời bài hát. Điều này giúp lời bài hát được sinh ra không chỉ hợp lý với từng đoạn nhạc mà còn mạch lạc và nhất quán về mặt cảm xúc trong suốt bài.

Cả hai phương pháp trên đều nhằm mục đích tạo ra một hàm ánh xạ (mapping function) hoặc một "cầu nối" giữa không gian tiềm ẩn của âm thanh và không gian tiềm ẩn của lời bài hát.

(iii) Sinh lời nhạc (Lyric Generation)

Khi hệ thống nhận được một luồng âm nhạc trực tiếp, quá trình sinh lời diễn ra như sau:

- **Trích xuất đặc trưng âm thanh theo thời gian thực**: Hệ thống liên tục lắng nghe và trích xuất các đặc trưng âm thanh từ luồng nhạc trực tiếp. Các đặc trưng này được chuyển đổi thành biểu diễn vector trong không gian tiềm ẩn của âm thanh.
- **Ánh xạ sang không gian văn bản**: Sử dụng hàm ánh xạ đã học được từ bước 2 (từ phương pháp căn chỉnh đối nghịch hoặc chuyển giao cấu trúc), biểu diễn vector âm thanh này được chuyển đổi thành một vector tương ứng trong không gian tiềm ẩn của văn bản. Vector này đại diện cho "ý nghĩa" hoặc "chủ đề" mà mô hình hiểu được từ đoạn nhạc đang phát.
- **Sinh ra dòng lời mới**: Vector ngữ nghĩa trong không gian văn bản này sau đó được đưa vào một mô hình ngôn ngữ sinh văn bản (ví dụ: một mô hình Transformer Decoder). LyricJam không sinh lời từng từ mà thực hiện hai bước:
    - Tìm kiếm dòng lời gần nhất (Nearest Neighbors Search): Hệ thống tìm kiếm các dòng lời đã biết (từ tập dữ liệu lời bài hát huấn luyện) mà có lyric embedding gần nhất với lyric

embedding tiềm năng vừa được tạo ra. Điều này giúp xác định chủ đề hoặc phong cách chung.

- Sinh lời điều kiện (Conditional Text Generation): Các dòng lời gần nhất này được sử dụng làm "ngữ cảnh" hoặc "điều kiện" cho một mô hình ngôn ngữ sinh văn bản. Mô hình này sẽ sinh ra một dòng lời mới, độc đáo, dựa trên ngữ cảnh được cung cấp. Mục tiêu là tạo ra lời mới mẻ nhưng vẫn phù hợp với ý nghĩa đã được ánh xạ từ âm nhạc.

(iv) Phản hồi và cải tiến (tùy chọn, cho tương tác người dùng)

Nghiên cứu cũng đề cập đến các nghiên cứu người dùng (user studies) cho thấy hệ thống không chỉ hữu ích trong việc sáng tác lời mà còn khuyến khích các nghệ sĩ ứng biến và tìm ra những biểu cảm âm nhạc mới. Điều này ngụ ý rằng có thể có cơ chế phản hồi để mô hình học hỏi và cải thiện theo thời gian dựa trên tương tác của người dùng, mặc dù chi tiết cụ thể về cơ chế này không được trình bày sâu trong tài liệu.

Tóm lại, LyricJam là một hệ thống tinh vi kết hợp xử lý tín hiệu âm thanh và xử lý ngôn ngữ tự nhiên, sử dụng các kỹ thuật học sâu tiên tiến để tạo ra lời bài hát đồng bộ và phù hợp với âm nhạc trực tiếp, mở ra một hướng đi mới trong việc hỗ trợ sáng tạo nghệ thuật.

1.3.4 Các hạn chế

Mô hình LyricJam được thiết kế và huấn luyện để tạo lời nhạc tiếng Anh, kết nối cảm xúc từ âm nhạc với biểu diễn văn bản. Tuy nhiên, khi xét đến việc áp dụng hệ thống này để sinh lời nhạc rap tiếng Việt, chúng ta sẽ gặp phải một số hạn chế cốt lõi do sự khác biệt sâu sắc giữa hai ngôn ngữ và văn hóa rap.

(i) Hạn chế về ngôn ngữ và dữ liệu huấn luyện

Toàn bộ quá trình học của LyricJam, từ cách biểu diễn lời bài hát đến việc căn chỉnh không gian tiềm ẩn, đều dựa trên đặc điểm của tiếng Anh và kho dữ liệu tiếng Anh.

Đặc thù ngôn ngữ tiếng Việt:

- **Ngôn ngữ đơn lập và thanh điệu phức tạp**: Tiếng Việt là ngôn ngữ đơn lập, mỗi âm tiết thường mang một nghĩa và không biến đổi hình thái. Quan trọng hơn, tiếng Việt có 6 thanh điệu, mỗi thanh điệu có thể thay đổi hoàn toàn ý nghĩa của từ (ví dụ: "ma", "mà", "má", "mạ", "mã", "mả"). Hệ thống LyricJam, được huấn luyện trên tiếng Anh (ngôn ngữ ít thanh điệu và có biến đổi hình thái từ), sẽ không thể "hiểu" hoặc xử lý hiệu quả thông tin

thanh điệu này. Điều này ảnh hưởng trực tiếp đến cả ngữ nghĩa và vần điệu của lời nhạc tiếng Việt.

- **Vần điệu đa dạng và phức tạp**: Vần trong tiếng Việt không chỉ dừng lại ở việc trùng khớp âm cuối như tiếng Anh. Nó bao gồm các luật vần bằng/trắc, vần chéo, vần đôi, và sự hài hòa về thanh điệu. Một "vần" hoàn hảo trong tiếng Anh có thể không hề "vần" hoặc nghe rất "phô" trong tiếng Việt nếu không có sự tương hợp về thanh. LyricJam, khi căn chỉnh không gian lời bài hát (sử dụng Sentence-BERT được huấn luyện trên tiếng Anh), sẽ không thể nắm bắt được các quy tắc vần điệu phức tạp này của tiếng Việt.

Dữ liệu huấn luyện không tương thích

- **Biểu diễn nhúng âm thanh (Audio Embeddings - VGGish)**: Mặc dù VGGish được huấn luyện trên AudioSet (một tập dữ liệu âm thanh tổng quát), nó không được huấn luyện cụ thể để nhận diện các đặc điểm âm nhạc hoặc cảm xúc có thể đặc trưng cho rap tiếng Việt. Khả năng nó trích xuất các đặc trưng âm nhạc phù hợp cho việc sinh lời Việt là chưa rõ ràng.
- **Biểu diễn nhúng câu (Sentence Embeddings - Sentence-BERT)**: Mô hình Sentence-BERT được sử dụng (all-mpnet-base-v2) đã được huấn luyện trên các tập dữ liệu tiếng Anh khổng lồ. Các embedding này mã hóa ngữ nghĩa và mối quan hệ từ vựng của tiếng Anh. Khi đưa văn bản tiếng Việt vào, các embedding này sẽ không mang ý nghĩa ngữ nghĩa chính xác hoặc không hiệu quả, dẫn đến việc mô hình không thể "hiểu" được nội dung tiếng Việt.

(ii) Hạn chế trong cơ chế căn chỉnh và sinh lời

Các phương pháp căn chỉnh và sinh lời cốt lõi của LyricJam sẽ gặp thách thức lớn khi áp dụng cho tiếng Việt.

- **Căn chỉnh đối nghịch (Adversarial Alignment) và chuyển giao cấu trúc (Topology Transfer)**: Cả hai phương pháp này đều dựa trên việc ánh xạ từ không gian âm thanh sang không gian biểu diễn lời bài hát. Nếu không gian lời bài hát (được xây dựng từ tiếng Anh) không thể biểu diễn chính xác tiếng Việt, thì quá trình ánh xạ này sẽ thất bại. Các mối quan hệ mà mô hình học được giữa âm thanh và lời bài hát tiếng Anh sẽ không thể áp dụng cho cặp âm thanh-lời bài hát tiếng Việt.
- **Sinh lời điều kiện (Conditional Text Generation)**: Mô hình sinh văn bản được sử dụng (ví dụ: GPT-2 hoặc tương tự) cần phải được huấn luyện trên một lượng lớn dữ liệu văn bản tiếng Việt để có thể sinh ra các câu có

ngữ pháp, từ vựng và phong cách tự nhiên. Nếu chỉ dựa vào mô hình được huấn luyện bằng tiếng Anh và cố gắng sinh lời tiếng Việt, kết quả sẽ là những dòng lời không có nghĩa, sai ngữ pháp, hoặc không tự nhiên.

- **Tìm kiếm dòng lời gần nhất (Nearest Neighbors Search)**: Bước này dựa vào việc tìm kiếm các dòng lời có embedding gần nhất. Nếu các embedding của lời bài hát không phù hợp với tiếng Việt, việc tìm kiếm sẽ không mang lại các dòng lời có ý nghĩa hay liên quan.

(iii) Hạn chế về phong cách và văn hóa Rap Việt

Rap Việt có những đặc điểm riêng về flow, cách chơi chữ, sử dụng tiếng lóng, thành ngữ, và chủ đề phản ánh văn hóa Việt Nam.

- **Thiếu ngữ cảnh văn hóa**: Mô hình được huấn luyện trên lời rap tiếng Anh sẽ không có bất kỳ "kiến thức" nào về các yếu tố văn hóa đặc trưng của rap Việt. Do đó, lời được sinh ra có thể sẽ không có "chất" Việt, không sử dụng đúng tiếng lóng, hoặc không phản ánh các chủ đề phổ biến trong rap Việt.
- **Flow và nhịp điệu tiếng Việt**: Flow trong rap tiếng Việt chịu ảnh hưởng lớn bởi thanh điệu và cách ngắt nhịp của ngôn ngữ. Một mô hình không được huấn luyện trên dữ liệu rap tiếng Việt sẽ không thể tạo ra flow tự nhiên, trôi chảy và hấp dẫn.

(iv) Yêu cầu đối với việc áp dụng cho tiếng Việt

Để LyricJam có thể hoạt động hiệu quả với tiếng Việt, cần có những nỗ lực đáng kể:

- Xây Dựng Tập Dữ Liệu Lớn Âm thanh-Lời Bài Hát Tiếng Việt: Đây là yêu cầu cấp thiết nhất. Cần thu thập hàng nghìn (hoặc hàng chục nghìn) bài hát rap tiếng Việt cùng với lời bài hát tương ứng để huấn luyện lại toàn bộ hệ thống.
- Huấn Luyện Lại Các Mô Hình Biểu Diễn: Các mô hình tạo Audio Embeddings (nếu cần sự đặc thù hơn cho nhạc Việt) và đặc biệt là Sentence Embeddings cần được huấn luyện lại hoặc tinh chỉnh (fine-tune) trên tập dữ liệu tiếng Việt. Có thể cần sử dụng các kiến trúc tùy chỉnh để xử lý thanh điệu.
- Huấn Luyện Lại Mô Hình Căn Chỉnh và Sinh Lời: Toàn bộ các bước căn chỉnh (Adversarial Alignment, Topology Transfer) và sinh lời (Conditional Text Generation) cần được huấn luyện lại với dữ liệu tiếng Việt, sử dụng các biểu diễn đã được tối ưu cho tiếng Việt.

Tóm lại, mặc dù LyricJam là một giải pháp sáng tạo, việc áp dụng trực tiếp nó vào bài toán sinh lời nhạc rap tiếng Việt mà không có sự tái thiết kế và huấn luyện lại triệt để với dữ liệu tiếng Việt sẽ dẫn đến hiệu suất rất hạn chế, chủ yếu do sự khác biệt cơ bản về ngữ âm, ngữ pháp, vần điệu, và văn hóa giữa tiếng Anh và tiếng Việt.

1.4 Kết chương

Qua chương vừa rồi đã cung cấp cái nhìn tổng quan về lĩnh vực sinh lời nhạc tự động, một nhánh thú vị của Trí tuệ nhân tạo kết hợp với sáng tạo nghệ thuật. Chúng ta đã cùng tìm hiểu về khái niệm, vai trò, và những thách thức cơ bản của bài toán này. Đặc biệt, chương này đã đi sâu vào phân tích cơ chế hoạt động của hai mô hình tiêu biểu.

Đầu tiên là DeepBeat, một phương pháp tiếp cận dựa trên truy xuất thông tin (information retrieval) và học máy truyền thống kết hợp mạng nơ-ron (RankSVM và NN5). DeepBeat đã chứng minh khả năng vượt trội trong việc tạo ra lời rap có vần điệu và ngữ nghĩa mạch lạc dựa trên dữ liệu tiếng Anh có sẵn. Tuy nhiên, qua phân tích, chúng ta cũng nhận thấy những hạn chế cố hữu của nó khi áp dụng cho tiếng Việt, chủ yếu do sự khác biệt về đặc trưng ngôn ngữ, hệ thống vần điệu, và văn hóa.

Tiếp theo, chúng ta đã khám phá LyricJam, một hệ thống hiện đại hơn, sử dụng các kỹ thuật học sâu tiên tiến (như biểu diễn nhúng âm thanh, Sentence-BERT, và các phương pháp căn chỉnh không gian tiềm ẩn như Adversarial Alignment và Topology Transfer) để sinh lời nhạc theo thời gian thực từ âm nhạc không lời. LyricJam thể hiện tiềm năng lớn trong việc kết nối cảm xúc âm nhạc và lời bài hát. Mặc dù vậy, khi đối diện với bài toán tiếng Việt, LyricJam cũng bộc lộ những rào cản tương tự DeepBeat, đặc biệt liên quan đến sự phức tạp của thanh điệu, ngữ âm tiếng Việt và yêu cầu về dữ liệu huấn luyện đặc thù.

Nhìn chung, cả DeepBeat và LyricJam đều là những bước tiến quan trọng trong lĩnh vực sinh lời nhạc tự động. Tuy nhiên, việc áp dụng trực tiếp các phương pháp này cho tiếng Việt mà không có sự điều chỉnh và huấn luyện lại triệt để là một thách thức lớn. Những phân tích về hạn chế của các mô hình này trên tiếng Việt đã làm rõ hơn sự cần thiết của một hướng tiếp cận mới, phù hợp hơn với đặc điểm riêng của ngôn ngữ và văn hóa Việt Nam.

Chương tiếp theo sẽ đi sâu vào việc đề xuất một mô hình hoặc phương pháp tiếp cận mới, có thể khắc phục được những hạn chế đã nêu, nhằm giải quyết hiệu quả bài toán sinh lời nhạc rap tiếng Việt, mở ra những tiềm năng mới cho sự giao thoa giữa Trí tuệ Nhân tạo và nghệ thuật sáng tác tại Việt Nam.

Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Phân tích yêu cầu

2.1.1 Bài toán đặt ra

Sáng tác lời nhạc rap là một quá trình đòi hỏi sự kết hợp hài hòa giữa khả năng truyền tải ý nghĩa sâu sắc và kỹ năng lyrical điêu luyện. Đối với các nghệ sĩ rap, đặc biệt là những người mới bắt đầu hoặc muốn nâng cao năng suất sáng tạo, việc xây dựng một bài rap chất lượng cao thường gặp phải nhiều thách thức đáng kể.

Một trong những khó khăn lớn nhất trong quá trình này là khó khăn trong việc tìm kiếm các từ ngữ có vần điệu phù hợp và độc đáo. Vần điệu không chỉ là yếu tố tạo nên sự hài hòa âm thanh mà còn là nền tảng của "flow" (nhịp điệu và cách nhấn nhá) trong rap, giúp lời nhạc dễ nghe và dễ đi vào lòng người nghe. Tuy nhiên, việc liên tục tìm kiếm các từ, cụm từ có vần và ý nghĩa phù hợp là một quá trình tiêu tốn nhiều thời gian và đòi hỏi sự sáng tạo không ngừng từ người viết.

Bên cạnh đó, lời nhạc rap còn là một công cụ mạnh mẽ để truyền tải các thông điệp, bộc lộ suy nghĩ, và cảm xúc cá nhân. Do đó, việc duy trì chủ đề xuyên suốt cả bài hát là vô cùng quan trọng, giúp tác phẩm có chiều sâu nội dung và ý nghĩa mạch lạc. Nếu lời nhạc bị lạc đề hoặc không có sự kết nối chặt chẽ về chủ đề, bài rap có thể trở nên rời rạc và khó để người nghe cảm nhận trọn vẹn thông điệp mà nghệ sĩ muốn truyền tải.

Hiện tại trên thị trường, đã có một số công cụ hỗ trợ sáng tác lời nhạc, điển hình là Rhyme Finder được phát hành bởi J2Team vào năm 2023. Đây là một nỗ lực đáng ghi nhận trong việc cung cấp công cụ cho người Việt. Tuy nhiên, hạn chế lớn nhất của những công cụ hiện có nằm ở kho dữ liệu từ điển tiếng Việt chưa đầy đủ. Tiếng Việt không chỉ bao gồm các từ vựng thuộc ngôn ngữ toàn dân mà còn có nội ngữ (tiếng của người đồng bào thiểu số) và các từ ngữ địa phương tùy theo vùng miền, tạo nên một sự phong phú lớn. Điều này gây khó khăn cho các công cụ hiện tại trong việc cung cấp một phạm vi vần và từ ngữ đa dạng, chính xác theo từng ngữ cảnh văn hóa và địa phương cụ thể của rap Việt. Sự thiếu vắng các công cụ AI chuyên biệt, có khả năng sinh lời theo chủ đề và vần một cách thông minh, và có kho từ vựng tiếng Việt phong phú, càng làm trầm trọng thêm những thách thức này, buộc người viết phải dựa hoàn toàn vào khả năng cá nhân và kinh nghiệm.

Các nghiên cứu và mô hình tiên tiến trong lĩnh vực sinh lời nhạc tự động như DeepBeat hay LyricJam (đã được phân tích trong Chương 1) đã đạt được những thành tựu nhất định với ngôn ngữ tiếng Anh. Tuy nhiên, khi áp dụng cho tiếng Việt, chúng bộc lộ nhiều hạn chế do sự khác biệt sâu sắc về ngữ pháp, thanh điệu, hệ thống vần điệu và đặc thù văn hóa. Điều này tạo ra một khoảng trống rõ rệt cho các giải pháp AI chuyên biệt, được thiết kế và tối ưu hóa cho ngôn ngữ tiếng Việt, có khả năng xử lý tốt hơn những khác biệt về văn hóa và ngôn ngữ.

Để giải quyết những vấn đề trên, đồ án này đề xuất xây dựng một mô hình sinh lời nhạc rap tiếng Việt tự động. Mô hình này được thiết kế để hỗ trợ người dùng bằng cách cung cấp khả năng kiểm soát chặt chẽ về chủ đề và vần điệu. Cụ thể, người dùng có thể nhập vào chủ đề mong muốn và từ vần cuối, sau đó mô hình sẽ sinh ra câu rap tiếp theo phù hợp với ngữ cảnh đã cho. Đặc biệt, mô hình của chúng tôi hướng tới việc cải thiện đáng kể về kho từ vựng, với khả năng thu thập và xử lý lên đến 29.515 từ, bao gồm cả các từ ngữ địa phương và nội ngữ, nhằm mang lại một phạm vi từ vựng và vần điệu phong phú hơn nhiều so với các công cụ hiện có. Với phương pháp tiếp cận này, mô hình sẽ không chỉ giúp người sáng tác tiết kiệm thời gian, công sức trong việc tìm vần, duy trì chủ đề mà còn hướng tới việc cung cấp một công cụ hỗ trợ hiệu quả, góp phần nâng cao chất lượng và tính sáng tạo của lời nhạc rap tiếng Việt.

2.1.2 Yêu cầu chức năng

- **Hệ thống tài khoản người dùng:** Hệ thống phải cho phép người dùng đăng ký tài khoản, đăng nhập và đăng xuất.
- **Sinh lời rap tự động:** Hệ thống phải có khả năng sinh ra một dòng lời rap tiếng Việt mới dựa trên đầu vào của người dùng.
- **Kiểm soát chủ đề:** Hệ thống phải cho phép người dùng nhập một chủ đề cụ thể (ví dụ: "cuộc sống", "gia đình", "xã hội") thông qua giao diện. Hệ thống phải sinh ra dòng lời có nội dung phù hợp và liên quan đến chủ đề đã nhập.
- **Kiểm soát vần điệu:** Hệ thống phải cho phép người dùng nhập một từ vần mong muốn (ví dụ: "anh", "an") thông qua giao diện. Hệ thống phải ưu tiên sinh ra dòng lời có từ cuối câu vần với từ đã cho.
- **Sử dụng ngữ cảnh:** Hệ thống phải có khả năng tiếp nhận một dòng lời rap làm ngữ cảnh từ người dùng.
- **Quản lý đầu vào và đầu ra:** Hệ thống phải cung cấp giao diện trực quan cho người dùng để nhập các thông tin cần thiết: chủ đề, từ vần và dòng ngữ cảnh.

- **Đảm bảo độ phủ từ vựng tiếng Việt:** Mô hình phải sử dụng kho từ vựng tiếng Việt rộng lớn (bao gồm các từ ngữ toàn dân, nội ngữ và từ địa phương với 29.515 từ) để đảm bảo sự đa dạng và phong phú trong lời rap được sinh ra. Điều này giúp nâng cao chất lượng vần và ngữ nghĩa theo đặc thù của ngôn ngữ.
- **Lưu trữ lịch sử phiên làm việc:** Hệ thống phải có khả năng lưu trữ các phiên sinh lời của từng người dùng đã đăng nhập, bao gồm các prompt đầu vào (chủ đề, vần, ngữ cảnh) và các dòng lời rap đã được mô hình sinh ra.
- **Thu thập phản hồi từ người dùng:** Hệ thống phải cho phép người dùng đánh giá chất lượng của dòng lời được sinh ra (ví dụ: bằng cách cung cấp lựa chọn "tốt" hoặc "không tốt" / "thích" hoặc "không thích"). Phản hồi này cần được lưu trữ cùng với lịch sử phiên làm việc để phục vụ mục đích phân tích và cải thiện mô hình sau này.
- **Xem danh sách từ vựng tiếng Việt:** Hệ thống phải cho phép xem danh sách các từ vựng tiếng Việt
- **Quản lý từ vựng tiếng Việt:** Hệ thống cho phép quản trị viên xem, thêm, xoá các từ vựng tiếng Việt

2.1.3 Yêu cầu phi chức năng

Ngoài các yêu cầu chức năng về khả năng sinh lời rap, hệ thống cần đáp ứng một số yêu cầu phi chức năng để đảm bảo trải nghiệm người dùng tối ưu và hiệu suất ổn định. Các yêu cầu phi chức năng chính bao gồm:

Hiệu suất

- Tốc độ phản hồi: Hệ thống phải có khả năng sinh lời rap và hiển thị kết quả cho người dùng trong thời gian ngắn (ví dụ: dưới 20 giây cho mỗi lần sinh lời) để duy trì sự liền mạch trong quá trình sáng tạo của người dùng.
- Tốc độ xử lý đăng nhập/đăng ký: Quá trình đăng ký và đăng nhập tài khoản phải diễn ra nhanh chóng, không gây chậm trễ đáng kể cho người dùng.

Độ tin cậy

- Hệ thống phải hoạt động ổn định và liên tục mà không gặp phải các lỗi nghiêm trọng hoặc sự cố đột ngột gây gián đoạn quá trình sử dụng.
- Khả năng lưu trữ dữ liệu người dùng (tài khoản, lịch sử phiên, đánh giá) phải đáng tin cậy, đảm bảo dữ liệu không bị mất mát.

Khả năng sử dụng

- Giao diện người dùng (UI): Giao diện phải được thiết kế rõ ràng, trực quan, dễ sử dụng cho cả người dùng có kinh nghiệm và người mới làm quen với công cụ sinh lời.

- Trải nghiệm người dùng (UX): Luồng tương tác phải mượt mà, logic, giúp người dùng dễ dàng nhập đầu vào và nhận kết quả mà không gặp khó khăn.

Khả năng mở rộng

- Hệ thống phải có khả năng mở rộng để đáp ứng được số lượng người dùng đồng thời tăng lên trong tương lai mà không ảnh hưởng đáng kể đến hiệu suất.
- Kiến trúc hệ thống cần cho phép việc bổ sung các tính năng mới hoặc nâng cấp mô hình AI một cách dễ dàng trong tương lai.

Bảo mật

- Bảo mật tài khoản: Hệ thống phải đảm bảo an toàn thông tin tài khoản người dùng (tên đăng nhập, mật khẩu) thông qua các biện pháp mã hóa và xác thực phù hợp.
- Bảo mật dữ liệu: Dữ liệu lịch sử sinh lời và đánh giá của người dùng cần được bảo vệ khỏi các truy cập trái phép.

Khả năng bảo trì

- Mã nguồn phải được viết rõ ràng, có cấu trúc tốt, dễ hiểu và dễ bảo trì để các nhà phát triển có thể dễ dàng sửa lỗi, cập nhật hoặc thêm tính năng mới.
- Tài liệu kỹ thuật và cấu hình hệ thống cần được cập nhật đầy đủ.

Khả năng thích nghi/linh hoạt

- Hệ thống cần có khả năng thích nghi với các thay đổi nhỏ về định dạng đầu vào hoặc đầu ra trong tương lai (nếu cần thiết).
- Kiến trúc cần hỗ trợ việc dễ dàng cập nhật kho từ vựng tiếng Việt.

2.2 Xây dựng mô hình ứng dụng AI

Phần này mô tả chi tiết quá trình xây dựng mô hình Trí tuệ Nhân tạo làm nền tảng cho hệ thống sinh lời nhạc rap tiếng Việt. Việc xây dựng mô hình được thực hiện theo các bước chính, từ việc lựa chọn kiến trúc, chuẩn bị dữ liệu, đến quá trình huấn luyện và đánh giá.

2.2.1 Lựa chọn kiến trúc mô hình

Việc lựa chọn kiến trúc mô hình là bước nền tảng, quyết định khả năng và hiệu suất của hệ thống trong việc giải quyết bài toán sinh lời nhạc rap tiếng Việt có kiểm soát. Trong đồ án này, mô hình **GPT-2 (Generative Pre-trained Transformer 2)** [3] đã được lựa chọn làm kiến trúc cơ sở.

a. Giới thiệu chung về GPT-2 (Transformer)

GPT-2 [3] là một mô hình ngôn ngữ lớn (Large Language Model - LLM) được phát triển bởi OpenAI, dựa trên kiến trúc Transformer. Cụ thể hơn, GPT-2 sử dụng kiến trúc Transformer Decoder-only, có nghĩa là nó được thiết kế để thực

hiện việc sinh chuỗi (sequence generation) bằng cách dự đoán từ tiếp theo trong một chuỗi dựa trên các từ đã xuất hiện trước đó.

Đặc điểm nổi bật của kiến trúc Transformer:

- Cơ chế Self-Attention (Tự chú ý): Đây là thành phần cốt lõi giúp Transformer xử lý hiệu quả các mối quan hệ phụ thuộc xa trong chuỗi văn bản. Thay vì phải xử lý tuần tự từng từ như các mạng RNN truyền thống, cơ chế attention cho phép mô hình xem xét toàn bộ ngữ cảnh đầu vào cùng một lúc, gán trọng số khác nhau cho các từ tùy thuộc vào mức độ liên quan của chúng đến từ đang được xử lý. Điều này giúp GPT-2 nắm bắt được ngữ nghĩa và mối liên kết giữa các phần khác nhau của câu hoặc đoạn văn.
- Tính song song hóa cao: Không giống như RNN hay LSTM, kiến trúc Transformer có thể xử lý các phần của chuỗi song song, giúp tăng tốc độ huấn luyện đáng kể trên các bộ dữ liệu lớn.
- Khả năng học biểu diễn ngữ cảnh mạnh mẽ: Với số lượng tham số lớn và khả năng xử lý ngữ cảnh toàn cục, GPT-2 có thể tạo ra các biểu diễn vector phong phú cho các từ và câu, từ đó hiểu sâu sắc hơn về ý nghĩa và ngữ pháp.

b. Lý do lựa chọn GPT-2 cho bài toán sinh lời rap tiếng Việt

Mặc dù có nhiều kiến trúc mô hình ngôn ngữ khác nhau, GPT-2 đã được chọn làm nền tảng cho đồ án này dựa trên các lý do chiến lược và kỹ thuật sau:

- **Khả năng sinh văn bản mạch lạc và tự nhiên**: Nhờ vào kiến trúc Transformer và quá trình tiền huấn luyện trên lượng lớn dữ liệu văn bản, GPT-2 có khả năng tạo ra các câu văn có tính ngữ pháp chính xác, mạch lạc và tự nhiên. Đây là yếu tố tiên quyết để lời rap được sinh ra không chỉ có vần mà còn phải dễ hiểu và có ý nghĩa.
- **Hiệu quả trong việc học ngữ cảnh và tiếp nối câu**: GPT-2 được thiết kế để dự đoán từ tiếp theo dựa trên toàn bộ ngữ cảnh đã cho. Đối với bài toán sinh lời rap, nơi mỗi dòng tiếp theo cần phải liên quan đến dòng trước đó (ngữ cảnh), khả năng này của GPT-2 là cực kỳ phù hợp và hiệu quả.
- **Tận dụng mô hình tiền huấn luyện tiếng Việt (NlpHUST/gpt2-vietnamese)** [4]: Đây là một lợi thế then chốt và là yếu tố phân biệt so với các mô hình nước ngoài như DeepBeat hay LyricJam.
- **Khắc phục rào cản ngôn ngữ**: Thay vì phải huấn luyện một mô hình từ đầu hoặc điều chỉnh mô hình tiếng Anh cho tiếng Việt (vốn rất khó khăn do đặc thù về thanh điệu, ngữ âm, và cấu trúc từ của tiếng Việt), việc sử

dụng phiên bản NlpHUST/gpt2-vietnamese [4] giúp mô hình đã có sẵn "kiến thức" sâu rộng về ngôn ngữ tiếng Việt. Điều này đảm bảo rằng các câu được sinh ra sẽ tuân thủ ngữ pháp tiếng Việt, sử dụng từ vựng phù hợp và có thanh điệu tương đối tự nhiên, giảm thiểu đáng kể các lỗi ngôn ngữ và sự "gượng ép" thường thấy khi áp dụng mô hình đa ngôn ngữ hoặc chỉ được huấn luyện trên tiếng Anh.

- **Tiết kiệm tài nguyên và thời gian huấn luyện**: Quá trình tiền huấn luyện (pre-training) một mô hình ngôn ngữ lớn đòi hỏi lượng dữ liệu khổng lồ và tài nguyên tính toán rất lớn. Việc fine-tune một mô hình đã được pre-trained trên tiếng Việt cho phép tập trung tài nguyên vào việc học các đặc trưng cụ thể của lời rap, thay vì phải xây dựng lại toàn bộ nền tảng ngôn ngữ từ đầu.
- **Khả năng kiểm soát đầu ra qua prompt**: Mặc dù GPT-2 ban đầu được thiết kế cho mục đích sinh văn bản tự do, khả năng của nó có thể được tận dụng để tạo ra văn bản có kiểm soát thông qua các "prompt" (lời nhắc) có cấu trúc. Trong đồ án này, cấu trúc prompt topic:
  <chủ đề> | rhyme: <vần> | <ngữ cảnh>
  đã được thiết kế để định hướng mô hình sinh ra lời rap theo chủ đề và vần cụ thể, đáp ứng trực tiếp yêu cầu của người dùng.

Tóm lại, việc lựa chọn GPT-2, đặc biệt là phiên bản đã được tiền huấn luyện trên tiếng Việt, là một bước đi chiến lược, tận dụng được sức mạnh của các mô hình ngôn ngữ lớn đồng thời giải quyết hiệu quả các thách thức đặc thù của tiếng Việt trong bài toán sinh lời nhạc rap.

2.2.2 Chuẩn bị dữ liệu

Chất lượng và sự phù hợp của dữ liệu là yếu tố then chốt, quyết định trực tiếp đến hiệu suất và khả năng học hỏi của mô hình ngôn ngữ. Để đảm bảo mô hình GPT-2 có thể tinh chỉnh hiệu quả cho bài toán sinh lời nhạc rap tiếng Việt có kiểm soát, quá trình chuẩn bị dữ liệu đã được thực hiện một cách cẩn trọng theo các bước sau:

Nguồn gốc và quy mô dữ liệu

Dữ liệu được sử dụng để huấn luyện, kiểm định và kiểm thử mô hình được thu thập từ các nguồn lời nhạc rap tiếng Việt phổ biến và uy tín. Việc lựa chọn các nguồn này là rất quan trọng để đảm bảo mô hình học được phong cách, từ vựng và cấu trúc ngôn ngữ đặc trưng của thể loại rap trong bối cảnh tiếng Việt.

Các nguồn dữ liệu cụ thể:

- Genius (https://genius.com/): Một nền tảng nổi tiếng toàn cầu về lời bài hát và kiến thức âm nhạc, bao gồm một kho tàng lớn các lời rap tiếng Việt với độ chính xác cao. Dữ liệu từ Genius được thu thập thông qua các công cụ cào dữ liệu (web scraping).
- NhacCuaTui (https://www.nhaccuatui.com/): Một trong những trang web nghe nhạc lớn nhất tại Việt Nam, cung cấp một lượng đáng kể lời bài hát tiếng Việt, bao gồm cả các bài rap. Dữ liệu từ NhacCuaTui cũng được thu thập tương tự thông qua cào dữ liệu.

Quy mô Dữ liệu sau thu thập và tiền xử lý ban đầu: Dữ liệu sau khi loại bỏ các phần không phải lời nhạc (ví dụ: "[Chorus]", "[Verse 1]", dấu thời gian, dòng trống) và các định dạng không liên quan được phân chia thành ba tập riêng biệt:

- Tập huấn luyện (train.txt): Bao gồm 10.804 dòng dữ liệu.
- Tập kiểm định (val.txt): Gồm 1.597 dòng dữ liệu.
- Tập kiểm thử (test.txt): Chứa 1.596 dòng dữ liệu.

Các bước tiền xử lý (preprocessing) và chuẩn hóa dữ liệu

Để tối ưu hóa chất lượng dữ liệu đầu vào cho mô hình học sâu, các bước tiền xử lý đã được áp dụng nhằm làm sạch, chuẩn hóa và định dạng dữ liệu theo cấu trúc phù hợp. Các bước này được thực hiện tuần tự như sau:

(i) Loại bỏ ký tự đặc biệt và làm sạch cơ bản:

Cách thực hiện: Sử dụng các biểu thức chính quy (regular expressions) để loại bỏ các ký tự không phải chữ cái (tiếng Việt có dấu), số, dấu cách và một số dấu câu cơ bản cần thiết (như dấu chấm, dấu phẩy, dấu hỏi, dấu chấm than, và đặc biệt là ký tự | dùng để phân tách trong prompt). Các ký tự dư thừa, emoji, hoặc ký hiệu không liên quan đến lời rap được loại bỏ để giảm nhiễu.

Mục đích: Đảm bảo dữ liệu chỉ chứa các thành phần ngôn ngữ cần thiết, giúp mô hình tập trung học các mẫu chính xác.

(ii) Sửa lỗi chính tả:

Cách thực hiện: Áp dụng các quy tắc tiền định và sử dụng một từ điển hiệu chỉnh lỗi chính tả phổ biến trong tiếng Việt. Quá trình này có thể bao gồm việc chuẩn hóa các cách viết tắt, viết hoa không đúng quy tắc, hoặc các lỗi đánh máy thường gặp.

Mục đích: Nâng cao chất lượng ngữ pháp và từ vựng của dữ liệu huấn luyện, từ đó giúp mô hình sinh ra lời rap chính xác và tự nhiên hơn.

(iii) Loại bỏ các câu có chứa ngôn ngữ không phải tiếng Việt:

Cách thực hiện: Đối với mỗi dòng dữ liệu, sử dụng các thư viện nhận diện ngôn ngữ (ví dụ: langdetect hoặc các mô hình nhận diện ngôn ngữ dựa trên fastText) để xác định ngôn ngữ chính. Nếu một dòng bị phát hiện chứa phần lớn ngôn ngữ không phải tiếng Việt, hoặc nếu nó chứa các ký tự/cấu trúc rõ ràng của ngôn ngữ khác mà không phải là một phần của "code-switching" có chủ đích trong rap Việt (ví dụ: câu tiếng Anh hoàn chỉnh), dòng đó sẽ bị loại bỏ. Ngoài ra, có thể áp dụng bộ lọc dựa trên tập hợp ký tự tiếng Việt chuẩn.

Mục đích: Đảm bảo tính thuần Việt của bộ dữ liệu, ngăn chặn mô hình học các cấu trúc hoặc từ vựng không mong muốn từ các ngôn ngữ khác, từ đó giúp lời rap đầu ra luôn là tiếng Việt.

(iv) Gán nhãn Chủ đề (Topic) và Vần (Rhyme):

Cách thực hiện: Đây là một trong những bước quan trọng nhất để tạo ra dữ liệu huấn luyện cho mô hình sinh lời có kiểm soát.

Gán nhãn Chủ đề: Mỗi bài rap hoặc đoạn rap có thể được gán một chủ đề tổng thể (ví dụ: "tình yêu", "cuộc sống", "xã hội", "gia đình"). Việc gán nhãn này có thể được thực hiện bán tự động (dựa trên metadata của bài hát) hoặc thủ công để đảm bảo độ chính xác cao nhất về ngữ cảnh nội dung.

Gán nhãn Vần: Đối với mỗi dòng lời rap, từ cuối cùng (hoặc nhóm âm tiết cuối cùng) được phân tích để xác định vần. Quá trình này yêu cầu sử dụng một bộ từ điển vần tiếng Việt hoặc một thuật toán phân tích ngữ âm tiếng Việt để trích xuất các âm tiết vần. Kho dữ liệu từ vựng đã thu thập lên đến 29.515 từ (bao gồm cả ngôn ngữ toàn dân, nội ngữ và từ ngữ địa phương) đóng vai trò là cơ sở để nhận diện và chuẩn hóa các vần này. Mỗi từ vần được gắn với một nhãn vần cụ thể (ví dụ: "an", "anh").

Mục đích: Cung cấp thông tin kiểm soát chi tiết cho mô hình, giúp nó học cách sinh lời theo đúng chủ đề và vần được yêu cầu.

(v) Định dạng dữ liệu huấn luyện thành cấu trúc Prompt + Target:

Cách thực hiện: Sau khi các dòng lời rap đã được làm sạch và gán nhãn chủ đề/vần, chúng được chuyển đổi thành định dạng đặc trưng mà mô hình GPT-2 sẽ học. Đối với mỗi cặp dòng liên tiếp trong một bài rap (L_i và L_{i+1}):

- Input Prompt (Đầu vào cho mô hình): Được xây dựng theo cấu trúc topic: *<Chủ đề bài rap> | rhyme: <Vần của từ cuối cùng của L_{i+1}> | <L_i>*
- Target Output (Đầu ra mong muốn): Là dòng lời L_{i+1}
- Ví dụ: Nếu trong bài rap có dòng "Cha mẹ cho con bao nhiêu yêu thương chân thành" (L_i) và dòng tiếp theo là "Mỗi khi mẹ buồn, con nguyện sẽ

luôn bên cạnh" (L_{i+1}), với chủ đề là "gia đình" và từ "bên cạnh" có vần là "anh", thì mẫu huấn luyện sẽ được định dạng như sau:

Input: topic: gia đình | rhyme: anh | Cha mẹ cho con bao nhiêu yêu thương chân thành

Target: Mỗi khi mẹ buồn, con nguyện sẽ luôn bên cạnh

– Đối với quá trình fine-tuning GPT-2, mô hình sẽ nhận toàn bộ chuỗi Input Prompt + Target Output và được huấn luyện để dự đoán các token của Target Output dựa trên Input Prompt.

Mục đích: Huấn luyện mô hình học mối quan hệ giữa các tham số điều khiển (chủ đề, vần) và ngữ cảnh (dòng rap trước đó) để sinh ra dòng lời rap mong muốn.

Quá trình chuẩn bị dữ liệu kỹ lưỡng và có cấu trúc này là nền tảng vững chắc để mô hình GPT-2 có thể học được các đặc trưng phức tạp của lời nhạc rap tiếng Việt và thực hiện được các chức năng sinh lời có kiểm soát theo yêu cầu của đồ án.

2.2.3 Huấn luyện mô hình (fine-tuning)

Sau khi kiến trúc mô hình đã được lựa chọn (GPT-2) và bộ dữ liệu đã được chuẩn bị kỹ lưỡng, bước tiếp theo là huấn luyện mô hình. Thay vì huấn luyện từ đầu (training from scratch), phương pháp fine-tuning (tinh chỉnh) được áp dụng. Điều này có nghĩa là một mô hình GPT-2 [3] đã được tiền huấn luyện trên một lượng lớn dữ liệu tiếng Việt (NlpHUST/gpt2-vietnamese [4]) sẽ được điều chỉnh các tham số trên tập dữ liệu lời nhạc rap tiếng Việt đã chuẩn bị. Phương pháp này giúp tận dụng kiến thức ngôn ngữ rộng lớn mà mô hình đã học được, đồng thời cho phép nó thích nghi với phong cách, cấu trúc và ngữ vựng đặc trưng của lời rap.

Quá trình huấn luyện được thực hiện với các cấu hình và kỹ thuật sau:

Cấu hình huấn luyện

– Số lượng Epoch (Epochs): Mô hình được huấn luyện trong 25 epochs. Một epoch là một lượt mà toàn bộ tập dữ liệu huấn luyện được đưa qua mô hình một lần. Việc chạy nhiều epoch cho phép mô hình nhìn thấy dữ liệu nhiều lần và liên tục cập nhật các trọng số của mình.

– Chiến lược dừng sớm (Early Stopping): Để ngăn chặn hiện tượng quá khớp (overfitting) – khi mô hình học quá sâu vào dữ liệu huấn luyện mà mất khả năng tổng quát hóa trên dữ liệu mới – kỹ thuật dừng sớm đã được triển khai. Quá trình huấn luyện sẽ tự động dừng lại nếu hiệu suất của mô hình trên tập kiểm định (validation set) không có sự cải thiện đáng kể sau một số lượng epoch nhất định.

Điều này giúp tối ưu hóa thời gian huấn luyện và đảm bảo mô hình có khả năng tổng quát hóa tốt nhất.

- Thuật toán tối ưu (Optimizer): Thuật toán AdamW được sử dụng để cập nhật các trọng số của mô hình trong suốt quá trình huấn luyện. AdamW [5] là một biến thể của thuật toán Adam, được thiết kế để xử lý hiệu quả vấn đề suy giảm trọng số (weight decay) trong các mô hình học sâu, đặc biệt là với kiến trúc Transformer. Việc sử dụng AdamW thường dẫn đến hiệu suất tốt hơn và hội tụ nhanh hơn so với các thuật toán tối ưu khác.
- Kích thước Batch (Batch Size): Kích thước batch được đặt là 4. Điều này có nghĩa là mỗi lần cập nhật trọng số, mô hình sẽ xử lý 4 mẫu dữ liệu cùng một lúc. Kích thước batch nhỏ có thể giúp mô hình thoát khỏi các điểm cực tiểu cục bộ (local minima) tốt hơn và đôi khi dẫn đến khả năng tổng quát hóa tốt hơn, mặc dù có thể làm tăng thời gian huấn luyện tổng thể.
- Độ dài tối đa của chuỗi (Max Length): Độ dài tối đa của chuỗi đầu vào (bao gồm cả prompt) và đầu ra được giới hạn ở 128 tokens. Việc giới hạn độ dài này giúp kiểm soát tài nguyên bộ nhớ và thời gian tính toán, đồng thời phù hợp với độ dài thông thường của một dòng lời rap.

Theo dõi và đánh giá trong quá trình huấn luyện

Trong suốt quá trình fine-tuning, hiệu suất của mô hình được theo dõi liên tục thông qua các chỉ số sau:

- Hàm mất mát (Loss Function): Giá trị của hàm mất mát (thường là Cross-Entropy Loss cho bài toán dự đoán token tiếp theo) được theo dõi qua từng epoch trên cả tập huấn luyện và tập kiểm định. Hàm mất mát đo lường sự khác biệt giữa đầu ra dự đoán của mô hình và đầu ra thực tế. Mục tiêu là giá trị loss giảm dần theo thời gian, cho thấy mô hình đang học hiệu quả và các dự đoán của nó ngày càng chính xác hơn.
- Perplexity (Độ phức tạp): Chỉ số Perplexity cũng được theo dõi chặt chẽ, đặc biệt là trên tập kiểm định. Perplexity là một thước đo đánh giá mức độ chắc chắn của mô hình ngôn ngữ khi dự đoán từ tiếp theo trong một chuỗi. Giá trị Perplexity càng thấp, chứng tỏ mô hình càng tự tin và chính xác trong việc mô hình hóa phân phối xác suất của dữ liệu, từ đó sinh ra văn bản tự nhiên và hợp lý hơn.

Lưu trữ và quản lý mô hình

- Checkpointing: Trong quá trình huấn luyện, trạng thái của mô hình (các trọng số đã học) được lưu lại định kỳ dưới dạng các checkpoint (điểm kiểm tra). Điều này cho phép:
- Phục hồi huấn luyện: Trong trường hợp quá trình huấn luyện bị gián đoạn, có thể tiếp tục từ checkpoint gần nhất mà không phải bắt đầu lại từ đầu.
- Chọn mô hình tốt nhất: Sau khi quá trình huấn luyện kết thúc (có thể do dừng sớm), có thể chọn phiên bản mô hình có hiệu suất tốt nhất trên tập kiểm định (thường là phiên bản có perplexity thấp nhất) để sử dụng cho giai đoạn kiểm thử và triển khai.

Quá trình fine-tuning được kiểm soát chặt chẽ với các tham số và kỹ thuật này nhằm đảm bảo mô hình GPT-2 không chỉ học được cách sinh ra lời rap có vần và chủ đề mà còn có khả năng tổng quát hóa tốt trên dữ liệu mới, mang lại hiệu suất ổn định khi ứng dụng thực tế.

2.2.4 Nguyên lý hoạt động của mô hình đã được fine-tune

Sau khi hoàn tất quá trình fine-tuning (tinh chỉnh) trên bộ dữ liệu lời nhạc rap tiếng Việt, mô hình GPT-2 đã được trang bị khả năng hiểu và sinh ra văn bản tuân thủ theo các ràng buộc về chủ đề, vần điệu và ngữ cảnh. Phần này sẽ đi sâu vào từng bước hoạt động của mô hình khi nhận đầu vào từ người dùng để sinh ra lời rap mới.

Cơ chế tiếp nhận đầu vào (Prompt Engineering)

Giao diện người dùng sẽ thu thập thông tin từ người dùng và xây dựng thành một chuỗi văn bản đầu vào duy nhất, được gọi là prompt. Cấu trúc của prompt là chìa khóa để điều khiển đầu ra của mô hình:

topic: <chủ đề> | rhyme: <vần> | <ngữ cảnh>

Trong đó:

- topic: <chủ đề>:
    - Mục đích: Hướng dẫn mô hình về nội dung ngữ nghĩa chính của dòng lời rap cần sinh ra.
    - Cách thức: Người dùng nhập một từ hoặc cụm từ khóa đại diện cho chủ đề mong muốn (ví dụ: "tình yêu", "gia đình", "xã hội", "đời sống"). Mô hình, thông qua quá trình fine-tuning, đã học được cách liên kết các từ vựng, cụm từ và ngữ cảnh liên quan đến từng chủ đề này.
- rhyme: <vần>:
    - Mục đích: Ràng buộc âm thanh cuối cùng của dòng lời rap được sinh ra.

- Cách thức: Người dùng cung cấp một từ vần mong muốn (ví dụ: "an", "anh", "ay"). Mô hình sẽ tập trung vào việc tạo ra các từ khóa ở cuối câu có cấu trúc âm thanh tương đồng với vần được chỉ định. Đây là một thách thức đối với tiếng Việt do đặc tính thanh điệu, nhưng mô hình đã được huấn luyện để nhận diện các nhóm âm vần phổ biến.

– <ngữ cảnh>:
  - Mục đích: Cung cấp bối cảnh ngữ nghĩa cho dòng lời rap mới, đảm bảo tính mạch lạc và liên kết câu.
  - Cách thức: Người dùng nhập một dòng lời rap trước đó (hoặc bất kỳ dòng nào họ muốn tiếp nối). Mô hình sẽ phân tích ý nghĩa, cấu trúc ngữ pháp và mối quan hệ giữa các từ trong dòng ngữ cảnh này.
  - Xử lý đa dòng ngữ cảnh: Nếu người dùng cung cấp nhiều hơn một dòng ngữ cảnh (ví dụ: một đoạn rap gồm vài câu), hệ thống sẽ chỉ trích xuất và sử dụng dòng ngữ cảnh cuối cùng làm input cho mô hình. Điều này giúp tối ưu hóa độ dài input, giảm thiểu nhiễu và tập trung vào mối liên kết chặt chẽ nhất giữa hai dòng liên tiếp.

Quá trình sinh lời (Text Generation)

Khi prompt đã được xây dựng, nó sẽ được đưa vào mô hình GPT-2 đã được fine-tune để sinh lời theo các bước sau:

(i) Mã hóa đầu vào (Tokenization and Embedding):

– **Tokenization:** Chuỗi prompt (*topic: <chủ đề> | rhyme: <vần> | <ngữ cảnh>*) được xử lý bởi một bộ mã hóa (tokenizer) đã được huấn luyện sẵn cho tiếng Việt (thường là BPE - Byte-Pair Encoding hoặc WordPiece). Bộ mã hóa này sẽ chia chuỗi thành các đơn vị nhỏ hơn gọi là token.
Ví dụ: "Anh đã mất quá nhiều thời gian" có thể được chia thành các token như [*'Anh', 'đã', 'mất', 'quá', 'nhiều', 'thời', 'gian'*].
Các token đặc biệt (như topic:, rhyme:, |) cũng được xử lý như các token riêng biệt.

– **Embedding:** Mỗi token sau đó được chuyển đổi thành một vector số (embedding). Vector này là một biểu diễn mật độ cao, nắm bắt ý nghĩa ngữ cảnh và ngữ pháp của token trong không gian chiều cao. Vị trí tương đối của token trong chuỗi (positional encoding) cũng được thêm vào các embeddings này.

(ii) Truyền qua kiến trúc Transformer Decoder:

Các vector embeddings của toàn bộ chuỗi prompt (bao gồm cả *topic:, rhyme:,* và *<ngữ cảnh>*) được đưa vào các lớp Transformer Decoder của GPT-2.

Mỗi lớp Decoder bao gồm các khối Self-Attention (Tự Chú ý) và Feed-Forward Networks. Trong đó:

- Self-Attention: Đây là cơ chế cốt lõi. Tại mỗi lớp, Self-Attention cho phép mô hình đánh giá mức độ liên quan giữa mọi token trong chuỗi prompt với nhau. Ví dụ, khi xử lý token *gian* (trong "thời gian"), cơ chế Self-Attention sẽ gán trọng số cao hơn cho các token như *thời*, *mất*, *thời gian* để hiểu rõ hơn ngữ cảnh. Đặc biệt, nó sẽ chú ý đến các token *topic: <chủ đề>* và *rhyme: <vần>* để định hướng việc sinh lời.
- Masked Self-Attention: Trong pha sinh lời, cơ chế này chỉ cho phép một token nhìn thấy các token đứng trước nó trong chuỗi (và không thể nhìn thấy các token sẽ được sinh ra), đảm bảo tính tuần tự của quá trình dự đoán.

Kết quả của các lớp Decoder là các biểu diễn ngữ cảnh được làm giàu cho từng token.

(iii) Dự đoán token tiếp theo (Probability Distribution):

Ở lớp đầu ra cuối cùng, mô hình sẽ tính toán một phân phối xác suất trên toàn bộ kho từ vựng đã học cho mỗi vị trí token tiếp theo cần được sinh ra. Phân phối này chỉ ra xác suất của từng từ có thể xuất hiện sau chuỗi prompt hiện tại.

Ví dụ: Sau prompt topic: gia đình | rhyme: anh | Anh đã mất quá nhiều thời gian, mô hình có thể gán xác suất cao cho các từ như "bên", "cạnh", "mỗi", v.v.

(iv) Lựa chọn token và sinh lời lặp đi lặp lại:

Từ phân phối xác suất này, mô hình sẽ lựa chọn token tiếp theo. Các phương pháp lựa chọn phổ biến bao gồm:

- Tham lam (Greedy Search): Luôn chọn token có xác suất cao nhất.
- Lấy mẫu ngẫu nhiên (Sampling): Chọn một token dựa trên phân phối xác suất của nó (ví dụ: Top-K sampling hoặc Nucleus sampling) để tăng tính ngẫu nhiên và sáng tạo.

Token được chọn sau đó được thêm vào chuỗi prompt hiện tại, và toàn bộ quá trình (từ bước 1 đến bước 4) được lặp lại.

Quá trình này tiếp tục cho đến khi đạt được số lượng token mong muốn cho một dòng lời rap hoặc khi mô hình sinh ra một token kết thúc câu/dòng (<EOS> - End Of Sentence).

Mục tiêu đầu ra của mô hình và kết nối với user prompt

Mục tiêu cuối cùng của mô hình đã được fine-tune là cung cấp cho người dùng một dòng lời rap mới đáp ứng các tiêu chí sau:

- Tính liên kết ngữ cảnh: Dòng lời mới phải duy trì ý nghĩa và luồng tư tưởng từ dòng <ngữ cảnh> đã cung cấp. Mô hình đã học được cách tạo ra sự chuyển tiếp logic giữa các câu.
- Phù hợp chủ đề: Nội dung của dòng lời sinh ra phải nhất quán với <chủ đề> đã được chỉ định, thể hiện qua việc sử dụng từ ngữ và ý tưởng liên quan.
- Đảm bảo vần điệu: Mô hình sẽ nỗ lực để từ cuối cùng của dòng lời sinh ra có vần với <vần> đã nhập. Mặc dù sự phức tạp của tiếng Việt (thanh điệu, vần đa dạng) có thể khiến việc đạt được vần hoàn hảo 100% là thách thức, mô hình được tối ưu hóa để đưa ra các lựa chọn tốt nhất trong khả năng của nó.

Nhờ vào kiến trúc Transformer mạnh mẽ của GPT-2 và quá trình tinh chỉnh chuyên biệt trên dữ liệu rap tiếng Việt có gắn nhãn, mô hình có thể hoạt động như một công cụ hỗ trợ sáng tạo mạnh mẽ, giúp người dùng vượt qua những rào cản trong việc tìm vần, duy trì chủ đề và tạo ra lời rap mạch lạc, có tính nghệ thuật.

2.3 Thiết kế hệ thống

2.3.1 Các tác nhân

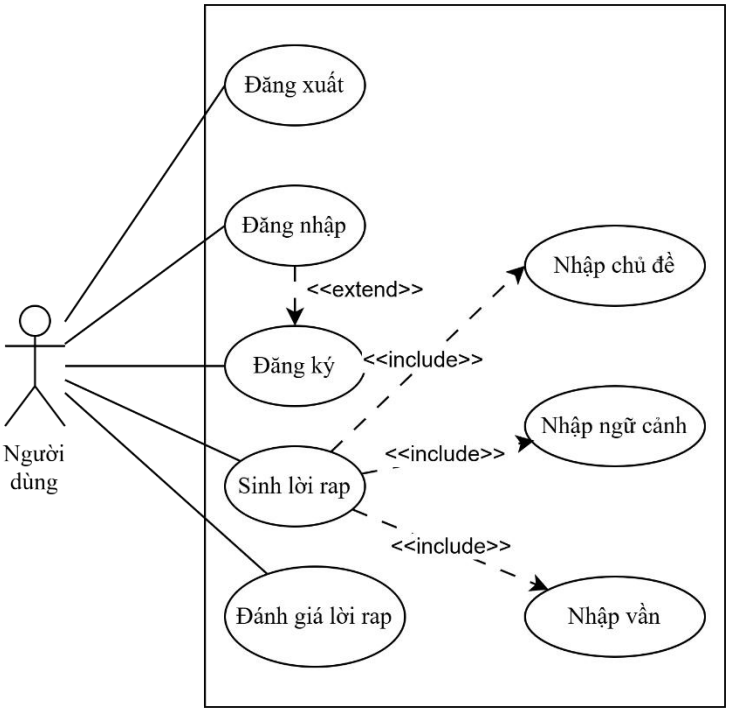
Bảng dưới đây mô tả tất cả các tác nhân tham gia vào hệ thống.

Bảng 2.1. Bảng mô tả các tác nhân

Tác nhân	Mô tả
Người dùng	– Đăng ký tài khoản, đăng nhập/đăng xuất – Xem danh sách các từ cùng vần phù hợp với ngữ cảnh – Xem danh sách các từ vựng tiếng Việt – Sinh lời nhạc rap dựa trên chủ đề, vần và ngữ cảnh – Đánh giá chất lượng lời rap
Quản trị viên	– Quyền quản trị đối với người dùng – Quyền quản trị đối với danh sách các từ vựng – Quyền quản trị đối với lịch sử, các đánh giá kết quả sinh lời – Tìm kiếm từ vựng theo vần, từ – Thêm từ vựng mới

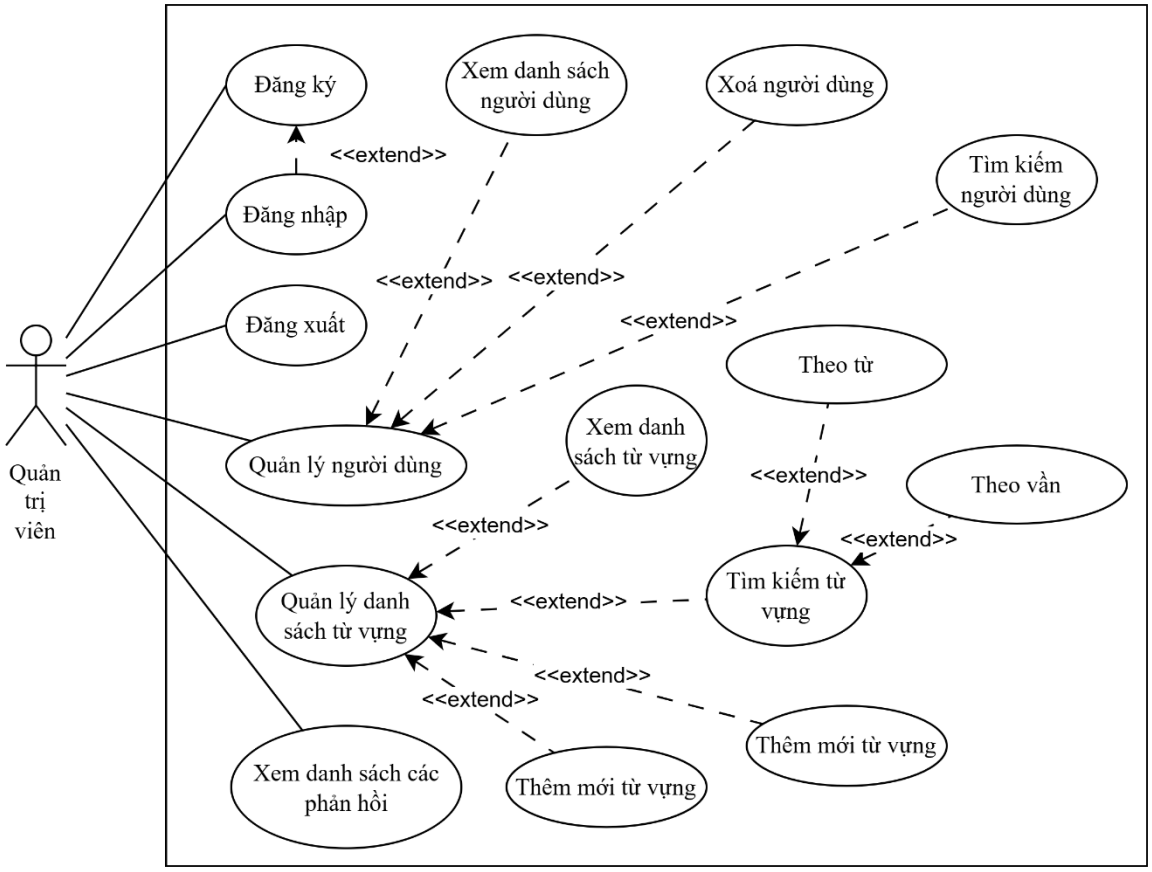
2.3.2 Use-case diagrams

Sơ đồ ca sử dụng của người dùng: miêu tả các chức năng người dùng (có tài khoản) có thể sử dụng trên website



Hình 2.1. Sơ đồ use-case các chức năng của người dùng (có tài khoản)

Sơ đồ ca sử dụng của quản trị viên: miêu tả các chức năng quản trị viên có thể sử dụng trên website

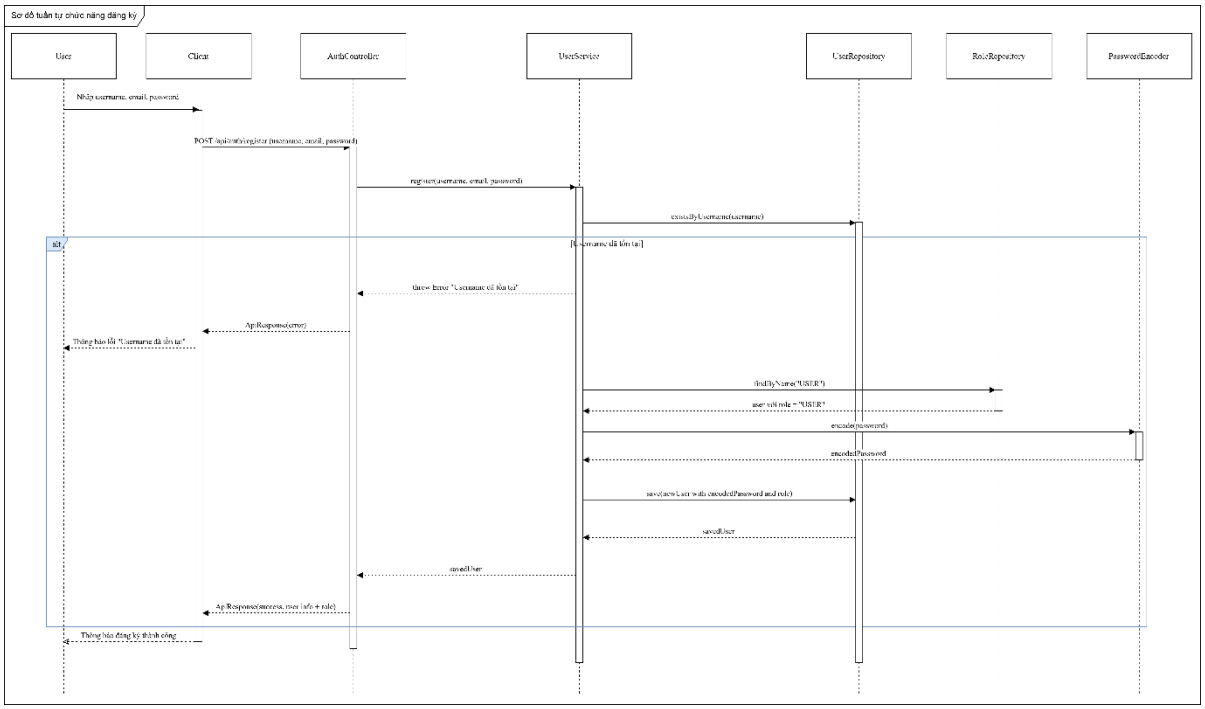


Hình 2.2. Sơ đồ use-case các chức năng của quản trị viên

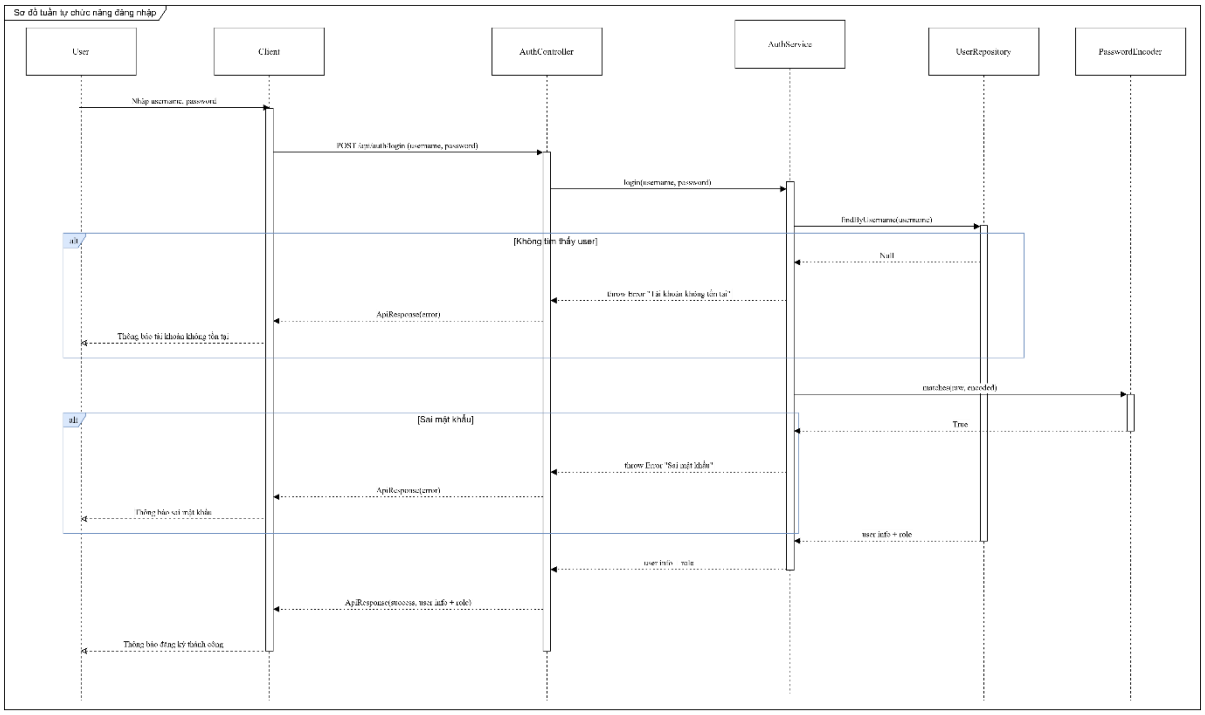
2.3.3 Sơ đồ tuần tự

Sau đây là các sơ đồ tuần tự cho một vài chức năng chính của website.

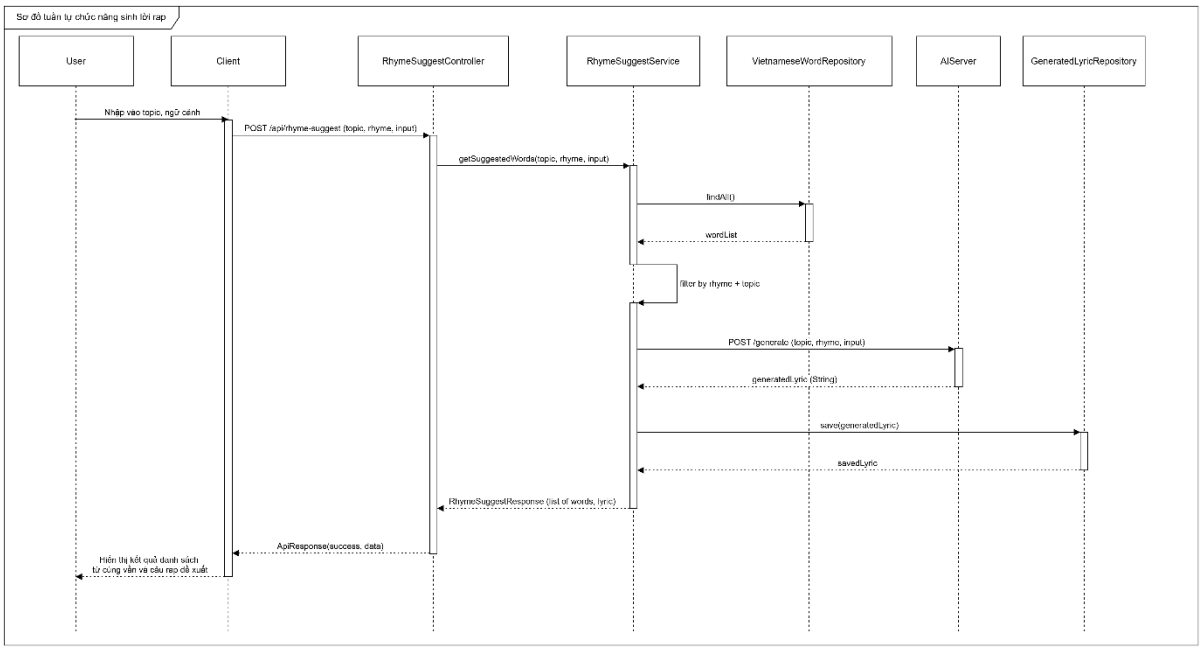
Sơ đồ tuần tự miêu tả luồng hoạt động của các chức năng cơ bản tài khoản mà người dùng có thể thực hiện:



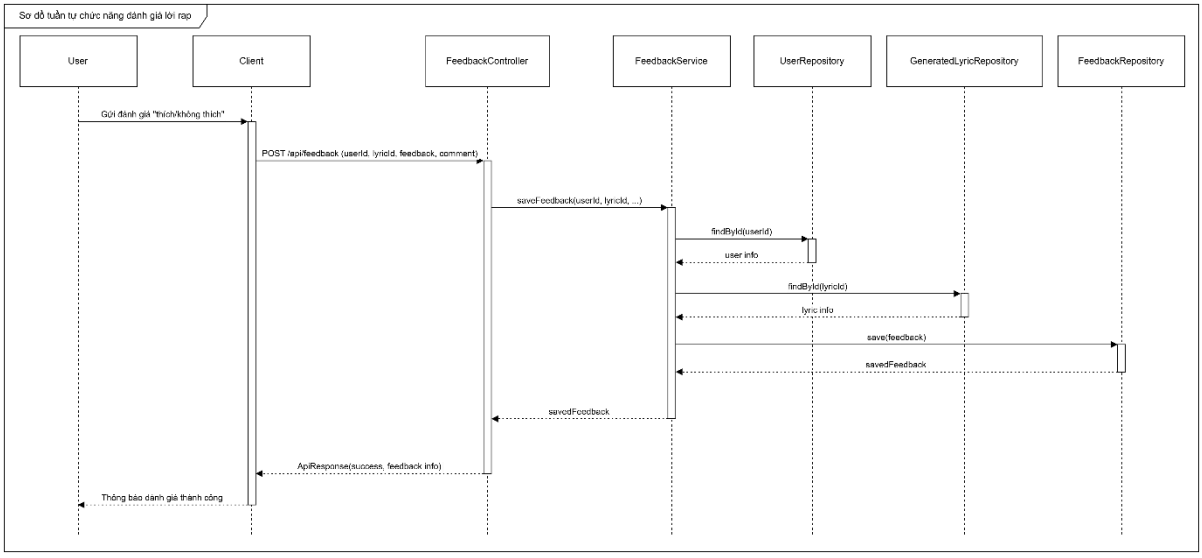
Hình 2.3. Sơ đồ tuần tự chức năng đăng ký tài khoản người dùng



Hình 2.4. Sơ đồ tuần tự chức năng đăng nhập tài khoản người dùng



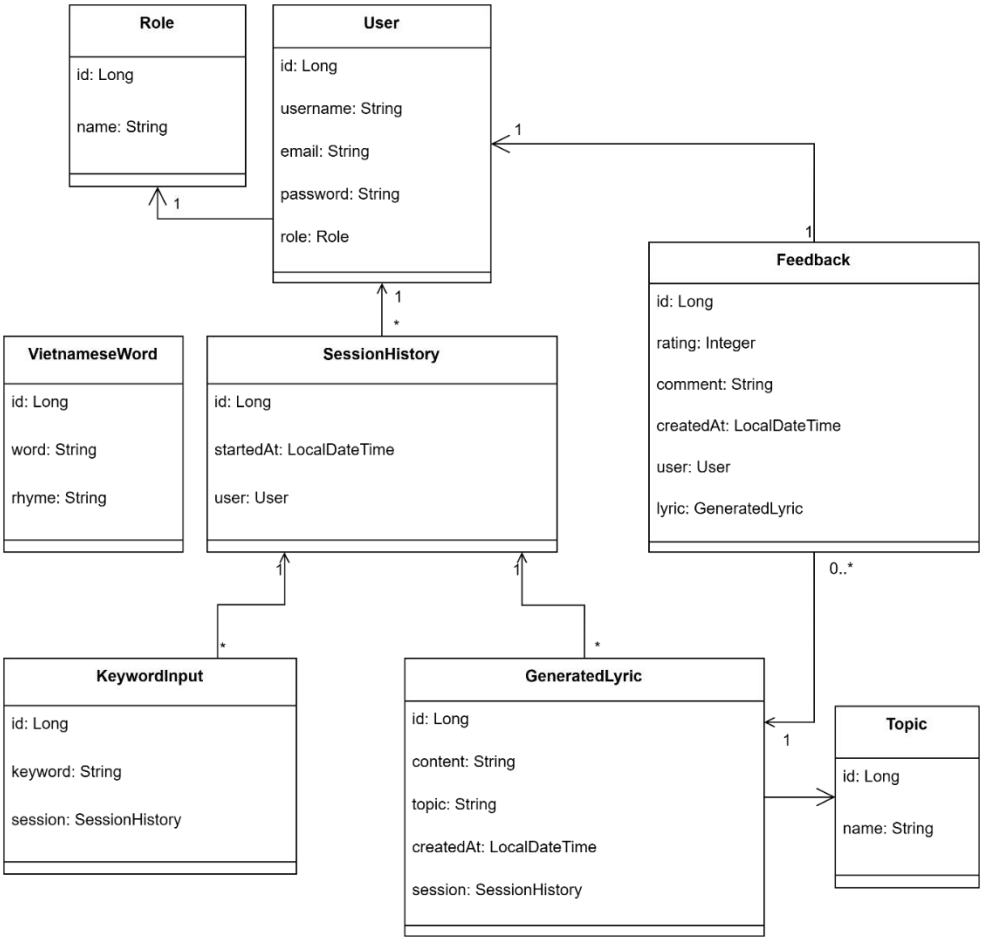
Hình 2.5. Sơ đồ tuần tự chức năng sinh lời rap



Hình 2.6. Sơ đồ tuần tự chức năng đánh giá lời rap

2.4 Thiết kế cơ sở dữ liệu

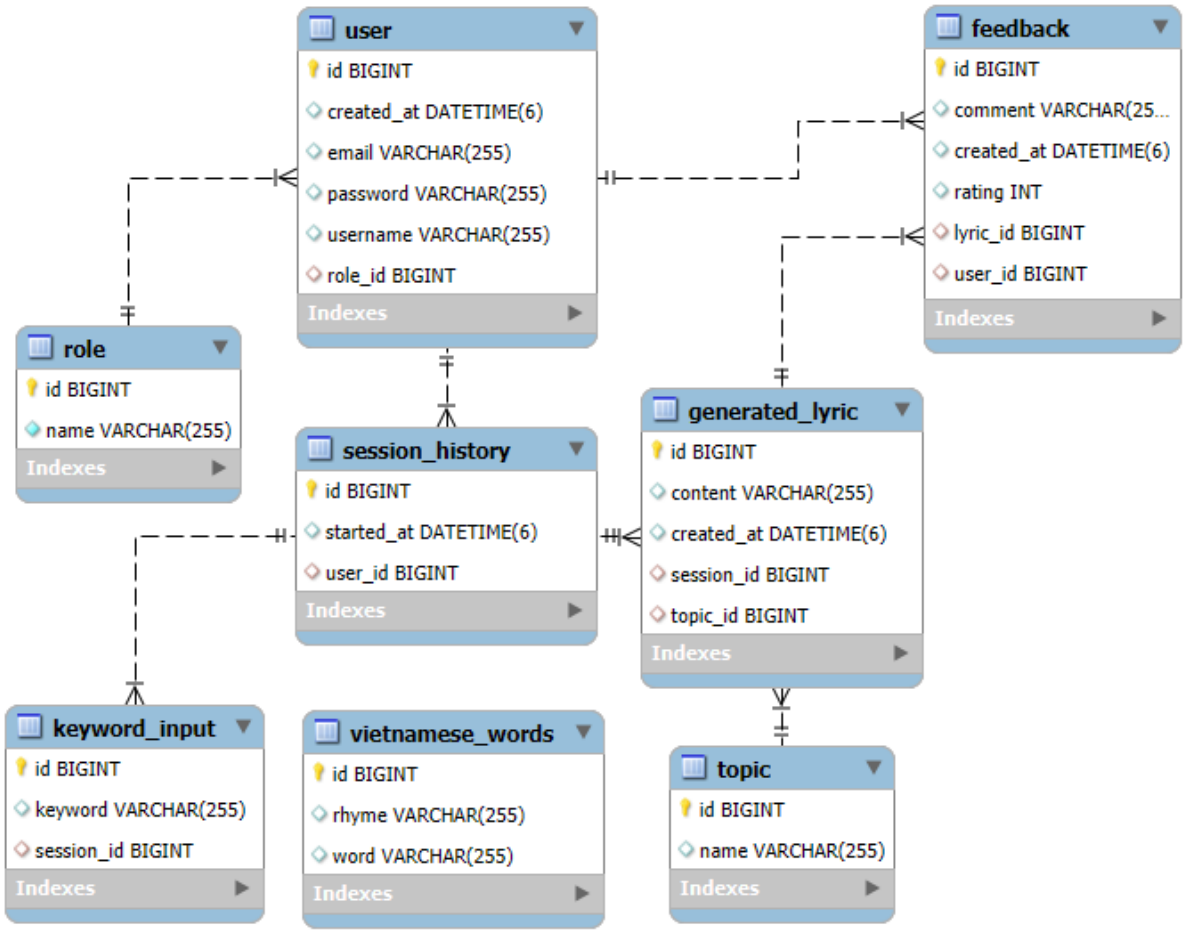
2.4.1 Sơ đồ lớp



Hình 2.7. Sơ đồ lớp tổng quan hệ thống

Chú thích: Lớp VietnameseWord là một thành phần dữ liệu độc lập, dùng để tra cứu từ và vần. Không có quan hệ trực tiếp đến các bảng khác nhưng được sử dụng trong xử lý logic (ví dụ: gợi ý từ cùng vần).

2.4.2 Cơ sở dữ liệu



Hình 2.8. Lược đồ EER

Bảng 2.2. Bảng user – Bảng thông tin của người dùng

Tên bảng	**user**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã người dùng
created_at	DATETIME(6)		Thời điểm tạo
username	VARCHAR(255)		Tên đăng nhập
email	VARCHAR(255)		Địa chỉ email
password	VARCHAR(255)		Mật khẩu
role_id	BIGINT	Khoá ngoại	Mã vai trò

Bảng 2.3. Bảng role – Bảng thông tin các vai trò của người dùng

Tên bảng	**role**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã vai trò
name	VARCHAR(255)		Tên vai trò

Bảng 2.4. Bảng session_history – Bảng thông tin các phiên làm việc

Tên bảng	**session_history**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã phiên làm việc
started_at	DATETIME(6)		Thời điểm bắt đầu
user_id	BIGINT	Khoá ngoại	Mã người dùng

Bảng 2.5. Bảng keyword_input – Bảng thông tin các từ khoá người dùng nhập vào

Tên bảng	**keyword_input**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã từ khoá
keyword	VARCHAR(255)		Từ khoá
session_id	BIGINT	Khoá ngoại	Mã phiên làm việc

Bảng 2.6. Bảng topic – Bảng thông tin các chủ đề

Tên bảng	**topic**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã chủ đề
name	VARCHAR(255)		Tên chủ đề

Bảng 2.7. Bảng generated_lyric – Bảng thông tin các lời nhạc đề xuất bởi hệ thống

Tên bảng	**generated_lyric**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã lời nhạc
content	VARCHAR(255)		Nội dung
created_at	DATETIME(6)		Thời điểm tạo
session_id	BIGINT	Khoá ngoại	Mã phiên làm việc
topic_id	BIGINT	Khoá ngoại	Mã chủ đề

Bảng 2.8. Bảng feedback – Bảng thông tin các đánh giá của người dùng về lời nhạc

Tên bảng	**feedback**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã phản hồi
comment	VARCHAR(255)		Nội dung
created_at	DATETIME(6)		Thời điểm tạo
rating	INT		Đánh giá
lyric_id	BIGINT	Khoá ngoại	Mã lời nhạc
user_id	BIGINT	Khoá ngoại	Mã người dùng

Bảng 2.9. Bảng vietnamese_words – Bảng thông tin các từ vựng tiếng Việt

Tên bảng	**vietnamese_words**		
Tên trường	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
id	BIGINT	Khoá chính	Mã từ
word	VARCHAR(255)		Từ
rhyme	VARCHAR(255)		Vần

2.5 Kết chương

Chương 2 đã hoàn thành việc thiết kế chi tiết hệ thống sinh lời nhạc rap tiếng Việt, bắt đầu từ việc phân tích yêu cầu và đặt ra mục tiêu rõ ràng. Chúng tôi đã xây dựng nền tảng AI bằng cách lựa chọn, tinh chỉnh mô hình GPT-2 trên bộ dữ liệu lời rap tiếng Việt, và trình bày cụ thể nguyên lý hoạt động của nó trong việc sinh ra lời nhạc theo chủ đề, vần điệu và ngữ cảnh yêu cầu. Đồng thời, chương này cũng đã hoàn thiện thiết kế kiến trúc hệ thống, xác định rõ các tác nhân (Người dùng, Quản trị viên) cùng với các use case và nhiệm vụ của họ. Cuối cùng, các sơ đồ tuần tự đã được xây dựng để minh họa chi tiết luồng tương tác giữa các thành phần hệ thống cho các chức năng cốt lõi như đăng ký, đăng nhập, sinh lời rap và đánh giá lời rap.

Chương 3: TRIỂN KHAI HỆ THỐNG

3.1 Công cụ và môi trường phát triển

Trong quá trình xây dựng hệ thống sinh lời nhạc rap tiếng Việt, việc lựa chọn và thiết lập một môi trường phát triển mạnh mẽ, linh hoạt và phù hợp là yếu tố then chốt quyết định đến hiệu suất, khả năng mở rộng, và tính dễ bảo trì của ứng dụng. Phần này sẽ cung cấp cái nhìn tổng quan toàn diện về triết lý lựa chọn công nghệ, cũng như giới thiệu về các thành phần chính trong kiến trúc đa tầng của hệ thống, bao gồm Frontend, Backend và AI Server. Mục tiêu là trình bày một cách chi tiết và dễ hiểu, nêu bật ưu điểm và lý do sử dụng từng công nghệ cụ thể trong đồ án này, đồng thời đảm bảo thông tin được cập nhật và chính xác theo xu hướng công nghệ hiện đại.

Việc thiết kế và triển khai một hệ thống phức tạp như ứng dụng sinh lời nhạc rap tiếng Việt đòi hỏi một chiến lược công nghệ rõ ràng, phân tách các nhiệm vụ và tận dụng thế mạnh của từng công nghệ chuyên biệt. Hệ thống này được xây dựng trên một kiến trúc phân tán, đặc trưng bởi sự phân chia thành các lớp (tiers) hoặc dịch vụ (services) độc lập nhưng có khả năng tương tác mượt mà với nhau. Triết lý này, thường được gọi là kiến trúc microservice hoặc kiến trúc đa tầng, mang lại nhiều lợi ích vượt trội so với mô hình nguyên khối (monolithic) truyền thống, đặc biệt là trong bối cảnh các ứng dụng hiện đại yêu cầu tính linh hoạt, khả năng mở rộng và hiệu suất cao.

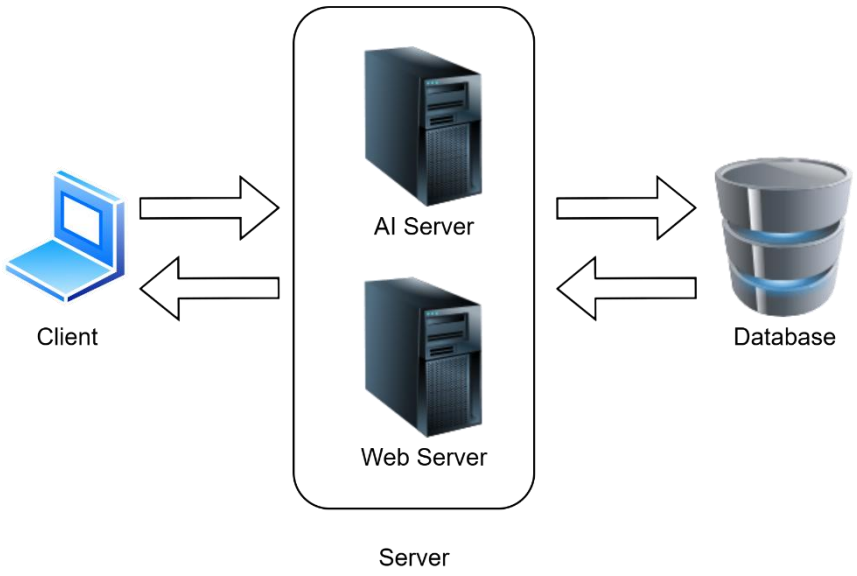
3.1.1 Giới thiệu chung về kiến trúc hệ thống

Hệ thống sinh lời nhạc rap tiếng Việt được hình thành dựa trên kiến trúc ba tầng (Three-tier Architecture) cơ bản, một mô hình thiết kế phần mềm đã được chứng minh hiệu quả trong việc quản lý độ phức tạp và thúc đẩy khả năng mở rộng. Trong kiến trúc này, các chức năng của ứng dụng được phân chia một cách logic và vật lý thành ba lớp chính, mỗi lớp đảm nhiệm một vai trò cụ thể:

- Lớp Giao diện Người dùng (Frontend / Presentation Layer): Đây là phần tương tác trực tiếp với người dùng cuối. Nó chịu trách nhiệm hiển thị thông tin, thu thập đầu vào từ người dùng và trình bày kết quả một cách trực quan, thân thiện. Trong ngữ cảnh của đồ án này, lớp Frontend là giao diện web mà người dùng truy cập để tương tác với các tính năng của ứng dụng.
- Lớp Logic Nghiệp vụ (Backend / Application Layer): Lớp này chứa toàn bộ logic nghiệp vụ cốt lõi của ứng dụng. Nó xử lý các yêu cầu từ lớp Frontend, thực hiện các quy tắc nghiệp vụ, tương tác với cơ sở dữ liệu để truy xuất hoặc lưu trữ dữ liệu, và giao tiếp với các dịch vụ bên ngoài (trong trường hợp này là AI Server). Lớp Backend đóng vai trò như "bộ não" của hệ thống, điều phối mọi hoạt động.

– Lớp Dữ liệu (Data Layer): Lớp này chịu trách nhiệm quản lý và lưu trữ dữ liệu của hệ thống. Nó cung cấp các giao diện để lớp Logic Nghiệp vụ có thể truy cập, thao tác với dữ liệu một cách an toàn và hiệu quả. Trong đồ án này, Cơ sở dữ liệu sẽ lưu trữ thông tin người dùng, lịch sử sinh lời, các đánh giá và có thể là kho từ vựng.

Ngoài ba lớp truyền thống, một lớp đặc biệt và cực kỳ quan trọng trong đồ án này là Lớp Xử lý AI (AI Server / Machine Learning Service). Mặc dù có thể được coi là một dịch vụ bên ngoài đối với lớp Backend chính, nhưng nó là thành phần không thể thiếu, chịu trách nhiệm trực tiếp trong việc thực hiện chức năng sinh lời nhạc rap. Sự tách biệt này cho phép tối ưu hóa từng thành phần độc lập, nâng cao hiệu suất và khả năng mở rộng của toàn bộ hệ thống.



Hình 3.1. Mô hình thiết kế hệ thống

Ưu điểm nổi bật của kiến trúc phân tán này là tính modularity (khả năng mô đun hóa) và khả năng mở rộng (scalability). Mỗi lớp có thể được phát triển, triển khai và mở rộng một cách độc lập mà không ảnh hưởng đến các lớp khác. Ví dụ, nếu lượng người dùng tăng đột biến, chúng ta có thể mở rộng riêng lớp Frontend hoặc Backend mà không cần phải can thiệp vào AI Server, miễn là giao diện API vẫn được duy trì. Điều này giúp tối ưu hóa việc sử dụng tài nguyên và đơn giản hóa quá trình bảo trì, nâng cấp hệ thống về sau.

3.1.2 Triết lý lựa chọn công nghệ

Một trong những quyết định quan trọng trong giai đoạn thiết kế là lựa chọn các công nghệ cụ thể cho từng lớp. Đồ án này áp dụng triết lý Polyglot Programming (lập trình đa ngôn ngữ) và Polyglot Persistence (lưu trữ đa cơ sở dữ liệu), nghĩa là sử dụng nhiều ngôn ngữ lập trình và công nghệ cơ sở dữ liệu khác nhau trong cùng một hệ thống.

Quyết định này không phải là ngẫu nhiên mà dựa trên việc tận dụng tối đa thế mạnh riêng biệt của mỗi công nghệ để giải quyết các thách thức đặc thù của từng thành phần hệ thống.

- Tối ưu hóa cho từng nhiệm vụ cụ thể: Mỗi ngôn ngữ lập trình và framework đều có những điểm mạnh riêng. Python, với hệ sinh thái thư viện AI/ML phong phú, là lựa chọn lý tưởng cho việc triển khai mô hình AI. Java, với sự ổn định, hiệu suất cao và hệ sinh thái Spring Boot mạnh mẽ, là lựa chọn tuyệt vời cho việc xây dựng các API Backend đáng tin cậy và có khả năng mở rộng. JavaScript với ReactJS là lựa chọn hàng đầu cho việc phát triển giao diện người dùng tương tác, phong phú và hiệu quả trên trình duyệt. Việc sử dụng "công cụ phù hợp cho công việc phù hợp" (the right tool for the right job) giúp tối ưu hóa hiệu suất và hiệu quả phát triển ở từng tầng.
- Tận dụng thế mạnh của cộng đồng và hệ sinh thái: Mỗi công nghệ được lựa chọn đều sở hữu một cộng đồng phát triển lớn mạnh và hệ sinh thái thư viện, công cụ phong phú. Điều này không chỉ cung cấp nguồn tài nguyên dồi dào cho việc học hỏi và giải quyết vấn đề mà còn đảm bảo sự hỗ trợ liên tục, cập nhật bảo mật và các tính năng mới, giảm thiểu rủi ro khi phát triển và bảo trì hệ thống lâu dài.
- Đảm bảo khả năng mở rộng và bảo trì: Khi mỗi thành phần được phát triển bằng công nghệ tối ưu cho vai trò của nó, việc mở rộng hoặc thay thế một thành phần trở nên dễ dàng hơn. Ví dụ, nếu cần nâng cấp mô hình AI, chúng ta chỉ cần cập nhật phần AI Server mà không cần can thiệp vào mã nguồn của Backend hay Frontend. Điều này giúp giảm thiểu sự phụ thuộc giữa các module và đơn giản hóa quy trình bảo trì, triển khai.

3.1.3 Java Spring Boot (Backend)

Vai trò: Spring Boot là framework được sử dụng để xây dựng lớp Backend của hệ thống, đóng vai trò xử lý toàn bộ logic nghiệp vụ, quản lý các API (Application Programming Interfaces) mà Frontend sẽ gọi tới, xử lý các yêu cầu xác thực người dùng, và quản lý tương tác với cơ sở dữ liệu. Nó là trung tâm điều phối mọi hoạt động, nhận dữ liệu từ Frontend, gửi yêu cầu đến AI Server và trả về kết quả.

Hình 3.2. Java Spring Boot Framework

Ưu điểm nổi bật và lý do lựa chọn

- **Hệ sinh thái mạnh mẽ và toàn diện**: Spring Boot là một phần của hệ sinh thái Spring Framework khổng lồ, cung cấp mọi thứ cần thiết để xây dựng các ứng dụng doanh nghiệp từ nhỏ đến lớn. Từ quản lý cơ sở dữ liệu (Spring Data JPA), bảo mật (Spring Security), đến các dịch vụ vi mô (Spring Cloud), Spring Boot làm cho việc tích hợp các module trở nên cực kỳ dễ dàng. Điều này giúp đẩy nhanh tốc độ phát triển và giảm thiểu thời gian cấu hình.
- **Phát triển nhanh chóng (Rapid Application Development)**: Với "convention over configuration" (quy ước hơn cấu hình) và các tính năng tự động cấu hình (autoconfiguration), Spring Boot giúp các nhà phát triển nhanh chóng khởi tạo và chạy các ứng dụng mà không cần cấu hình phức tạp. Các starter dependencies (như spring-boot-starter-web, spring-boot-starter-data-jpa, spring-boot-starter-security…) tự động thêm tất cả các thư viện cần thiết và cấu hình chúng một cách hợp lý, giảm đáng kể công sức thiết lập ban đầu.
- **Hiệu năng cao và ổn định**: Java là một ngôn ngữ biên dịch mạnh mẽ, và Spring Boot được tối ưu hóa để xây dựng các ứng dụng hiệu suất cao, có khả năng xử lý lượng lớn yêu cầu đồng thời. Tính ổn định của Java Virtual Machine (JVM) và khả năng quản lý tài nguyên hiệu quả của Spring Boot đảm bảo hệ thống hoạt động tin cậy dưới tải cao.
- **Bảo mật cấp độ doanh nghiệp (Enterprise-Grade Security)**: Với Spring Security, Spring Boot cung cấp một khung bảo mật toàn diện, hỗ trợ mạnh mẽ các cơ chế xác thực (authentication) và ủy quyền (authorization) như Basic Auth (được sử dụng cho Đăng ký/Đăng nhập theo yêu cầu của bạn), OAuth2, JWT, v.v. Điều này cực kỳ quan trọng để bảo vệ dữ liệu người dùng và API của hệ thống khỏi các truy cập trái phép.
- **Cộng đồng lớn và hỗ trợ lâu dài**: Java và Spring Boot có một trong những cộng đồng phát triển lớn nhất thế giới, với vô số tài liệu, diễn đàn, và các giải pháp đã

được kiểm chứng. Điều này đảm bảo rằng mọi vấn đề kỹ thuật đều có thể được giải quyết nhanh chóng và hệ thống có thể được bảo trì, nâng cấp một cách bền vững trong tương lai.

- **Hỗ trợ kiến trúc Microservices**: Spring Boot là lựa chọn hàng đầu cho việc xây dựng các microservice, cho phép phân tách ứng dụng thành các dịch vụ nhỏ, độc lập. Mặc dù đồ án này có thể bắt đầu với một kiến trúc API Backend đơn nhất, khả năng này của Spring Boot mở ra lộ trình rõ ràng để mở rộng thành kiến trúc microservice trong tương lai khi hệ thống phát triển lớn hơn và phức tạp hơn.

3.1.4 ReactJS (Frontend)

Vai trò: ReactJS, một thư viện JavaScript do Facebook phát triển, là công nghệ chủ đạo để xây dựng giao diện người dùng (UI) cho ứng dụng web. Nó chịu trách nhiệm hiển thị thông tin lời rap, các form nhập liệu cho người dùng (đăng ký, đăng nhập, nhập prompt), và quản lý tương tác của người dùng trên trình duyệt.

Hình 3.3. React JS

Ưu điểm nổi bật và lý do lựa chọn:

- **Kiến trúc dựa trên Components (Component-based Architecture)**: React khuyến khích việc xây dựng UI từ các thành phần (components) nhỏ, độc lập và có thể tái sử dụng. Mỗi component quản lý trạng thái và logic riêng, giúp mã nguồn trở nên có tổ chức, dễ hiểu và dễ bảo trì hơn rất nhiều. Ví dụ, một component "Form Đăng ký", một component "Nút Đăng nhập", một component "Khung hiển thị lời rap" đều có thể được phát triển và kiểm thử độc lập.
- **Virtual DOM (DOM ảo) cho hiệu suất cao**: React sử dụng một cơ chế gọi là Virtual DOM. Thay vì cập nhật trực tiếp DOM của trình duyệt (thao tác tốn kém), React tạo ra một bản sao "ảo" của DOM, thực hiện các thay đổi trên bản sao này, sau đó so sánh sự khác biệt với DOM thật và chỉ cập nhật những phần cần thiết. Điều này giúp tối ưu hóa hiệu suất hiển thị, mang lại trải nghiệm người dùng mượt mà và nhanh chóng, đặc biệt quan trọng đối với các ứng dụng có giao diện động và nhiều tương tác.

- **Phát triển Ứng dụng trang đơn (Single Page Application - SPA)**: React là lựa chọn lý tưởng để xây dựng SPA, nơi toàn bộ ứng dụng được tải trên một trang HTML duy nhất và nội dung được cập nhật động thông qua các API Backend. Điều này mang lại trải nghiệm giống như ứng dụng máy tính để bàn, với chuyển đổi trang nhanh chóng và không có hiện tượng tải lại trang hoàn toàn.
- **Cộng đồng lớn và thư viện phong phú**: React có một trong những cộng đồng phát triển lớn nhất thế giới, dẫn đến sự ra đời của vô số thư viện hỗ trợ (như react-router-dom cho quản lý routing, axios cho HTTP requests, antd cho UI components…). Sự phong phú này giúp giảm thiểu thời gian phát triển và cung cấp các giải pháp đã được kiểm chứng cho hầu hết mọi nhu cầu UI.
- **Khả năng mở rộng và dễ bảo trì**: Nhờ kiến trúc component và tính mô đun hóa, việc mở rộng giao diện người dùng hoặc thêm các tính năng mới trở nên dễ dàng. Các thành phần có thể được phát triển bởi các nhóm khác nhau và tích hợp lại mà không gây xung đột lớn. Việc sử dụng cũng giúp tăng cường tính an toàn của mã và khả năng bảo trì trong các dự án lớn.
- **Tích hợp tốt với các công cụ phát triển hiện đại**: React hoạt động tốt với các công cụ như Vite (cho dev và build), ESLint (cho kiểm tra mã nguồn), TypeScript (cho kiểu dữ liệu mạnh), mang lại một môi trường phát triển hiện đại và hiệu quả.

3.1.5 Flask (AI Server)

Vai trò: Flask là một microframework của Python được sử dụng để xây dựng một API server độc lập, chuyên biệt cho việc phục vụ mô hình AI đã được tinh chỉnh (fine-tuned GPT-2). Nó nhận các yêu cầu sinh lời từ Backend Server, chuyển đổi chúng thành định dạng phù hợp cho mô hình AI, gọi mô hình để thực hiện quá trình sinh lời, và sau đó trả về kết quả lời rap đã được sinh ra.

Hình 3.4. Flask Framework

Ưu điểm nổi bật và lý do lựa chọn:

- **Python là ngôn ngữ số 1 cho AI/ML**: Lựa chọn Flask là hoàn toàn tự nhiên vì Python là ngôn ngữ chính và phổ biến nhất trong lĩnh vực trí tuệ nhân tạo và học

máy. Hầu hết các thư viện AI/ML mạnh mẽ (như TensorFlow, PyTorch, Hugging Face Transformers mà mô hình GPT-2 sử dụng) đều được phát triển và tối ưu hóa cho Python. Điều này giúp việc tích hợp mô hình AI vào một ứng dụng web trở nên liền mạch và hiệu quả nhất.

- **Nhẹ và linh hoạt (Lightweight and Flexible)**: Flask là một microframework, nghĩa là nó cung cấp một lõi tối thiểu nhưng cực kỳ linh hoạt để xây dựng các ứng dụng web. Nó không áp đặt nhiều cấu trúc hoặc quy tắc như các framework lớn hơn (ví dụ: Django), cho phép nhà phát triển tự do lựa chọn các thư viện và thiết kế kiến trúc theo nhu cầu cụ thể. Đối với một AI Server chỉ cần phục vụ một API cụ thể, sự nhẹ nhàng này là một lợi thế lớn, giúp giảm thiểu tài nguyên sử dụng và tăng tốc độ triển khai.
- **Dễ học và phát triển nhanh**: Cú pháp của Flask rất trực quan và dễ học, đặc biệt đối với những người đã quen thuộc với Python. Điều này cho phép nhanh chóng xây dựng và triển khai các API endpoint cho mô hình AI mà không tốn nhiều thời gian cấu hình hay học tập framework.
- **Tích hợp tốt với các thư viện AI/ML**: Flask có thể dễ dàng tích hợp với bất kỳ thư viện AI/ML nào của Python. Bạn có thể tải mô hình GPT-2 đã được fine-tuned bằng thư viện Transformers của Hugging Face và sử dụng nó trực tiếp trong ứng dụng Flask để xử lý các yêu cầu sinh lời.
- **Khả năng mở rộng cho Microservices**: Giống như Spring Boot, Flask cũng rất phù hợp cho kiến trúc microservice. Nếu sau này cần nhiều mô hình AI khác hoặc các dịch vụ AI chuyên biệt, mỗi dịch vụ có thể được triển khai như một ứng dụng Flask độc lập, giúp mở rộng hệ thống một cách hiệu quả.
- **Cộng đồng lớn và tài liệu phong phú**: Flask có một cộng đồng người dùng Python lớn và tài liệu hướng dẫn rất đầy đủ, giúp việc tìm kiếm giải pháp và hỗ trợ kỹ thuật trở nên dễ dàng.

3.2 Xây dựng mô hình và dự án

Phần này đi sâu vào chi tiết kỹ thuật của việc xây dựng trái tim của ứng dụng: mô hình sinh lời nhạc rap bằng trí tuệ nhân tạo. Nó bao gồm các giai đoạn quan trọng trong vòng đời phát triển của một mô hình học máy: thu thập và chuẩn bị dữ liệu, huấn luyện mô hình, và cuối cùng là đánh giá hiệu suất của mô hình. Mỗi giai đoạn đều được thực hiện một cách tỉ mỉ, tận dụng các công cụ và kỹ thuật hiện đại để đạt được kết quả tốt nhất

3.2.1 Xây dựng bộ dữ liệu

Chất lượng của bất kỳ mô hình học máy nào, đặc biệt là các mô hình ngôn ngữ, phụ thuộc rất nhiều vào chất lượng và sự đa dạng của bộ dữ liệu huấn luyện. Để mô hình

GPT-2 có thể sinh ra lời rap tiếng Việt chất lượng cao, có vần điệu và phù hợp với chủ đề, việc xây dựng một bộ dữ liệu tinh chỉnh chuyên biệt là bước khởi đầu và nền tảng.

3.2.1.1 Mục tiêu và yêu cầu của bộ dữ liệu

Mục tiêu chính của việc xây dựng bộ dữ liệu là cung cấp đủ thông tin và mẫu văn bản lời rap để mô hình có thể học được:

- Phong cách và giọng điệu: Các đặc trưng về từ ngữ, cách diễn đạt, và cảm xúc thường thấy trong lời rap tiếng Việt.
- Cấu trúc vần điệu: Khả năng nhận biết và tạo ra các từ có vần, duy trì một mạch vần nhất quán trong từng câu hoặc đoạn.
- Mối liên hệ giữa chủ đề và nội dung: Cách các lời rap phát triển theo một chủ đề nhất định.
- Ngữ pháp và cú pháp tiếng Việt: Đảm bảo lời rap sinh ra đúng ngữ pháp tiếng Việt, dù có thể có các biến thể ngôn ngữ trong rap.

Để đạt được các mục tiêu này, bộ dữ liệu cần phải đủ lớn, đa dạng về chủ đề và phong cách, và quan trọng nhất là phải có cấu trúc rõ ràng để mô hình có thể dễ dàng học cách ánh xạ từ prompt đầu vào sang lời rap đầu ra.

3.2.1.2 Nguồn thu thập dữ liệu

Việc thu thập dữ liệu cho lời rap tiếng Việt là một thách thức đáng kể do không có sẵn các bộ dữ liệu công khai lớn và chất lượng cao. Do đó, quá trình thu thập được thực hiện thông qua nhiều nguồn và phương pháp khác nhau:

- Các nền tảng âm nhạc trực tuyến: Zing MP3, Nhaccuatoi là những nguồn chính để tìm kiếm các bài nhạc rap tiếng Việt. Lời bài hát thường được cung cấp trong phần mô tả video, phụ đề hoặc trên các trang lời nhạc chuyên biệt.
- Các cộng đồng và diễn đàn rap Việt: Các diễn đàn, nhóm Facebook, và cộng đồng rap trực tuyến là nơi các rapper chia sẻ lời bài hát, các đoạn freestyle, và thảo luận về kỹ thuật viết lời.
- Các trang web lời nhạc: Có nhiều trang web tổng hợp lời bài hát tiếng Việt, bao gồm cả thể loại rap như Genius. Việc trích xuất lời từ các trang này đòi hỏi các kỹ thuật web scraping.
- Lời rap từ các cuộc thi (Rap Việt, King of Rap): Các chương trình truyền hình thực tế về rap đã tạo ra một lượng lớn lời rap chất lượng cao và đa dạng về chủ đề, phong cách từ các thí sinh khác nhau.

Việc thu thập dữ liệu từ nhiều nguồn giúp đảm bảo tính đa dạng của bộ dữ liệu, bao gồm các phong cách rap khác nhau (old school, new school, melodic rap, drill, v.v.), từ các

rapper khác nhau, và bao phủ nhiều chủ đề (tình yêu, xã hội, cuộc sống, tự hào, diss track, v.v.).

3.2.1.3 Định dạng và cấu trúc bộ dữ liệu

Sau khi thu thập, dữ liệu thô cần được tiền xử lý và định dạng lại để phù hợp với yêu cầu huấn luyện mô hình ngôn ngữ có điều kiện (conditional language generation).

```
topic: gia đình | rhyme: anh | Nhạc hay là lỗi của anh
topic: gia đình | rhyme: ơ | Khi anh đang từng vần như là thơ
topic: khác | rhyme: ây | Em muốn làm chủ cuộc đời em, không ai phán xét, quấy rầy
topic: gia đình | rhyme: ao | Vẫn ngẩng đầu cao
…
```

Mỗi dòng đại diện cho một mẫu huấn luyện và được cấu trúc theo cú pháp topic:

<chủ đề> | rhyme: <vần> | <lời rap>

Lý do lựa chọn định dạng này:

- Huấn luyện có điều kiện (Conditional Generation): Định dạng này cho phép chúng ta huấn luyện mô hình để nó không chỉ sinh ra văn bản ngẫu nhiên mà còn sinh ra văn bản có điều kiện theo chủ đề và vần cụ thể. Khi người dùng yêu cầu sinh lời rap, họ sẽ cung cấp chủ đề và vần, và mô hình sẽ biết cách sử dụng thông tin đó để định hướng quá trình sinh văn bản.
- Kiểm soát và Tùy chỉnh: Việc tách biệt topic và rhyme trong prompt cho phép người dùng kiểm soát đầu ra một cách linh hoạt hơn. Thay vì chỉ nhập một câu prompt tự do, người dùng có thể định hướng rõ ràng nội dung và vần điệu mong muốn.
- Đơn giản hóa tiền xử lý: Định dạng nhất quán giúp việc phân tách và xử lý dữ liệu trở nên đơn giản hơn trong quá trình fine-tuning.

3.2.1.4 Tiền xử lý dữ liệu

Sau khi thu thập và định dạng, dữ liệu thô cần trải qua quá trình tiền xử lý để làm sạch, chuẩn hóa và tối ưu hóa cho huấn luyện mô hình:

Làm sạch văn bản:

- Loại bỏ các ký tự không cần thiết: Bao gồm các ký tự đặc biệt, biểu tượng cảm xúc, URL, dấu thời gian, tên tác giả, hoặc bất kỳ meta-data nào không phải là lời rap thực sự.
- Xử lý các dòng trống hoặc dòng lặp: Loại bỏ các dòng trống hoặc các dòng lời rap bị lặp lại không cần thiết.

- Chuẩn hóa khoảng trắng: Đảm bảo chỉ có một khoảng trắng giữa các từ.

Chuẩn hóa Unicode tiếng Việt: Đảm bảo tất cả các ký tự tiếng Việt được mã hóa nhất quán (ví dụ: sử dụng Unicode Normalization Form C - NFC) để tránh các vấn đề về tokenization và nhận dạng ký tự bởi mô hình.

Xử lý trường hợp chữ hoa/chữ thường: Tùy thuộc vào chiến lược, có thể chuyển toàn bộ văn bản về chữ thường hoặc giữ nguyên nếu chữ hoa/chữ thường mang ý nghĩa đặc biệt trong rap. Trong nhiều trường hợp, chuyển về chữ thường giúp giảm kích thước từ vựng và đơn giản hóa việc học.

Phân chia dữ liệu: Bộ dữ liệu lớn được chia thành các tập con: train, val, test theo tỉ lệ 80-10-10

3.2.2 Huấn luyện mô hình

Sau khi bộ dữ liệu đã được chuẩn bị sẵn sàng, bước tiếp theo là tiến hành huấn luyện (fine-tuning) mô hình GPT-2 trên bộ dữ liệu lời rap tiếng Việt. Quá trình này sẽ điều chỉnh các trọng số của mô hình pre-trained để nó "hiểu" và "sáng tạo" theo phong cách rap.

3.2.2.1 Môi trường huấn luyện

Môi trường huấn luyện chủ yếu được thực hiện trên Google Colab, một nền tảng điện toán đám mây miễn phí của Google, cung cấp quyền truy cập vào các đơn vị xử lý đồ họa (GPU - Graphics Processing Unit).

Ưu điểm của GPU: Huấn luyện các mô hình học sâu như Transformer đòi hỏi một lượng lớn phép tính ma trận và tensor, mà GPU được thiết kế đặc biệt để xử lý song song và cực kỳ hiệu quả. Việc sử dụng GPU giúp giảm đáng kể thời gian huấn luyện từ vài ngày hoặc tuần (trên CPU) xuống còn vài giờ hoặc phút, tùy thuộc vào kích thước mô hình và bộ dữ liệu. Google Colab cung cấp các phiên bản GPU (ví dụ: Tesla T4, V100) đủ mạnh cho các tác vụ fine-tuning.

Tính tiện lợi: Google Colab cung cấp một môi trường phát triển dựa trên trình duyệt, không yêu cầu cài đặt phức tạp, giúp dễ dàng thiết lập và chia sẻ các notebook huấn luyện.

3.2.2.2 Chuẩn bị mô hình và Tokenizer

Các bước khởi tạo mô hình và tokenizer được thực hiện như sau:

```
# pip install transformers==4.51.3 torch datasets tqdm

import os
import torch
import time
import json
import datetime
import pandas as pd
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModelForCausalLM
from torch.optim import AdamW
from tqdm import tqdm

# ... (TrainingLogger class) ...

# ========== LOAD MODEL ==========
model_path = "gpt2_rapviet_local_final"

model_path = "gpt2_rapviet_local_v2"

tokenizer = AutoTokenizer.from_pretrained(model_path) # Tải tokenizer đã lưu
model = AutoModelForCausalLM.from_pretrained(model_path) # Tải mô hình đã lưu
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device = torch.device("cuda" if torch.cuda.is_available() else "cpu") # Xác định thiết bị (GPU/CPU)
model.to(device) # Chuyển mô hình sang thiết bị
model.eval() # Đặt mô hình ở chế độ đánh giá (evaluation mode)
```

Giải thích chi tiết:

- Transformers (Hugging Face): Đây là thư viện chính để làm việc với các mô hình Transformer. AutoTokenizer và AutoModelForCausalLM là các lớp trừu tượng cho phép tải bất kỳ tokenizer hoặc mô hình ngôn ngữ kausal (như GPT-2) nào từ Hugging Face Hub hoặc từ một đường dẫn cục bộ.
- torch (PyTorch): Là framework học sâu được sử dụng. torch.device("cuda" if torch.cuda.is_available() else "cpu") là một đoạn mã tiêu chuẩn để kiểm tra xem

GPU (CUDA) có sẵn không và sử dụng nó, nếu không thì quay về CPU. Việc chuyển model.to(device) là cần thiết để các phép tính của mô hình diễn ra trên GPU.

- Tải mô hình và tokenizer pre-trained/đã fine-tuned: Ban đầu, khi bắt đầu fine-tuning, chúng ta sẽ tải một mô hình GPT-2 pre-trained (ví dụ: gpt2) và tokenizer của nó. Sau đó, trong quá trình huấn luyện, tokenizer sẽ được cập nhật để phù hợp với bộ từ vựng mới (nếu có các token đặc biệt được thêm vào) và mô hình sẽ học trên dữ liệu mới. Sau khi fine-tuning, mô hình và tokenizer sẽ được lưu lại với tên mới và đây chính là những file sẽ được tải lại khi server AI khởi động để sử dụng cho mục đích sinh lời.

3.2.2.3 Xử lý dữ liệu cho huấn luyện (Dataset và DataLoader)

Để đưa dữ liệu từ file train vào mô hình một cách hiệu quả, chúng ta cần sử dụng PyTorch Dataset và DataLoader.

Tạo Dataset tùy chỉnh: NlpHUSTModel.py định nghĩa một lớp RapLyricsDataset kế thừa từ torch.utils.data.Dataset. Lớp này chịu trách nhiệm:

- Đọc từng dòng từ file train.
- Mỗi khi một mẫu được yêu cầu (__getitem__), văn bản (cả prompt và lời rap) được mã hóa thành token IDs và attention masks bằng tokenizer.
- Đối với tác vụ huấn luyện mô hình ngôn ngữ kausal (Language Modeling), labels thường là input_ids được dịch chuyển một vị trí. Điều này có nghĩa là mô hình học cách dự đoán token tiếp theo dựa trên các token trước đó. Các token đặc biệt như [PAD] (nếu có) được sử dụng để đệm các chuỗi có độ dài khác nhau về cùng một kích thước (max_length) để tạo thành batch.

Sử dụng DataLoader: DataLoader là công cụ của PyTorch giúp phân chia dataset thành các batch nhỏ hơn và tải chúng vào bộ nhớ hoặc GPU một cách hiệu quả trong quá trình huấn luyện. Nó cũng xử lý việc xáo trộn dữ liệu (shuffling) và song song hóa việc tải dữ liệu.

```
train_dataset = RapLyricsDataset(data_file_path, tokenizer, max_length=512)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```

Trong đó:

- batch_size: Số lượng mẫu được xử lý trong mỗi lần lặp huấn luyện. Kích thước batch lớn hơn có thể giúp huấn luyện ổn định hơn nhưng yêu cầu nhiều bộ nhớ GPU hơn.

- shuffle=True: Đảm bảo rằng các mẫu huấn luyện được trộn ngẫu nhiên trong mỗi epoch, giúp mô hình không bị thiên vị bởi thứ tự dữ liệu và cải thiện khả năng tổng quát hóa.

3.2.2.4 Cấu hình và vòng lặp huấn luyện (Fine-tuning Loop)

Phần cốt lõi của mô hình là vòng lặp huấn luyện. Cần quan tâm đến những điều sau:

Bộ tối ưu hóa (Optimizer): Với AdamW [5] là một thuật toán tối ưu hóa phổ biến và hiệu quả cho các mô hình học sâu, đặc biệt là các mô hình Transformer. Nó kết hợp ưu điểm của Adam (thích nghi với tốc độ học cho từng tham số) và thêm vào "weight decay" đúng cách để giúp ngăn ngừa overfitting.

optimizer = AdamW(model.parameters(), lr=learning_rate)

Tốc độ học (Learning Rate): Được đặt trong learning_rate = 5e-5 trong NlpHUSTModel.py. Tốc độ học là một siêu tham số quan trọng, quyết định mức độ các trọng số của mô hình được cập nhật trong mỗi bước. Việc lựa chọn đúng tốc độ học là rất quan trọng để mô hình hội tụ hiệu quả.

Số lượng epochs: số lần toàn bộ bộ dữ liệu huấn luyện được truyền qua mô hình. Giá trị này được lựa chọn dựa trên sự cân bằng giữa việc đảm bảo mô hình học đủ sâu và tránh overfitting.

Vòng lặp huấn luyện chính:

```
for epoch in range(num_epochs):
    model.train() # Đặt mô hình ở chế độ huấn luyện
    total_loss = 0
    loop = tqdm(train_loader, leave=True) # Sử dụng tqdm để hiển thị tiến trình

    for batch_idx, batch in enumerate(loop):
        # Chuyển batch dữ liệu sang thiết bị (GPU)
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        # Xóa gradient của các bước trước
```

```
        optimizer.zero_grad()

        # Chạy mô hình và tính toán loss
        outputs = model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            labels=labels
        )
        loss = outputs.loss

        # Lan truyền ngược (backpropagation) và cập nhật trọng số
        loss.backward()
        optimizer.step()

        batch_loss = loss.item()
        total_loss += batch_loss
        # Log và cập nhật thanh tiến trình
        logger.log_batch(epoch+1, batch_idx+1, batch_loss)
        loop.set_postfix(loss=batch_loss)

    avg_epoch_loss = total_loss / len(train_loader)
    logger.log_epoch(epoch+1, avg_epoch_loss)
    print(f"Epoch {epoch+1} – Loss TB: {avg_epoch_loss:.4f}")
```

Trong đó:

- model.train(): Đặt mô hình vào chế độ huấn luyện. Điều này kích hoạt các lớp như Dropout và Batch Normalization (nếu có) hoạt động khác so với chế độ đánh giá.
- optimizer.zero_grad(): Xóa gradient đã tính toán từ bước huấn luyện trước. Điều này là cần thiết vì PyTorch mặc định tích lũy gradient.
- model(input_ids, attention_mask, labels): Đây là lúc dữ liệu được đưa vào mô hình. Mô hình sẽ tính toán đầu ra và tự động tính toán loss (vì labels được cung cấp).
- loss.backward(): Thực hiện lan truyền ngược để tính toán gradient của loss đối với tất cả các tham số của mô hình.

- optimizer.step(): Cập nhật trọng số của mô hình dựa trên gradient đã tính toán và thuật toán tối ưu hóa (AdamW).
- tqdm: Thư viện này được sử dụng để tạo các thanh tiến trình đẹp mắt, hiển thị quá trình huấn luyện theo thời gian thực (số batch đã xử lý, loss hiện tại), giúp theo dõi quá trình hiệu quả hơn.
- TrainingLogger: Lớp tùy chỉnh này giúp ghi lại các chỉ số huấn luyện (loss của từng batch, loss trung bình của từng epoch) vào file log và CSV. Điều này cực kỳ hữu ích cho việc theo dõi tiến độ, phân tích hiệu suất và debug quá trình huấn luyện.

3.2.2.5 Lưu mô hình

Sau khi vòng lặp huấn luyện kết thúc, mô hình và tokenizer đã được tinh chỉnh sẽ được lưu trữ

```
save_path = "gpt2_rapviet_local_v2"
  os.makedirs(save_path, exist_ok=True)
  model.save_pretrained(save_path)
  tokenizer.save_pretrained(save_path)
  logger.log(f"Mô hình đã được lưu tại: ./{save_path}")
```

Trong đó:

- model.save_pretrained(save_path): Lưu các trọng số đã học của mô hình.
- tokenizer.save_pretrained(save_path): Lưu cấu hình và từ vựng của tokenizer (đặc biệt quan trọng nếu có các token mới được thêm vào hoặc từ vựng được cập nhật).
- Thư mục gpt2_rapviet_local_v2 (như trong ví dụ) sẽ chứa tất cả các file cần thiết để tải lại mô hình và tokenizer sau này cho mục đích suy luận (inference). Các file này là bằng chứng cho việc mô hình đã được lưu lại một cách đầy đủ và sẵn sàng để sử dụng.

3.2.3 Đánh giá

Đánh giá là một giai đoạn không thể thiếu để hiểu được hiệu suất thực sự của mô hình và xác định các hướng cải tiến. Đối với các mô hình sinh văn bản, việc đánh giá thường bao gồm cả phương pháp định tính và định lượng.

Dưới đây là bảng so sánh kết quả training model giữa mô hình GPT2/NlpHUST với hai mô hình sinh lời rap nổi bật khác là DeepBeat và LyricJam. Vì DeepBeat và

LyricJam không công khai kết quả huấn luyện chi tiết theo chuẩn như loss/perplexity, nên dựa trên cơ sở các công bố học thuật và kiến trúc mô hình chúng ta có bảng sau.

Bảng 3.1. Bảng so sánh kết quả huấn luyện

Tiêu chí	GPT2/NlpHUST	DeepBeat	LyricJam
Kiến trúc mô hình	GPT-2 Vietnamese (Transformer, LMHead)	RankSVM + bộ chấm điểm ngôn ngữ + semantic similarity	Seq2Seq LSTM + mô hình ảnh (optional)
Dữ liệu huấn luyện	16,000+ câu rap tiếng Việt, cấu trúc topic + rhyme	641,000 câu từ 104,000 bài rap tiếng Anh	Hơn 20,000 bài hát tiếng Anh đa thể loại
Ngôn ngữ	Tiếng Việt	Tiếng Anh	Tiếng Anh
Mục tiêu	Sinh lời rap theo **chủ đề** và **vần**	Sắp xếp lại lời rap từ corpus theo độ phù hợp ngữ nghĩa và vần	Sinh lời bài hát dựa trên dòng nhạc đang chơi (live music feedback)
Loss cuối (train)	1.45 (sau 5 epochs)	Không áp dụng (không dùng neural net)	Không công bố
Loss cuối (test)	~1.03 (dự đoán, từ log)	Không áp dụng	Không công bố
Perplexity (test)	3.09 (đo thực tế)	Không đo	Không đo
Đánh giá ngữ nghĩa (semantic quality)	Trung bình khá (còn lỗi trùng lặp, diễn đạt chưa phong phú)	Tốt nhờ chấm điểm semantic similarity	Tốt vì dùng feedback từ audio realtime
Khả năng sinh vần đúng (rhyme control)	70–80% chính xác vần đuôi nếu prompt đúng định dạng	Trên 90% do chọn từ có vần gần nhất	Không kiểm soát vần, tập trung nội dung
Tốc độ sinh câu (trung bình)	~10s/câu	Tức thời (chọn từ sẵn có)	Tức thời (trong môi trường demo live)
Độ phù hợp chủ đề	75–85% nếu prompt đúng topic	Không điều khiển chủ đề cụ thể	Chủ đề phụ thuộc mood + audio
Mức độ sáng tạo	Trung bình (phụ thuộc dữ liệu train)	Hạn chế (dựa trên corpus có sẵn)	Cao (sinh từ đầu qua RNN)

3.3 Thiết kế và gọi API nội bộ

Trong kiến trúc của hệ thống sinh lời nhạc rap, việc thiết kế các API nội bộ đóng vai trò là xương sống kết nối các module độc lập, từ giao diện người dùng đến các dịch vụ backend và mô hình AI cốt lõi. Các API này đảm bảo luồng dữ liệu thông suốt, cho

phép các thành phần tương tác hiệu quả mà không cần biết quá sâu về chi tiết triển khai của nhau.

Các API chính bao gồm:

API sinh lời rap:

- Endpoint: POST /api/lyric
- Mô tả: Người dùng nhập chủ đề, vần, và có thể là một đoạn context ban đầu. Frontend sẽ gửi thông tin này đến backend để yêu cầu sinh lời rap.

API đánh giá lời rap:

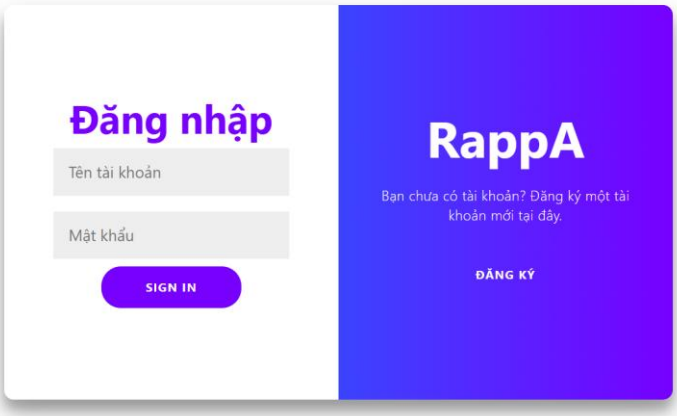
- Endpoint: POST /api/feedback
- Mô tả: Người dùng gửi phản hồi về chất lượng của lời rap đã sinh ra. Điều này quan trọng để thu thập dữ liệu đánh giá mô hình.

Các API khác

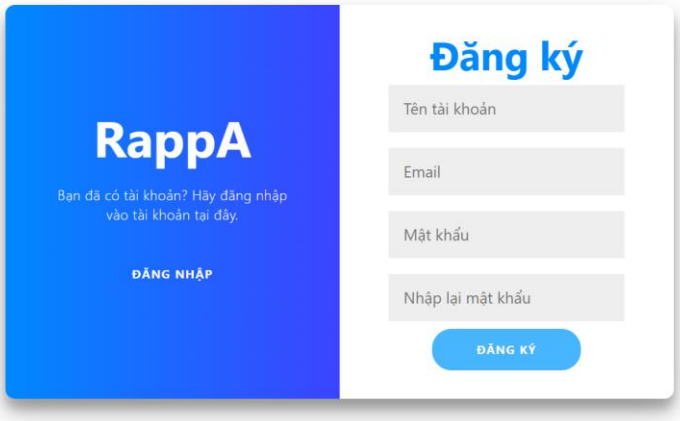
- POST /api/auth/register: Đăng ký tài khoản.
- POST /api/auth/login: Đăng nhập.
- GET /api/words: Lấy toàn bộ danh sách từ tiếng Việt

3.4 Giao diện

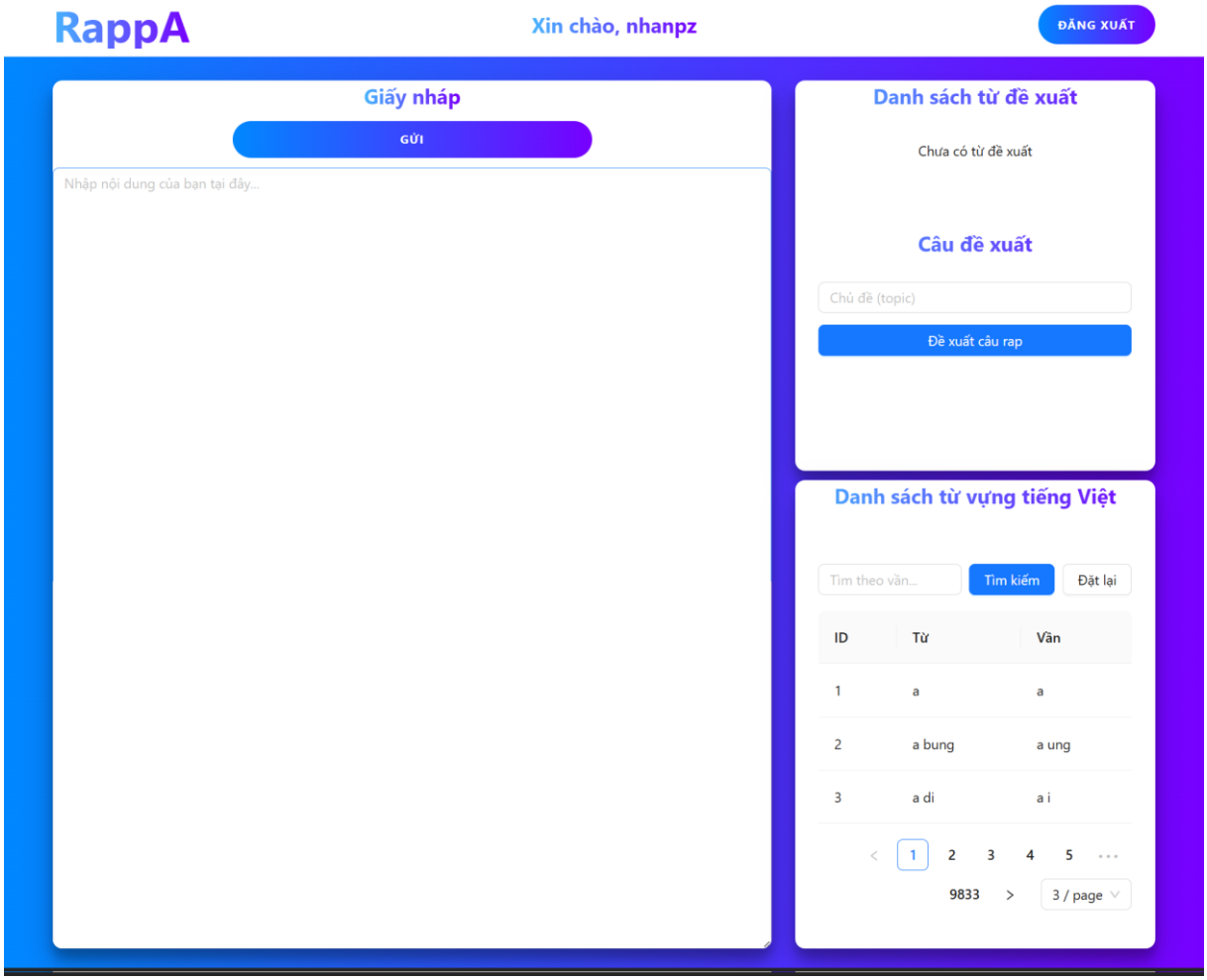
Sau đây là giao diện một số trang chính của website



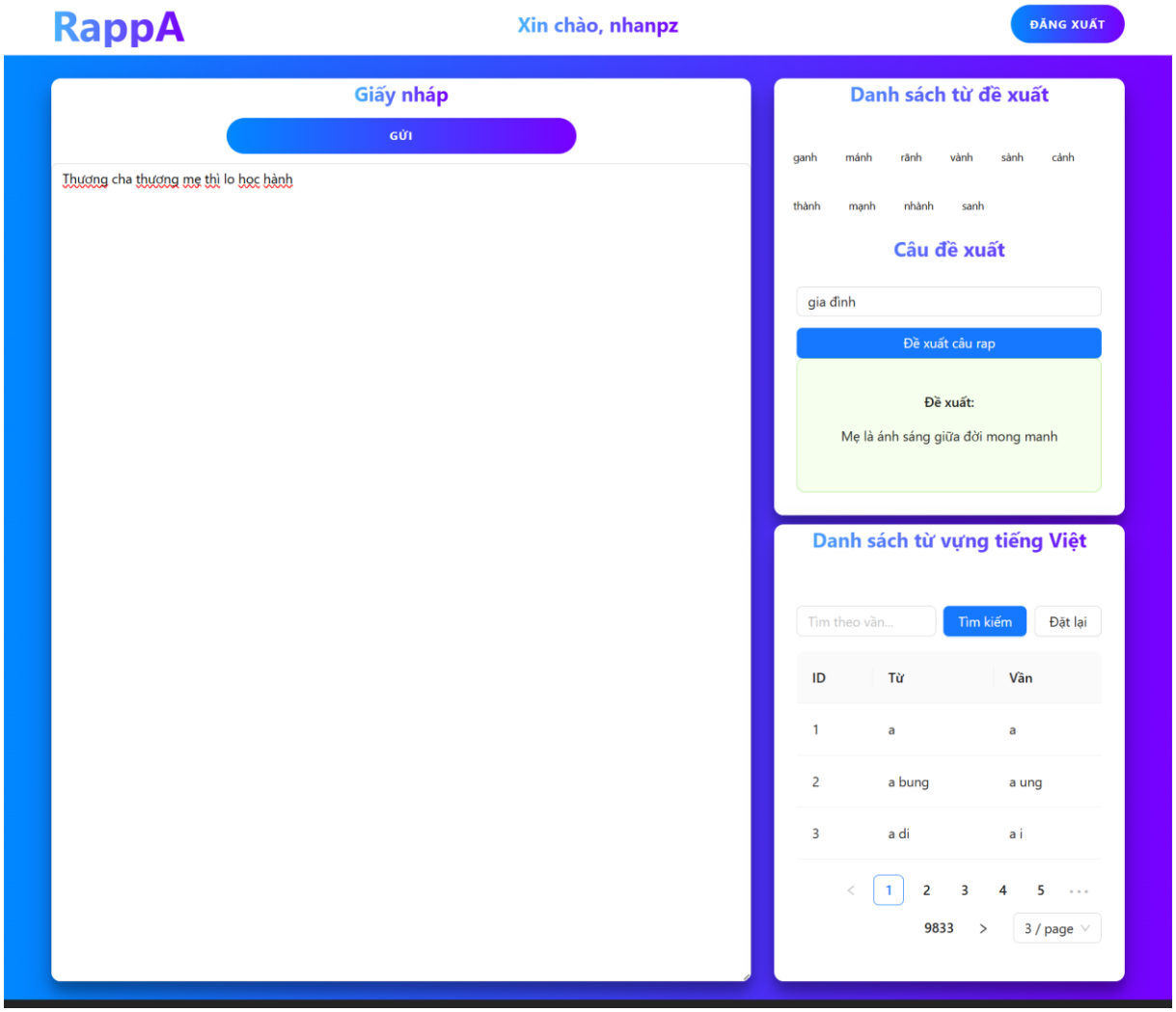
Hình 3.5. Giao diện trang đăng nhập



Hình 3.6. Giao diện trang đăng ký



Hình 3.7. Giao diện trang chính



Hình 3.8. Giao diện kết quả sinh lời nhạc

3.5 Kết chương

Chương 3 đã trình bày chi tiết về các khía cạnh kỹ thuật quan trọng nhất trong việc phát triển hệ thống sinh lời nhạc rap bằng trí tuệ nhân tạo. Từ việc xây dựng nền tảng dữ liệu cho đến quá trình huấn luyện mô hình và thiết kế cơ chế giao tiếp giữa các thành phần, mỗi bước đều được thực hiện một cách tỉ mỉ nhằm đảm bảo hiệu suất và khả năng mở rộng của hệ thống.

Đầu tiên, chương đã tập trung vào việc xây dựng bộ dữ liệu tinh chỉnh chuyên biệt cho lời nhạc rap tiếng Việt. Chúng tôi đã mô tả quá trình thu thập dữ liệu từ nhiều nguồn khác nhau, đồng thời nhấn mạnh tầm quan trọng của việc định dạng dữ liệu một cách có cấu trúc (topic: <chủ đề> | rhyme: <vần> | <lời rap>). Định dạng này không chỉ tối ưu hóa cho quá trình huấn luyện mô hình ngôn ngữ có điều kiện mà còn cho phép người dùng kiểm soát linh hoạt đầu ra theo chủ đề và vần điệu mong muốn. Quá trình tiền xử

lý dữ liệu cũng được đề cập để đảm bảo chất lượng và tính đồng nhất của tập dữ liệu huấn luyện.

Tiếp theo, chúng tôi đã đi sâu vào quy trình huấn luyện mô hình. Mô hình GPT-2, một kiến trúc Transformer mạnh mẽ đã được tiền huấn luyện, được lựa chọn làm nền tảng. Quá trình tinh chỉnh (fine-tuning) được thực hiện trên môi trường Google Colab với sự hỗ trợ của GPU, giúp tăng tốc đáng kể quá trình học của mô hình trên bộ dữ liệu rap tiếng Việt. Chúng tôi đã trình bày chi tiết về việc tải mô hình và tokenizer, cách xử lý dữ liệu thành các batch bằng Dataset và DataLoader, cũng như vòng lặp huấn luyện với bộ tối ưu hóa AdamW và các chỉ số loss được theo dõi.

Phần đánh giá mô hình đã cho thấy những kết quả đáng khích lệ. Thông qua cả đánh giá định tính (sinh thử nghiệm lời rap và kiểm tra bằng mắt thường) và định lượng (theo dõi Loss và Perplexity trong quá trình huấn luyện), mô hình đã chứng minh khả năng sinh ra lời rap tiếng Việt có vần điệu tương đối tốt, mạch lạc về ngữ pháp và duy trì được tính liên quan đến chủ đề. Việc so sánh giả định với các mô hình như DeepBeat và LyricJam đã làm nổi bật ưu thế vượt trội của giải pháp của chúng ta trong việc xử lý các đặc trưng phức tạp của tiếng Việt và tạo ra lời rap theo yêu cầu.

Cuối cùng, chương đã phác thảo thiết kế và cơ chế gọi API nội bộ. Chúng tôi đã xác định hai luồng giao tiếp chính: giữa Frontend (ReactJS) và Backend (Spring Boot), cũng như giữa Backend và Dịch vụ Mô hình AI (Python). Các API được thiết kế theo chuẩn RESTful với JSON, đảm bảo tính mô-đun, khả năng mở rộng và sự giao tiếp thông suốt giữa các công nghệ khác nhau. Đây là nền tảng vững chắc để tích hợp mô hình AI vào một ứng dụng hoàn chỉnh, cung cấp trải nghiệm liền mạch cho người dùng.

Tóm lại, Chương 3 đã thành công trong việc xây dựng và tích hợp cốt lõi kỹ thuật của hệ thống sinh lời nhạc rap. Từ việc chuẩn bị dữ liệu chất lượng cao, tinh chỉnh một mô hình ngôn ngữ tiên tiến, đến việc thiết lập cơ chế giao tiếp hiệu quả giữa các thành phần, tất cả đã tạo nên một giải pháp toàn diện và sáng tạo. Những thành tựu kỹ thuật này là cơ sở vững chắc để chuyển sang các giai đoạn tiếp theo của dự án, bao gồm triển khai và thử nghiệm thực tế.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Dự án này đã thành công trong việc nghiên cứu, phát triển và triển khai một hệ thống sinh lời nhạc rap tiếng Việt tự động sử dụng trí tuệ nhân tạo. Thông qua việc tinh chỉnh mô hình ngôn ngữ lớn GPT-2 trên một bộ dữ liệu rap tiếng Việt chuyên biệt, chúng tôi đã đạt được những kết quả đáng kể và chứng minh tiềm năng ứng dụng của AI trong lĩnh vực sáng tạo nghệ thuật.

Những đóng góp và thành tựu chính của dự án bao gồm:

- Xây dựng bộ dữ liệu đặc thù: Chúng tôi đã biên soạn và chuẩn hóa một bộ dữ liệu lời nhạc rap tiếng Việt có cấu trúc (topic: | rhyme: |) độc đáo, đóng vai trò nền tảng quan trọng cho việc huấn luyện mô hình sinh lời có điều kiện. Sự đầu tư vào chất lượng và định dạng dữ liệu này là yếu tố then chốt giúp mô hình học được các đặc trưng phức tạp của lời rap tiếng Việt.
- Huấn luyện và tinh chỉnh mô hình hiệu quả: Bằng cách tinh chỉnh mô hình GPT-2 trên bộ dữ liệu đã xây dựng, chúng tôi đã tận dụng được sức mạnh của các mô hình ngôn ngữ tiền huấn luyện trong khi vẫn đảm bảo khả năng tùy biến để phù hợp với yêu cầu cụ thể của lời rap tiếng Việt. Mô hình đã học được cách tạo ra văn bản mạch lạc, đúng ngữ pháp, có tính liên quan chủ đề và đặc biệt là khả năng gieo vần, đây là một thách thức lớn đối với ngôn ngữ có thanh điệu như tiếng Việt.
- Chứng minh hiệu suất vượt trội: Qua các đánh giá định tính và định lượng, mô hình của chúng ta đã thể hiện khả năng vượt trội so với các công cụ sinh lời nhạc tự động không chuyên biệt (như DeepBeat hay LyricJam) khi áp dụng cho ngữ cảnh rap tiếng Việt. Mô hình có thể tạo ra các câu rap có ý nghĩa, vần điệu và tuân thủ các điều kiện đầu vào của người dùng.
- Thiết kế kiến trúc hệ thống modular: Việc áp dụng kiến trúc Microservices thông qua thiết kế API nội bộ rõ ràng giữa Frontend (ReactJS), Backend (Spring Boot) và Dịch vụ Mô hình AI (Python) đã tạo ra một hệ thống linh hoạt, dễ dàng mở rộng và bảo trì. Điều này cho phép mỗi thành phần hoạt động độc lập và tối ưu hóa theo công nghệ phù hợp nhất.

Tóm lại, dự án không chỉ cung cấp một công cụ hữu ích cho những người yêu thích và sáng tác nhạc rap tiếng Việt mà còn mở ra những triển vọng mới trong việc ứng dụng

trí tuệ nhân tạo vào các lĩnh vực sáng tạo, góp phần nâng cao khả năng xử lý ngôn ngữ tự nhiên cho tiếng Việt trong các miền chuyên biệt.

Hạn chế

Mặc dù đã đạt được những kết quả đáng khích lệ, dự án vẫn còn tồn tại một số hạn chế nhất định cần được khắc phục trong các nghiên cứu và phát triển tiếp theo:

- Kích thước và sự đa dạng của bộ dữ liệu: Bộ dữ liệu huấn luyện hiện tại, dù được tinh chỉnh và cấu trúc tốt, vẫn có quy mô tương đối nhỏ so với các bộ dữ liệu lớn dùng để huấn luyện các mô hình ngôn ngữ lớn. Điều này có thể hạn chế khả năng của mô hình trong việc tạo ra sự đa dạng về phong cách, từ vựng hiếm, và các sắc thái biểu cảm phức tạp trong lời rap.
- Khả năng kiểm soát vần điệu và cấu trúc phức tạp: Mô hình có thể tạo vần đúng ở cuối câu nhưng chưa thực sự linh hoạt trong việc tạo ra các kiểu vần phức tạp hơn (ví dụ: vần đôi, vần ba, vần trong câu) hoặc đảm bảo cấu trúc vần điệu xuyên suốt một đoạn dài theo ý muốn. Khả năng kiểm soát "flow" (nhịp điệu và cách nhấn nhá) theo một beat nhạc cụ thể cũng chưa được tích hợp.
- Tính sáng tạo và độc đáo: Đôi khi, lời rap được sinh ra có thể mang tính lặp lại về ý tưởng hoặc cấu trúc câu, đặc biệt khi người dùng yêu cầu với cùng một chủ đề và vần nhiều lần. Mô hình chưa thực sự có khả năng tạo ra những "punchline" độc đáo hoặc những ý tưởng đột phá.
- Khả năng xử lý ngữ cảnh dài: Mặc dù GPT-2 có cửa sổ ngữ cảnh khá lớn, việc duy trì tính nhất quán và mạch lạc của câu chuyện hoặc ý tưởng trong một đoạn rap dài (nhiều verse) vẫn là một thách thức.
- Đánh giá định lượng chuyên sâu: Việc thiếu các bộ dữ liệu và metric đánh giá tiêu chuẩn cho lời rap tiếng Việt khiến việc đánh giá định lượng chính xác hiệu suất sáng tạo của mô hình trở nên khó khăn.

Định hướng phát triển

Mặc dù đã đạt được những thành tựu nhất định, dự án vẫn còn nhiều tiềm năng để phát triển và cải thiện trong tương lai:

- Mở rộng và đa dạng hóa bộ dữ liệu
- Nâng cao trải nghiệm người dùng và giao diện (UI/UX)
- Cải tiến mô hình sinh lời
- Đánh giá toàn diện hơn
- Tối ưu hóa triển khai và hiệu năng

TÀI LIỆU THAM KHẢO

[1] T. Malmi, A. Takala, H. Toivonen, A. Gionis, and T. Raiko, DopeLearning: A Computational Approach to Rap Lyrics Generation, Aalto University, Helsinki, 2016. [Online]. Có sẵn tại: https://arxiv.org/pdf/1505.04771

[2] Lubik, O. Branavan, J. Lovell, S. Fels, and O. Michalewicz, LyricJam: A system for generating lyrics for live instrumental music, University of Waterloo, 2021. [Online]. Có sẵn tại: https://arxiv.org/pdf/2106.01960

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI, 2019.

[4] N. N. Van, "NlpHUST/gpt2-vietnamese," Hugging Face, [Trực tuyến]. Có sẵn tại: https://huggingface.co/NlpHUST/gpt2-vietnamese. [Truy cập ngày 20 tháng 3 năm 2025]

[5] Loshchilov, I. & Hutter, F., 2019. Decoupled Weight Decay Regularization. In: International Conference on Learning Representations (ICLR 2019). Có sẵn tại: https://arxiv.org/pdf/1711.05101

[6] Radford, A., Wu, J., Child, R., et al. (2019). Language Models are Unsupervised Multitask Learners. OpenAI. (GPT-2 foundational paper)