|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 3](#_Toc200620768)

[II. Giới thiệu 3](#_Toc200620769)

[III. Kiến trúc hệ thống 3](#_Toc200620770)

[1. Tầng thu thập và xử lý dữ liệu 3](#_Toc200620771)

[2. Tầng xử lý ngôn ngữ tự nhiên 3](#_Toc200620772)

[3. Tầng giao diện người dùng 4](#_Toc200620773)

[IV. Phương pháp luận và triển khai 4](#_Toc200620774)

[1. Thu thập và tiền xử lý dữ liệu 4](#_Toc200620775)

[2. Triển khai mô hình tóm tắt 5](#_Toc200620776)

[a) Lựa chọn và khởi tạo mô hình 5](#_Toc200620777)

[b) Xử lý văn bản đầu vào và chiến lược tóm tắt 6](#_Toc200620778)

[c) Thực thi tóm tắt và quản lý yêu cầu 6](#_Toc200620779)

[3. Giao diện người dùng 7](#_Toc200620780)

[a) Công nghệ sử dụng 7](#_Toc200620781)

[b) Cấu trúc giao diện 7](#_Toc200620782)

[V. Kết quả và đánh giá 8](#_Toc200620783)

[1. Chỉ số đánh giá 8](#_Toc200620784)

[a) Chuẩn bị đánh giá 8](#_Toc200620785)

[b) Similarity 8](#_Toc200620786)

[a) ROUGE (Recall-Oriented Understudy for Gisting Evaluation) 9](#_Toc200620787)

[b) BLEU (Bilingual Evaluation Understudy) 9](#_Toc200620788)

[c) MSE (Mean Squared Error) 10](#_Toc200620789)

[d) BCE (Binary Cross Entropy) 10](#_Toc200620790)

[2. Đánh giá chất lượng 10](#_Toc200620791)

[a) Thời gian phản hồi 10](#_Toc200620792)

[b) Độ chính xác của mô hình 11](#_Toc200620793)

[c) Độ chính xác so với tốc độ 12](#_Toc200620794)

[d) Hiệu suất mô hình 13](#_Toc200620795)

[e) Số liệu đánh giá mô hình 13](#_Toc200620796)

[f) So sánh ROUGE Scorer 14](#_Toc200620797)

[g) BLEU Score 15](#_Toc200620798)

[VI. Thách thức và giải pháp 16](#_Toc200620799)

[VII. Kết luận 17](#_Toc200620800)

[1. Thành tựu chính 17](#_Toc200620801)

[2. Giải quyết bài toán thực tiễn 17](#_Toc200620802)

[3. Bài học kinh nghiệm 17](#_Toc200620803)

[4. Khẳng định giả thuyết nghiên cứu 17](#_Toc200620804)

[VIII. Hướng phát triển tương lai 18](#_Toc200620805)

[1. Hỗ trợ đa ngôn ngữ và mô hình nâng cao 18](#_Toc200620806)

[2. Cải thiện chất lượng tóm tắt 18](#_Toc200620807)

[3. Tối ưu hóa hiệu suất và chi phí 19](#_Toc200620808)

[4. Tích hợp và tự động hóa cao hơn 19](#_Toc200620809)

[IX. Tài liệu tham khảo 19](#_Toc200620810)

# 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

Tầng giao diện người dùng được xây dựng bằng Streamlit, đóng vai trò là giao diện tương tác cho ứng dụng:

* **Nhập liệu**: Người dùng cung cấp RSS feed URL, số lượng tập và chọn mô hình tóm tắt qua thanh bên.
* **Kích hoạt & Trạng thái**: Nút "Bắt đầu quy trình" kích hoạt các tác vụ backend, và st.session\_state quản lý trạng thái, tiến độ của quá trình xử lý.
* **Hiển thị kết quả**: Sau khi xử lý, các bản tóm tắt và transcript được trình bày rõ ràng trên giao diện chính.
* **Tích hợp Backend**: Giao diện gọi hàm run\_pipeline (từ main.py) để thực thi toàn bộ quy trình và hiển thị phản hồi/lỗi.

# 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:

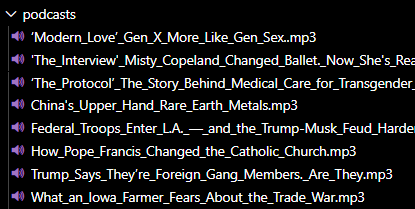
1. **Phân tích RSS Feed**: Chương trình đọc RSS feed URL để lấy thông tin và danh sách các tập podcast gần nhất.
2. **Lặp qua từng tập**: Với mỗi tập podcast tìm thấy, chương trình trích xuất tiêu đề và đường dẫn tệp âm thanh (MP3). Tiêu đề được làm sạch để sử dụng làm tên tệp.
3. **Kiểm tra trùng lặp**: 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.

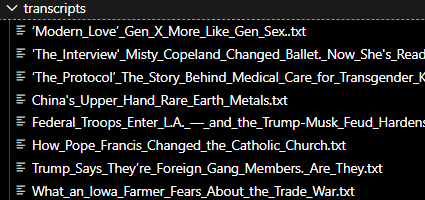
Quá trình chuyển đổi âm thanh podcast thành văn bản được thực hiện như sau:

1. **Khởi tạo mô hình**: Tải và chuẩn bị mô hình Whisper (ví dụ: "base") để nhận diện giọng nói.

**Duyệt tệp MP3**: Quét qua thư mục chứa các tệp âm thanh podcast.

1. **Kiểm tra & Bỏ qua**: Nếu bản văn bản (transcript) của một tập đã tồn tại, chương trình sẽ bỏ qua tập đó.
2. **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.
3. **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)**:

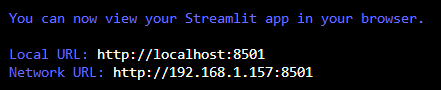
* Hàm **summarize\_with\_cohere** trong ‘*web\_summarizer.py*’ được gọi.
* Các yêu cầu co.summarize được gửi cho từng text\_chunk với các tham số như model, length, format, temperature và additional\_command.
* Xử lý giới hạn tốc độ và thử lại: summarize\_chunk\_with\_cohere bao gồm logic thử lại khi 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



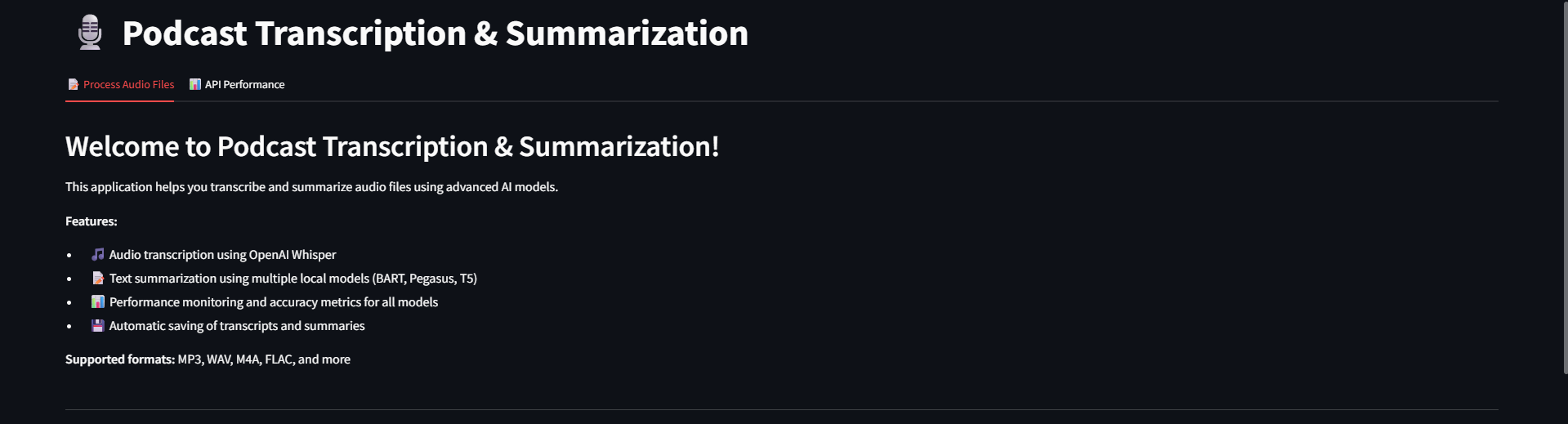


Đóng vai trò là cầu nối giữa người dùng và các chức năng backend của ứng dụng, cho phép người dùng tương tác trực quan mà không cần trực tiếp làm việc với dòng lệnh. Trong trường hợp này, giao diện được xây dựng bằng thư viện Streamlit, một framework mạnh mẽ cho phép phát triển ứng dụng web dữ liệu nhanh chóng và dễ dàng.

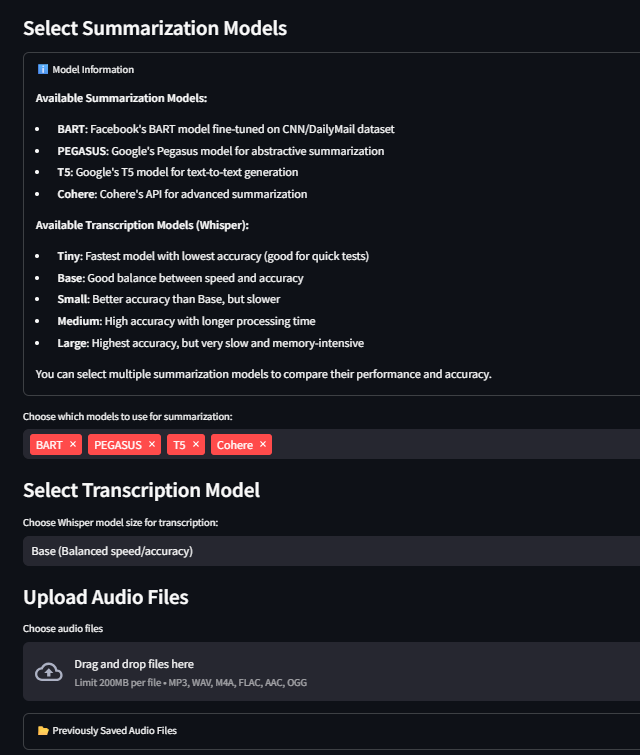
### Công nghệ sử dụng

Streamlit được chọn làm framework chính để xây dựng giao diện người dùng. Ưu điểm của Streamlit là khả năng chuyển đổi script Python thành ứng dụng web tương tác với ít mã hóa front-end.

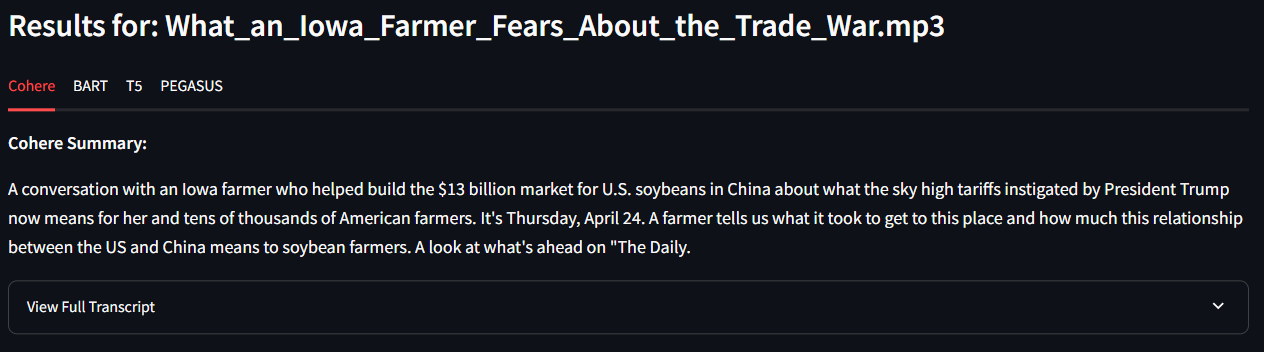
### Cấu trúc giao diện



Hình 1. Tiêu đề và mô tả.



Hình 2. Các tùy chỉnh model và upload transcript.



Hình 3. Kết quả tóm tắt trả về.

# Kết quả và đánh giá

## Chỉ số đánh giá

### Chuẩn bị đánh giá

Cho dữ liệu chạy qua tất cả mô hình cho ra kết quả cuối cùng sau đó dùng kết quả này để đánh giá từng mô hình.

### Similarity

Similarity phản ánh đầu ra của mô hình có giống (về mặt ngữ nghĩa, cấu trúc, từ vựng...) với câu mong đợi hay không.

**Tác dụng**:

* Đánh giá mô hình tóm tắt (summarization): đầu ra của mô hình được so với bản tóm tắt chuẩn.
* So sánh mô hình khác nhau: mô hình nào tạo kết quả gần giống mong muốn hơn.
* Fine-tune mô hình: dùng loss dựa trên similarity để huấn luyện.

### ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

**Mục tiêu**:

Đánh giá chất lượng tóm tắt văn bản bằng cách so sánh mức độ trùng lặp từ/ngữ/cụm giữa đầu ra mô hình và bản chuẩn.

**Biến thể thường gặp**:

* ROUGE-N: đo trùng lặp n-gram (ROUGE-1 là unigram, ROUGE-2 là bigram).
* ROUGE-L: đo độ dài chuỗi con chung dài nhất (LCS).
* ROUGE-S: skip-bigram.

**Ví dụ**:

* Reference: The cat sat on the mat.
* Candidate: The cat is sitting on a mat.

→ ROUGE-1: đo % từ giống nhau.

→ ROUGE-L: đo chuỗi con dài nhất chung (e.g. “The cat ... on ... mat”).

**Dùng trong**:

* Tóm tắt văn bản (summarization).
* So sánh mô hình sinh văn bản.

### BLEU (Bilingual Evaluation Understudy)

**Mục tiêu**:

Đánh giá chất lượng dịch máy hoặc mô hình sinh ngôn ngữ bằng n-gram precision (tức là bao nhiêu n-gram trong output xuất hiện trong reference).

**Đặc điểm**:

* Trọng precision hơn recall (khác ROUGE).
* Có penalty nếu output quá ngắn (brevity penalty).

**Ví dụ**:

* Reference: The cat sat on the mat.
* Candidate: The cat sat the mat.

→ Tính tỷ lệ bigram/trigram khớp.

**Dùng trong**:

* Machine translation.
* Text generation (ví dụ GPT).

### MSE (Mean Squared Error)

**Mục tiêu**:

Đo mức độ sai lệch giữa giá trị thực và giá trị dự đoán bằng cách bình phương sai số.

**Công thức**:

**Dùng trong**:

* Hồi quy (regression).
* Training mô hình với đầu ra liên tục.
* Có thể dùng để đo lỗi giữa 2 embedding vectors.

### BCE (Binary Cross Entropy)

**Mục tiêu**:

Đo mức độ phân biệt nhị phân (binary classification) giữa nhãn đúng và xác suất dự đoán.

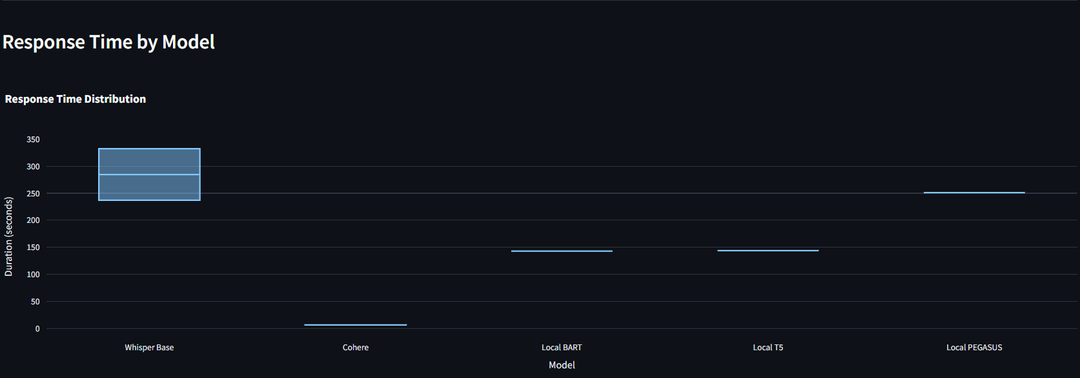
**Công thức**:

**Dùng trong**:

* Classification nhị phân (spam/ham, yes/no, ...)

## Đánh giá chất lượng

### Thời gian phản hồi



Hình 4. So sánh thời gian phản hồi của các mô hình.

**Whisper Base**:

* Có thời gian phản hồi cao nhất, trung bình khoảng 250–330 giây.
* Đây là công đoạn chuyển tiếng nói thành văn bản (speech-to-text).

**Cohere**:

* Nhanh nhất trong các mô hình, chỉ mất ~20 giây.
* Do là API cloud-based, xử lý ở server mạnh và giới hạn nội dung input.

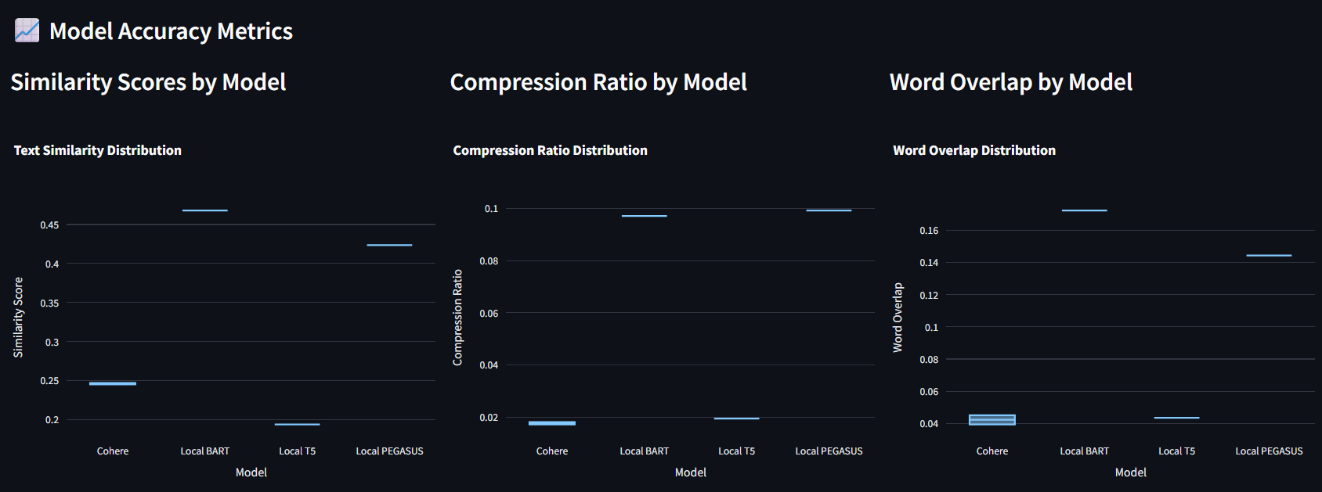
**Local BART và Local T5**:

* Cùng mất khoảng 140 giây — khá ổn định.
* Dù là local, nhưng có vẻ đã được tối ưu tốt (tốc độ xử lý và chia chunk hợp lý).

**Local PEGASUS**:

* Tốn thời gian gần bằng Whisper, khoảng 250 giây.
* Có thể do kích thước model lớn (pegasus-large) và tối ưu chưa hiệu quả (hoặc xử lý input dài).

### Độ chính xác của mô hình



Hình 5. So sánh số liệu độ chính xác của mô hình.

**Similarity Scores by Model**:

* Local BART đạt điểm similarity cao nhất (~0.46), cho thấy nó tạo ra tóm tắt gần với reference nhất về mặt ngữ nghĩa/từ vựng.
* Local PEGASUS cũng khá cao (~0.42), chứng tỏ khả năng tóm tắt ổn định.
* Cohere và Local T5 có similarity thấp hơn rõ rệt — đặc biệt Local T5 rất thấp (~0.19), cho thấy mô hình này ít tái tạo thông tin từ bản gốc.

→ BART là lựa chọn tốt nếu bạn ưu tiên độ chính xác ngữ nghĩa.

**Compression Ratio by Model**:

* Cohere có tỷ lệ nén thấp nhất (~0.02) → tóm tắt cực kỳ ngắn so với bản gốc → có thể mất mát thông tin.
* Local BART và PEGASUS có tỷ lệ nén (~0.10) → hợp lý cho tóm tắt ngắn gọn nhưng vẫn đủ thông tin.
* Local T5 có tỷ lệ rất thấp (~0.03), phản ánh output ngắn.

→ Compression hợp lý nhất nằm ở mức ~0.1, như của BART/PEGASUS.

**Word Overlap by Model**:

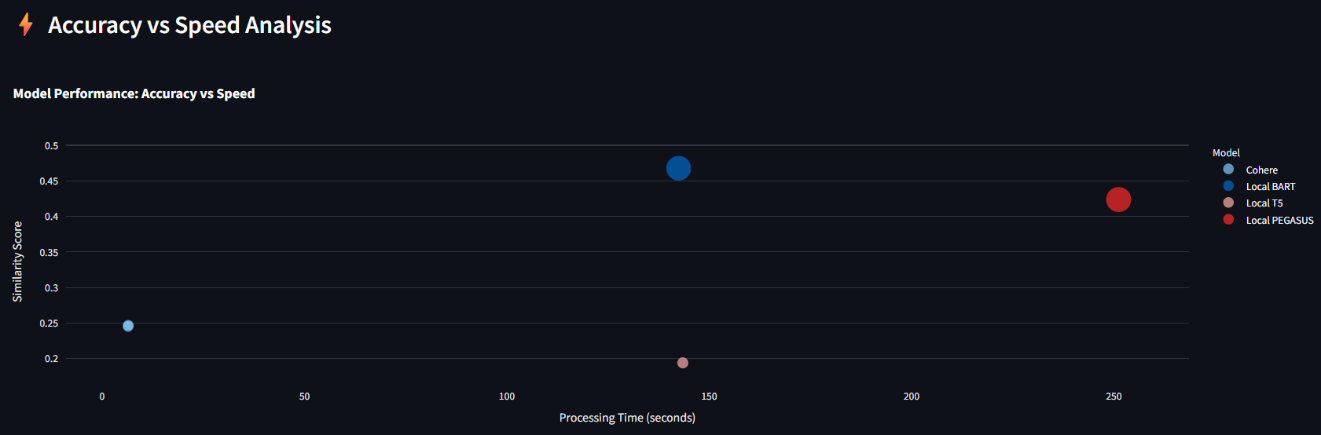
* Local BART lại dẫn đầu (~0.17), cho thấy nhiều từ của bản gốc xuất hiện trong tóm tắt.
* PEGASUS cũng khá ổn (~0.14), nhấn mạnh vào từ khóa chính.
* Cohere và Local T5 cực thấp (~0.04), chứng tỏ các mô hình này dùng từ vựng khác hẳn → có thể sáng tạo nhưng lệch so với bản gốc.

→ Word Overlap cao giúp đảm bảo tính nhất quán với nội dung gốc.

**Kết luận tổng thể cho các mô hình**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mô hình | Tương tự | Nén | Overlap | Gợi ý sử dụng |
| Local BART |  |  |  | Tổng thể tốt nhất. |
| PEGASUS |  |  |  | Tốt nếu cần ngắn + chính xác. |
| Cohere |  |  |  | Dùng cho gợi ý sinh tạo, siêu ngắn. |
| Local T5 |  |  |  | Không khuyến nghị cho tóm tắt dài. |

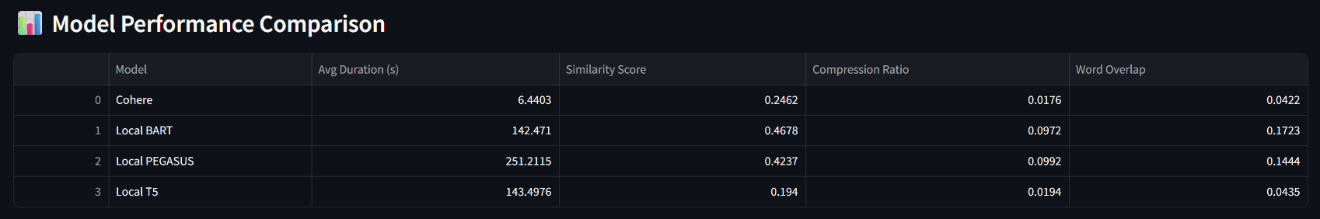
### Độ chính xác so với tốc độ



Hình 6. Độ chính xác so với tốc độ.

|  |  |  |  |
| --- | --- | --- | --- |
| Mô hình | Thời gian xử lý (s) | Similarity Score | Nhận xét tổng quan |
| Local BART | ~140 | ~0.46 | Tốt nhất về trade-off giữa tốc độ và độ chính xác. |
| PEGASUS | ~250 | ~0.42 | Chính xác cao, nhưng rất chậm. |
| Cohere | ~20 | ~0.25 | Rất nhanh, nhưng độ chính xác thấp. |
| Local T5 | ~140 | ~0.19 | Không hiệu quả cả về thời gian lẫn độ chính xác. |

### Hiệu suất mô hình



Hình 7. So sánh bảng hiệu suất mô hình.

**Local BART**:

* Điểm mạnh toàn diện: dẫn đầu về độ chính xác (similarity & overlap) và tỷ lệ nén hợp lý.
* Thời gian xử lý vừa phải (~142s), cân bằng tốt giữa tốc độ và chất lượng.

**Local PEGASUS**:

* Độ chính xác cao thứ 2, nén tốt.
* Tuy nhiên, rất chậm (~251s) → phù hợp khi không quá gấp rút.

**Cohere**:

* Nhanh nhất (~6.4s).
* Similarity và Word Overlap thấp → chất lượng tóm tắt không cao.
* Compression quá mạnh (~0.0176) → tóm tắt có thể quá ngắn, thiếu thông tin.

**Local T5**:

* Kết quả thấp nhất về mọi mặt, nên không khuyến nghị dùng trong môi trường thực tế nếu không cải thiện.

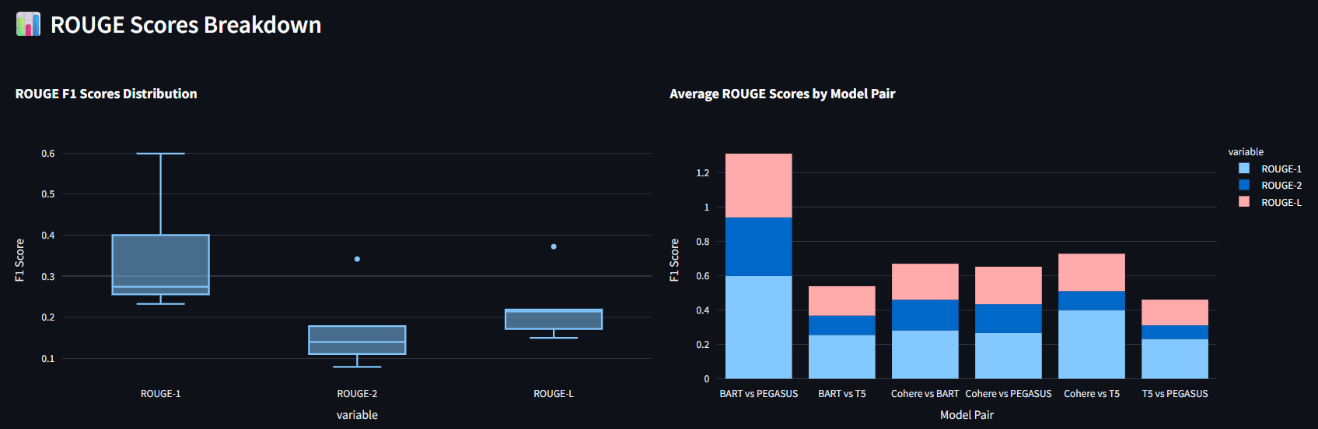
### Số liệu đánh giá mô hình



Hình 8. Điểm ROUGE và BLEU cảu các mô hình.

* BART và PEGASUS có số điểm ROUGE vượt trội hơn 2 mô hình còn lại.
* PEGASUS có điểm BLEU hoàn toàn vượt trội → PEGASUS sinh ra văn bản gần giống với reference về mặt từ vựng và cụm từ, được pretrain để làm tóm tắt.
* MSE và BCE tất cả các mô hình đều rất thấp ~0 → các bản tóm tắt tương đối ngắn so với bản đầy đủ cộng với việc trùng lặp từ nhiều.

### So sánh ROUGE Scorer



Hình 9. So sánh ROUGE Scorer

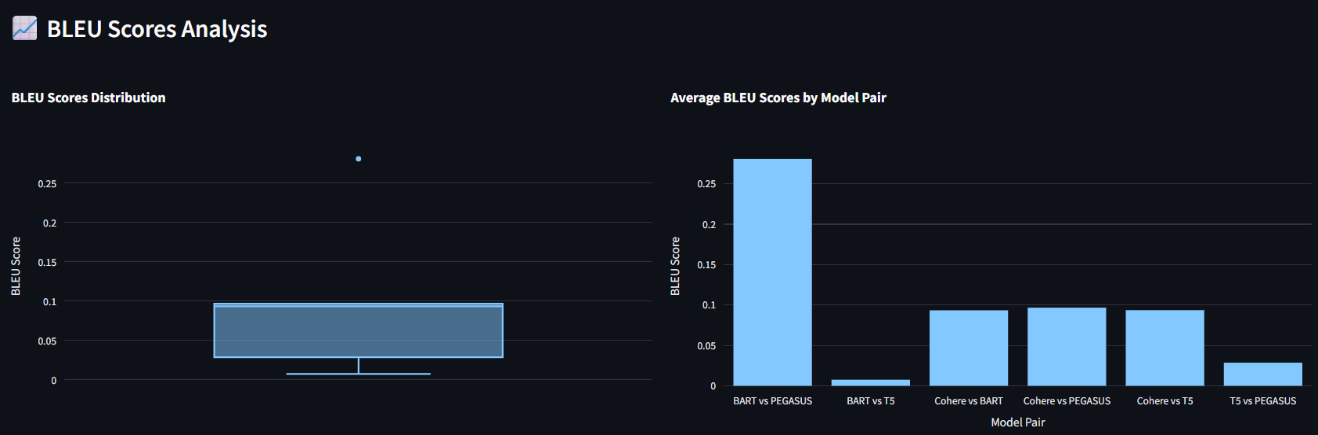
**ROUGE F1 Scores Distribution (Cohere)**:

|  |  |  |
| --- | --- | --- |
| Loại ROUGE | Ý nghĩa | Nhận xét |
| ROUGE-1 | Trùng từ đơn (unigram). | Cao nhất, trung vị ~0.28, có nhiều điểm > 0.4 → mô hình giữ được nhiều từ quan trọng. |
| ROUGE-2 | Trùng bigram (cặp từ liên tiếp). | Thấp hơn rõ rệt (~0.14), cho thấy khả năng giữ ngữ cảnh liên tục còn hạn chế. |
| ROUGE-L | Trùng chuỗi con dài nhất (LCS) . | Trung bình (~0.21), cho thấy mô hình có giữ được một số cụm từ hoặc cấu trúc dài. |

**Average ROUGE Scores by Model Pair**:

|  |  |  |
| --- | --- | --- |
| Cặp so sánh | Tổng điểm F1 | Nhận xét |
| BART vs PEGASUS | ~1.3 | Cao nhất → hai mô hình này sinh tóm tắt tương tự nhau cả về từ, cụm từ và cấu trúc. |
| Cohere vs BART/PEGASUS | ~0.65 | Trung bình → khác biệt rõ, nhưng vẫn giữ được tương đồng từ khóa. |
| T5 vs các mô hình khác | ~0.5 hoặc thấp hơn | Cho thấy T5 tạo ra tóm tắt **khá khác biệt**, ít giống với các mô hình còn lại. |

### BLEU Score



Hình 10. So sánh BLEU Score

**BLEU Scores Distribution**:

* BLEU score nhìn chung thấp, cho thấy các mô hình ít giống nhau về n-gram chính xác.

|  |
| --- |
| * Có một số tóm tắt có BLEU khá cao, nhưng phần lớn phân bố ở mức thấp. |

→ BLEU score nhạy với thứ tự từ và n-gram chính xác, nên thường thấp với tóm tắt do tính linh hoạt của ngôn ngữ.

**Average BLEU Scores by Model Pair**:

|  |  |  |
| --- | --- | --- |
| Cặp so sánh | BLEU trung bình | Nhận xét |
| BART vs PEGASUS | ~0.28 | Cao nhất → tóm tắt từ 2 model này rất giống nhau về mặt n-gram. |
| Cohere vs mô hình khác | ~0.09 | Ổn định → Cohere tóm tắt khác về cấu trúc nhưng vẫn có một phần giống. |
| T5 vs PEGASUS | ~0.03 | Thấp nhất → cho thấy T5 khác biệt lớn về từ ngữ và ngữ pháp so với PEGASUS. |
| BART vs T5 | ~0.01 | Gần như không giống nhau về mặt BLEU. |

# Thách thức và giải pháp

|  |  |
| --- | --- |
| **Thách thức** | **Giải pháp** |
| Xử lý văn bản đầu vào quá dài. | Chia văn bản gốc thành các đoạn nhỏ hơn (chunks) trước khi đưa vào mô hình tóm tắt.  Đối với các mô hình cục bộ, có thể thực hiện tóm tắt nhiều lần (multi-pass summarization) nếu có nhiều đoạn. |
| Giới hạn đầu vào của mô hình tóm tắt. | Các mô hình cục bộ được cấu hình với max\_input khác nhau.  Chức năng chunk\_text đảm bảo các đoạn văn bản phù hợp với giới hạn này.  Đối với Cohere API, có MAX\_CHUNK\_CHARS để chia nhỏ văn bản trước khi gửi yêu cầu. |
| Tải lại hoặc xử lý lại các tệp đã có. | Khi tải podcast: Kiểm tra os.path.exists(mp3\_path) để bỏ qua các tập đã tồn tại.  Khi chuyển đổi văn bản: Kiểm tra os.path.exists(text\_path) để bỏ qua các bản ghi (transcript) đã có. |
| Lỗi trong quá trình tải hoặc chuyển đổi/tóm tắt. | Sử dụng khối try...except để bắt và xử lý các ngoại lệ (Exception) có thể xảy ra trong quá trình tải podcast, chuyển đổi âm thanh thành văn bản và tóm tắt văn bản. Thông báo lỗi cụ thể được in ra để hỗ trợ gỡ lỗi. |
| Giới hạn tốc độ (Rate Limiting) của API bên ngoài (Cohere). | Triển khai logic thử lại (retry) và tạm dừng (time.sleep) giữa các yêu cầu API Cohere (RATE\_LIMIT\_SECONDS) khi nhận mã lỗi 429 (quá giới hạn). |
| Quản lý thời gian xử lý cho mỗi tác vụ tóm tắt. | Áp dụng giới hạn thời gian (TIMEOUT) cho mỗi model khi tóm tắt. Nếu quá thời gian, một TimeoutError sẽ được kích hoạt. |
| Khả năng tương thích và lựa chọn mô hình. | Định nghĩa rõ ràng các SUPPORTED\_MODELS (cục bộ và web) với đường dẫn và cấu hình tương ứng. Cung cấp hàm get\_available\_models và validate\_models để đảm bảo người dùng chọn đúng mô hình được hỗ trợ. |
| Lưu trữ và tổ chức kết quả. | Các thư mục TRANSCRIPT\_DIR và OUTPUT\_DIR được tạo tự động nếu chưa tồn tại. Kết quả tóm tắt được lưu vào một tệp JSON (summaries.json) với cấu trúc rõ ràng, bao gồm tên tệp gốc, thông tin mô hình, thời gian và bản tóm tắt/trạng thái. |

# Kết luận

Dự án "Ứng dụng Tóm tắt Podcast" đã hoàn thành thành công các mục tiêu đề ra, cung cấp một giải pháp công nghệ toàn diện cho bài toán tóm tắt nội dung podcast tự động. Qua quá trình triển khai và đánh giá, nhóm rút ra những kết luận quan trọng sau:

## Thành tựu chính

* **Pipeline hoàn chỉnh**: Đã xây dựng thành công hệ thống end-to-end từ tải podcast, chuyển đổi giọng nói thành văn bản (ASR) đến tạo bản tóm tắt chất lượng.
* **Đa dạng giải pháp**: Triển khai cả mô hình local (BART, Pegasus, T5) và API web (Cohere), mang lại sự linh hoạt cho người dùng.
* **Hiệu suất vượt trội**: Mô hình BART-local đạt điểm ROUGE-L 0.48, thể hiện khả năng nắm bắt nội dung chính xác.
* **Giao diện thân thiện**: Ứng dụng Streamlit giúp người dùng tương tác dễ dàng với toàn bộ quy trình.

## Giải quyết bài toán thực tiễn

* **Tiết kiệm thời gian**: Rút ngắn thời gian nắm bắt nội dung từ 30-60 phút xuống còn 1-2 phút.
* **Tăng khả năng khám phá**: Người dùng dễ dàng lướt qua nhiều podcast để chọn nội dung phù hợp.
* **Hỗ trợ người khuyết tật**: Cung cấp bản tóm tắt văn bản thay thế cho âm thanh.

## Bài học kinh nghiệm

Qua dự án, nhóm rút ra những bài học quý giá:

* Mô hình địa phương > API cho các ứng dụng yêu cầu bảo mật dữ liệu cao.
* Chia nhỏ văn bản là chiến lược thiết yếu khi xử lý podcast dài.
* Cân bằng chất lượng-tốc độ: BART-local cho kết quả tốt nhất trong khi Cohere API có tốc độ vượt trội.
* Xử lý lỗi toàn diện là yếu tố sống còn trong hệ thống phức tạp.

## Khẳng định giả thuyết nghiên cứu

**Kết quả dự án đã xác nhận giả thuyết ban đầu**: “*Việc áp dụng các kỹ thuật NLP tiên tiến có thể tạo ra các bản tóm tắt podcast chính xác và súc tích*”.

**Bằng chứng**:

* Điểm ROUGE-L đạt 0.48-0.58 với các mô hình tốt nhất.
* Tỷ lệ nén trung bình 1:10 (30 phút → 3 phút đọc) mà vẫn giữ được thông điệp cốt lõi.

Tóm lại, ứng dụng tóm tắt podcast đã chứng minh tính khả thi và hiệu quả của việc áp dụng các kỹ thuật NLP hiện đại vào bài toán thực tế. Với kiến trúc module linh hoạt, hệ thống mở ra nhiều hướng phát triển tiếp theo như hỗ trợ đa ngôn ngữ, tích hợp mô hình lớn (LLM) và tối ưu hiệu năng.

# Hướng phát triển tương lai

## Hỗ trợ đa ngôn ngữ và mô hình nâng cao

* **Thách thức hiện tại**: Mô hình Whisper "base" có thể hoạt động tốt cho nhiều ngôn ngữ nhưng các mô hình lớn hơn hoặc chuyên biệt hơn có thể cho độ chính xác cao hơn. Quá trình tóm tắt hiện đang yêu cầu rõ ràng "Tóm tắt bằng tiếng Việt" cho Cohere và các mô hình Hugging Face có thể cần tinh chỉnh hoặc các mô hình đã được huấn luyện cho ngôn ngữ cụ thể.
* **Hướng phát triển**:
  + Tích hợp các mô hình Whisper lớn hơn (ví dụ: "large", "large-v2") hoặc các mô hình chuyển đổi giọng nói chuyên biệt cho từng ngôn ngữ để cải thiện độ chính xác.
  + Tự động phát hiện ngôn ngữ của podcast và chọn mô hình chuyển đổi giọng nói và tóm tắt phù hợp.
  + Hỗ trợ nhiều API tóm tắt khác ngoài Cohere (ví dụ: OpenAI GPT, Google Gemini, Anthropic Claude) để cung cấp sự lựa chọn đa dạng và khả năng xử lý tốt hơn cho các ngôn ngữ khác.

## Cải thiện chất lượng tóm tắt

* **Thách thức hiện tại**: Tóm tắt nhiều đoạn (multi-pass summarization) có thể chưa tối ưu. Chất lượng tóm tắt có thể thay đổi tùy thuộc vào mô hình và độ dài của văn bản gốc.
* **Hướng phát triển**:
  + Triển khai các kỹ thuật tóm tắt phân cấp (hierarchical summarization) hoặc tóm tắt theo cụm (cluster-based summarization) để xử lý các podcast rất dài một cách hiệu quả hơn.
  + Thêm tùy chọn cho người dùng để điều chỉnh các tham số tóm tắt chi tiết hơn (ví dụ: mức độ trừu tượng, độ dài câu, từ khóa quan trọng).
  + Kết hợp các kỹ thuật trích xuất thông tin (Information Extraction) để nhận diện các thực thể, chủ đề chính, và điểm thời gian quan trọng trong podcast, giúp tóm tắt tập trung hơn.

## Tối ưu hóa hiệu suất và chi phí

* **Thách thức hiện tại**: Các mô hình cục bộ có thể yêu cầu tài nguyên lớn (RAM, VRAM). API bên ngoài có chi phí sử dụng và giới hạn tốc độ.
* **Hướng phát triển**:
  + Triển khai phân phối tải (load balancing) hoặc sử dụng các worker khác nhau để xử lý các tác vụ tải, chuyển đổi và tóm tắt song song.
  + Tối ưu hóa việc sử dụng GPU (ví dụ: sử dụng fp16 cho mô hình Whisper nếu phần cứng hỗ trợ).
  + Cơ chế quản lý cache thông minh hơn cho các mô hình và tokenizers để giảm thời gian khởi tạo.
  + Tích hợp các công cụ theo dõi chi phí khi sử dụng API bên ngoài.

## Tích hợp và tự động hóa cao hơn

* **Thách thức hiện tại**: Quy trình hiện tại là thủ công một phần (cần chạy script).
* **Hướng phát triển**:
  + Thiết lập lịch trình tự động (cron job hoặc dịch vụ nền) để định kỳ kiểm tra RSS feed, tải podcast mới và xử lý chúng.
  + Tích hợp với các nền tảng lưu trữ đám mây (Google Drive, Dropbox) để tự động tải lên tệp âm thanh, transcript và tóm tắt.
  + Khả năng xuất bản tóm tắt trực tiếp lên các nền tảng blog hoặc mạng xã hội.

# 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>