|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA CÔNG NGHỆ THÔNG TIN**    **Đồ án**  **Khoa học dữ liệu ứng dụng**  Dự án hướng ứng dụng  Ứng dụng tóm tắt podcast  **LỚP: CQ2021/21**  Giảng viên:  Lê Ngọc Thành  Lê Nhựt Nam  Võ Nam Thục Đoan  **Sinh viên thực hiện:**   |  |  | | --- | --- | | **MSSV** | **Tên thành viên** | | 21120348 | Nguyễn Trần Trình | | 21120441 | Dương Huỳnh Anh Duy | | 21120464 | Nguyễn Quốc Hưng |   Học kỳ II – Năm học 2024-2025 |

**Mục lục**

[I. Tổng quan dự án 2](#_Toc200400899)

[II. Giới thiệu 2](#_Toc200400900)

[III. Kiến trúc hệ thống 2](#_Toc200400901)

[1. Tầng thu thập và xử lý dữ liệu 2](#_Toc200400902)

[2. Tầng xử lý ngôn ngữ tự nhiên 2](#_Toc200400903)

[3. Tầng giao diện người dùng 3](#_Toc200400904)

[IV. Phương pháp luận và triển khai 3](#_Toc200400905)

[1. Thu thập và tiền xử lý dữ liệu 3](#_Toc200400906)

[2. Triển khai mô hình tóm tắt 3](#_Toc200400907)

[3. Giao diện người dùng 3](#_Toc200400908)

[V. Kết quả và đánh giá 3](#_Toc200400909)

[1. Chỉ số đánh giá 3](#_Toc200400910)

[2. Đánh giá chất lượng 3](#_Toc200400911)

[3. Demo ứng dụng 3](#_Toc200400912)

[VI. Thách thức và giải pháp 3](#_Toc200400913)

[1. Thách thức 3](#_Toc200400914)

[2. Giải pháp 3](#_Toc200400915)

[VII. Kết luận 3](#_Toc200400916)

[VIII. Hướng phát triển tương lai 3](#_Toc200400917)

[IX. Tài liệu tham khảo 3](#_Toc200400918)

# Tổng quan dự án

**Tên dự án**: Podcast Summarization App.

**Thời gian thực hiện**: 12 tuần.

**Thành viên nhóm**:

* 21120348 - Nguyễn Trần Trình.
* 21120441 - Dương Huỳnh Anh Duy (Nhóm trưởng).
* 21120464 - Nguyễn Quốc Hưng.

**Mục tiêu chính**: Phát triển ứng dụng sử dụng AI để tự động tóm tắt nội dung podcast, giúp người dùng nắm bắt thông tin chính nhanh chóng mà không cần nghe toàn bộ nội dung.

# Giới thiệu

Podcast đã trở thành phương tiện phổ biến để tiếp nhận thông tin và giải trí. Tuy nhiên, thời lượng dài của nhiều podcast gây khó khăn cho người dùng muốn nắm bắt nội dung chính một cách nhanh chóng. Dự án này giải quyết thách thức đó bằng cách phát triển một ứng dụng sử dụng kỹ thuật NLP và học máy để tạo bản tóm tắt tự động từ nội dung podcast.

# Kiến trúc hệ thống

Hệ thống được xây dựng với kiến trúc 3 tầng chính.

## Tầng thu thập và xử lý dữ liệu

Quy trình tự động hóa podcast bao gồm hai giai đoạn chính:

* **Tải xuống**: Tự động lấy tệp âm thanh từ RSS feed, kiểm tra và bỏ qua các tập đã có, sau đó lưu trữ chúng.
* **Chuyển đổi văn bản**: Sử dụng AI (Whisper) để chuyển đổi các tệp âm thanh thành văn bản, kiểm tra và bỏ qua các bản đã chuyển đổi, sau đó lưu lại dưới dạng text.

## Tầng xử lý ngôn ngữ tự nhiên

Triển khai nhiều phương pháp tóm tắt:

* **Mô hình local**:
  + BART-large.
  + Pegasus-dm.
  + T5-base.
* **Sử dụng API**:
  + Cohere.

## Tầng giao diện người dùng

# Phương pháp luận và triển khai

Trước tiên cần đảm bảo đã cài đặt theo yêu cầu của requirements.txt



## Thu thập và tiền xử lý dữ liệu

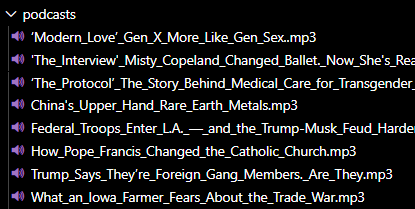


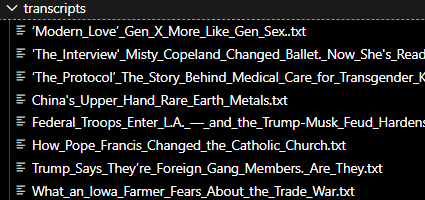
Quá trình tải podcast diễn ra tự động qua các bước chính sau:

1. **Phân tích RSS Feed**: Chương trình đọc RSS feed URL để lấy thông tin và danh sách các tập podcast gần nhất.
2. **Lặp qua từng tập**: Với mỗi tập podcast tìm thấy, chương trình trích xuất tiêu đề và đường dẫn tệp âm thanh (MP3). Tiêu đề được làm sạch để sử dụng làm tên tệp.
3. **Kiểm tra trùng lặp**: Trước khi tải, chương trình kiểm tra xem tệp MP3 của tập đó đã tồn tại trên máy tính chưa. Nếu có, tập đó sẽ bị bỏ qua để tránh tải lại.
4. **Tải và lưu tệp**: Nếu tệp chưa tồn tại, chương trình sẽ tải tệp MP3 từ đường dẫn đã lấy được. Tệp sau đó được lưu vào thư mục chỉ định với tên đã làm sạch.
5. **Xử lý lỗi**: Toàn bộ quá trình tải được giám sát để bắt và thông báo các lỗi có thể xảy ra (ví dụ: lỗi mạng, đường dẫn không hợp lệ), đảm bảo chương trình không bị dừng đột ngột.

Quá trình chuyển đổi âm thanh podcast thành văn bản được thực hiện như sau:

1. **Khởi tạo mô hình**: Tải và chuẩn bị mô hình Whisper (ví dụ: "base") để nhận diện giọng nói.
2. **Duyệt tệp MP3**: Quét qua thư mục chứa các tệp âm thanh podcast.
3. **Kiểm tra & Bỏ qua**: Nếu bản văn bản (transcript) của một tập đã tồn tại, chương trình sẽ bỏ qua tập đó.
4. **Chuyển đổi & Lưu**: Sử dụng mô hình Whisper để chuyển đổi âm thanh thành văn bản, sau đó lưu kết quả vào một tệp .txt tương ứng.
5. **Xử lý lỗi**: Bắt và thông báo các lỗi có thể xảy ra trong quá trình transcribe.





## Triển khai mô hình tóm tắt



Trọng tâm của ứng dụng, nơi các mô hình học máy được sử dụng để chuyển đổi nội dung văn bản dài (transcript của podcast) thành các bản tóm tắt ngắn gọn. Quá trình này được thiết kế để hỗ trợ cả mô hình chạy cục bộ và các dịch vụ tóm tắt dựa trên API, đồng thời quản lý hiệu quả tài nguyên và lỗi.

### Lựa chọn và khởi tạo mô hình

Ứng dụng hỗ trợ nhiều loại mô hình tóm tắt khác nhau, được định nghĩa trong ‘*main.py*’.

**Mô hình cục bộ (Local Models)**:

* ‘*local\_summarizer.py*’ chịu trách nhiệm tải các mô hình tóm tắt của **Hugging Face** (BART, Pegasus, T5) từ **AutoTokenizer** và **AutoModelForSeq2SeqLM**.
* Khi khởi tạo, LocalSummarizer sẽ tải tokenizer và mô hình tương ứng dựa trên model\_key được chọn (‘*bart-large*’, ‘*pegasus-dm*’, ‘*t5-base*’).
* Mô hình sẽ được chuyển sang thiết bị **cuda** (GPU) nếu có, nếu không sẽ dùng cpu.

**Mô hình dựa trên API Web (Web API Model - Cohere):**

* ‘*web\_summarizer.py*’ tích hợp với **Cohere API**.
* Một đối tượng cohere.Client được khởi tạo bằng COHERE\_API\_KEY để thiết lập kết nối với dịch vụ Cohere.
* Cohere sử dụng mô hình "command" để thực hiện tóm tắt.

### Xử lý văn bản đầu vào và chiến lược tóm tắt

Để xử lý các văn bản dài (transcript của podcast), ứng dụng áp dụng chiến lược chia nhỏ và tóm tắt theo từng phần.

**Chia nhỏ văn bản (Chunking)**:

* Trong ‘*LocalSummarizer.py*’ văn bản được mã hóa thành tokens và sau đó chia thành các đoạn nhỏ hơn, mỗi đoạn không vượt quá max\_input\_length của mô hình (1024 cho BART và Pegasus, 512 cho T5).
* Đối với Cohere API văn bản được chia thành các đoạn dựa trên số lượng ký tự (MAX\_CHUNK\_CHARS, mặc định 3000 ký tự) để tối ưu số lượng yêu cầu API và quản lý giới hạn. Các đoạn quá nhỏ có thể được gộp vào đoạn trước đó.

**Tóm tắt đa giai đoạn (Multi-pass summarization - Local Summarizer)**:

* Nếu văn bản gốc quá dài và phải chia thành nhiều chunks, ‘*LocalSummarizer.py*’ sẽ thực hiện tóm tắt hai lần.
* Giai đoạn đầu (first\_pass): Tóm tắt từng chunk riêng lẻ.
* Giai đoạn cuối (nếu cần): Các bản tóm tắt từ first\_pass được gộp lại, sau đó lại được chia thành chunks mới và tóm tắt lần nữa để tạo ra bản tóm tắt cuối cùng.

### Thực thi tóm tắt và quản lý yêu cầu

Hàm **summarize\_with\_model** trong ‘*main.py*’ là điểm điều phối chính cho việc gọi các mô hình tóm tắt.

**Tóm tắt cục bộ**:

* Hàm **summarize\_hf** trong ‘*local\_summarizer.py*’ được gọi, truyền vào văn bản, khóa mô hình, độ dài đầu vào tối đa và độ dài tóm tắt mong muốn.
* Mô hình generate được sử dụng để tạo tóm tắt với các tham số như max\_length, min\_length, num\_beams và early\_stopping.

**Tóm tắt qua API (Cohere)**:

* Hàm **summarize\_with\_cohere** trong ‘*web\_summarizer.py*’ được gọi.
* Các yêu cầu co.summarize được gửi cho từng text\_chunk với các tham số như model, length, format, temperature và additional\_command.
* Xử lý giới hạn tốc độ và thử lại: summarize\_chunk\_with\_cohere bao gồm logic thử lại khi gặp lỗi 429 (quá giới hạn tốc độ) và sử dụng time.sleep() để chờ giữa các yêu cầu, đảm bảo tuân thủ giới hạn API (5 calls/min).

**Ghi nhận kết quả**:

* Đối với mỗi lần tóm tắt, thời gian xử lý được ghi lại. Kết quả bao gồm ID của mô hình, thời gian thực hiện, bản tóm tắt và trạng thái (success hoặc failed).
* Các lỗi phát sinh trong quá trình tóm tắt (cục bộ hoặc API) đều được bắt và thông báo trong trường summary với trạng thái failed.

## Giao diện người dùng

# Kết quả và đánh giá

## Chỉ số đánh giá

## Đánh giá chất lượng

## Demo ứng dụng

# Thách thức và giải pháp

## Thách thức

## Giải pháp

# Kết luận

# Hướng phát triển tương lai

# Tài liệu tham khảo

Lewis, M. et al. (2019). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation".

Raffel, C. et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer".

Radford, A. et al. (2022). "Robust Speech Recognition via Large-Scale Weak Supervision".

Tài liệu API Cohere: <https://docs.cohere.com>

Repository GitHub: <https://github.com/NguyenQuocHung47/ADS_Project>