# Báo cáo

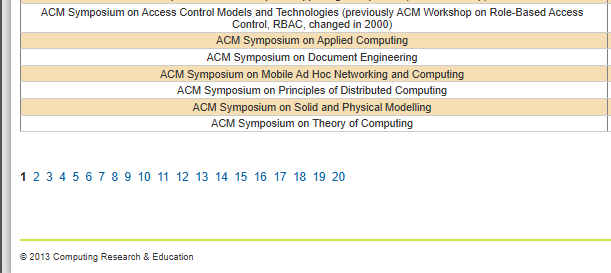
1. **Quy trình crawl và call API**

* **Gồm có 3 bước chính:**

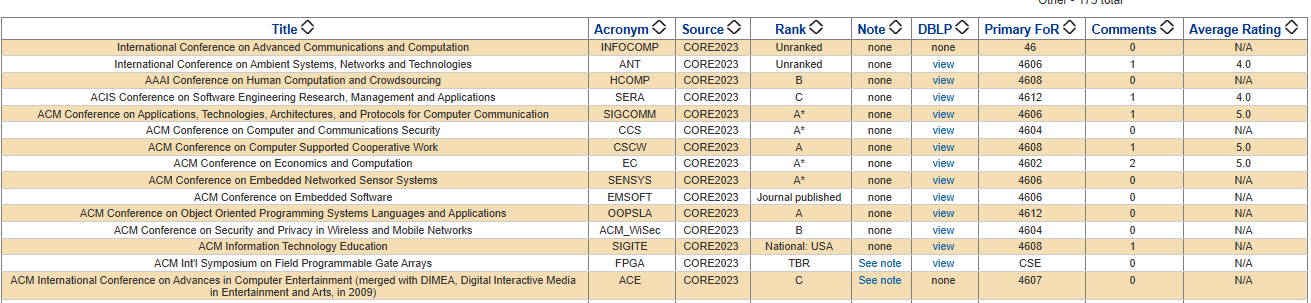
1. Lấy danh sách hội nghị từ Core Portal
2. Tìm và crawl thông tin mỗi hội nghị trên Google
3. Call API, gửi dữ liệu crawl được và nhận kết quả trả về

* **Chi tiết các bước:**

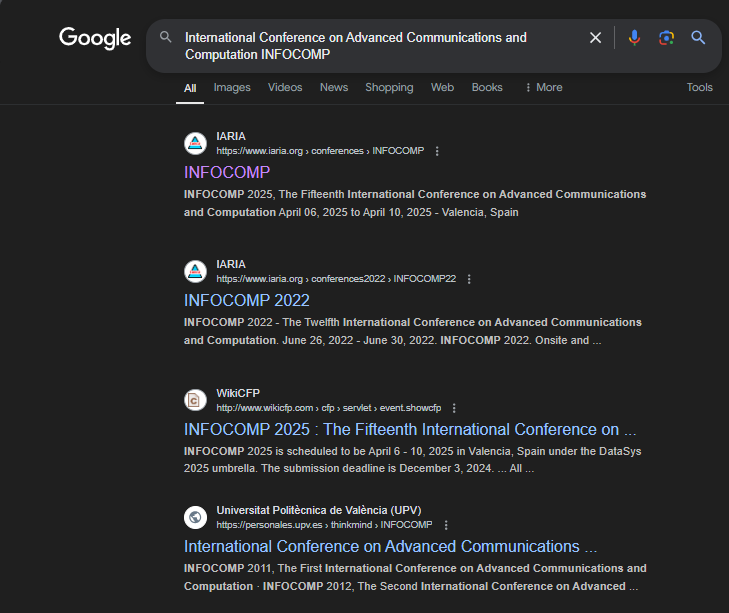
1. Truy cập trang chủ Core Portal, lấy tổng số phân trang



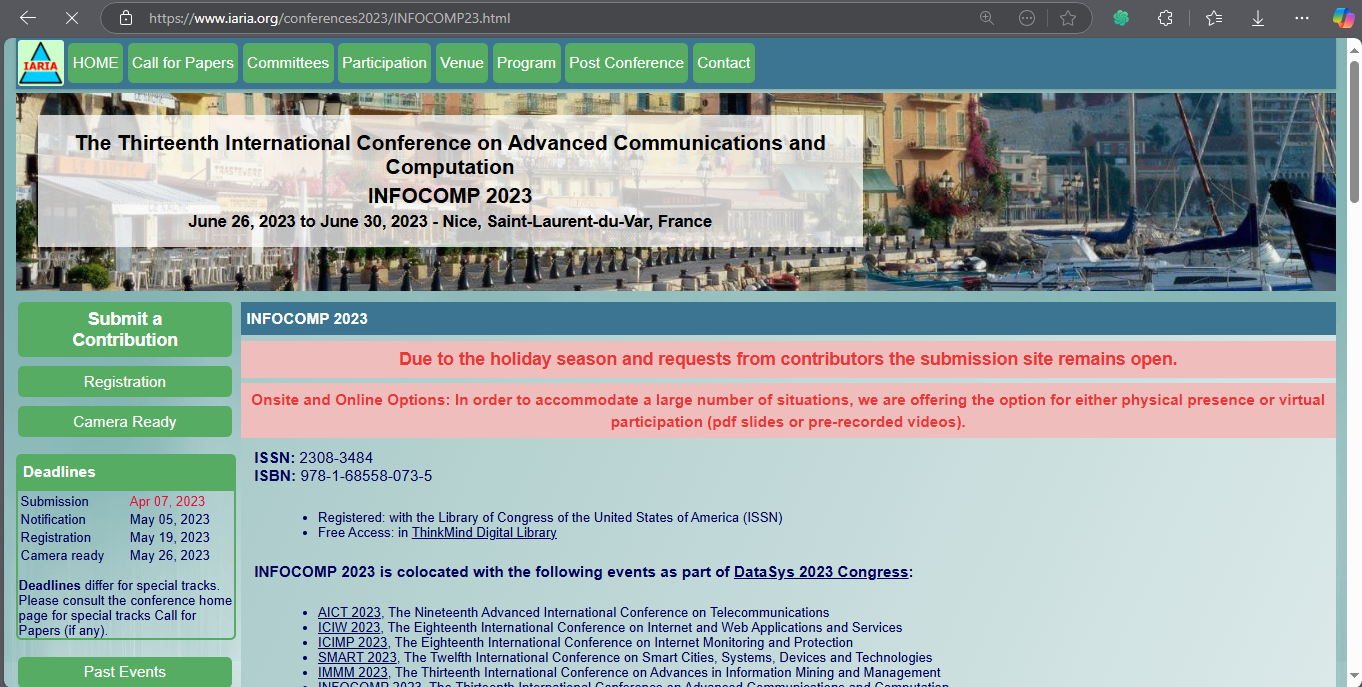
1. Truy cập từng phân trang để lấy toàn bộ danh sách hội nghị cùng các thông tin Acronym, Source, Rank,…..

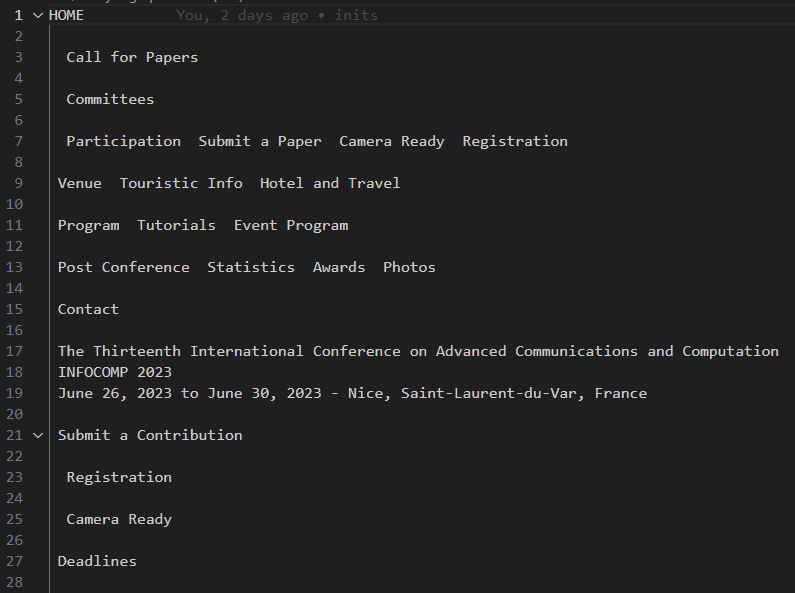


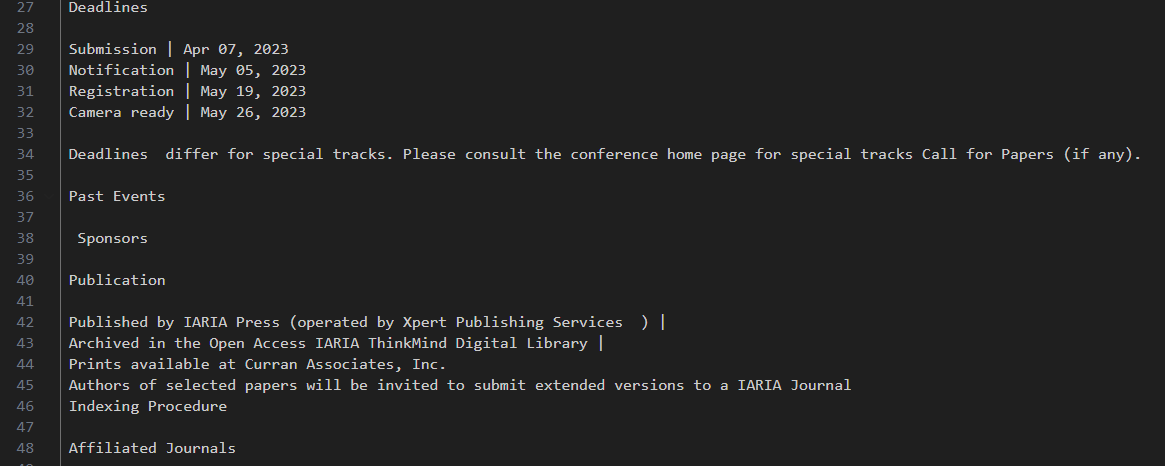
1. Sau khi crawl xong toàn bộ danh sách hội nghị trong tất cả phân trang, tìm thông tin của mỗi hội nghị trên Google với từ khóa: Tên hội nghị + Tên viết tắt (mỗi hội nghị truy cập 4 đường link đầu tiên)



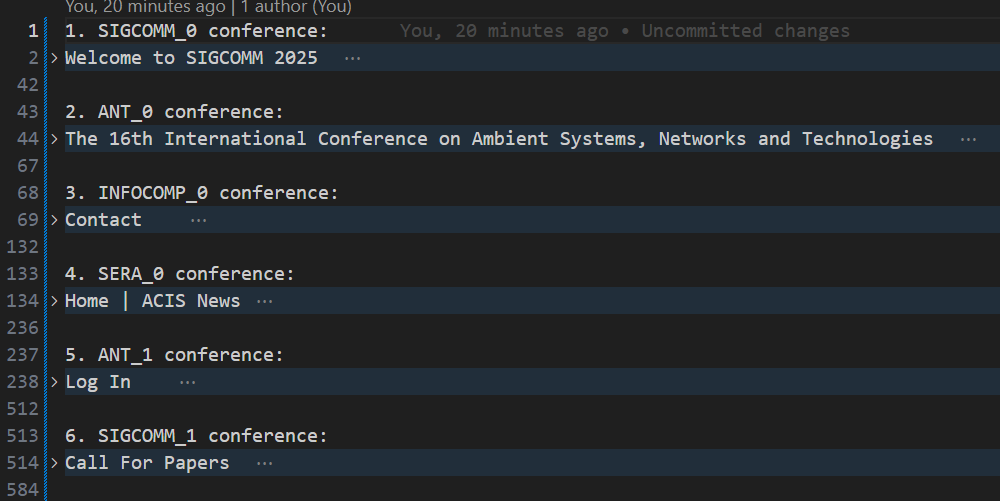
1. Truy cập từng đường link và lấy toàn bộ thông tin HTML của trang, trích xuất loại bỏ các thẻ HTML và chỉ giữ lại các phần văn bản:



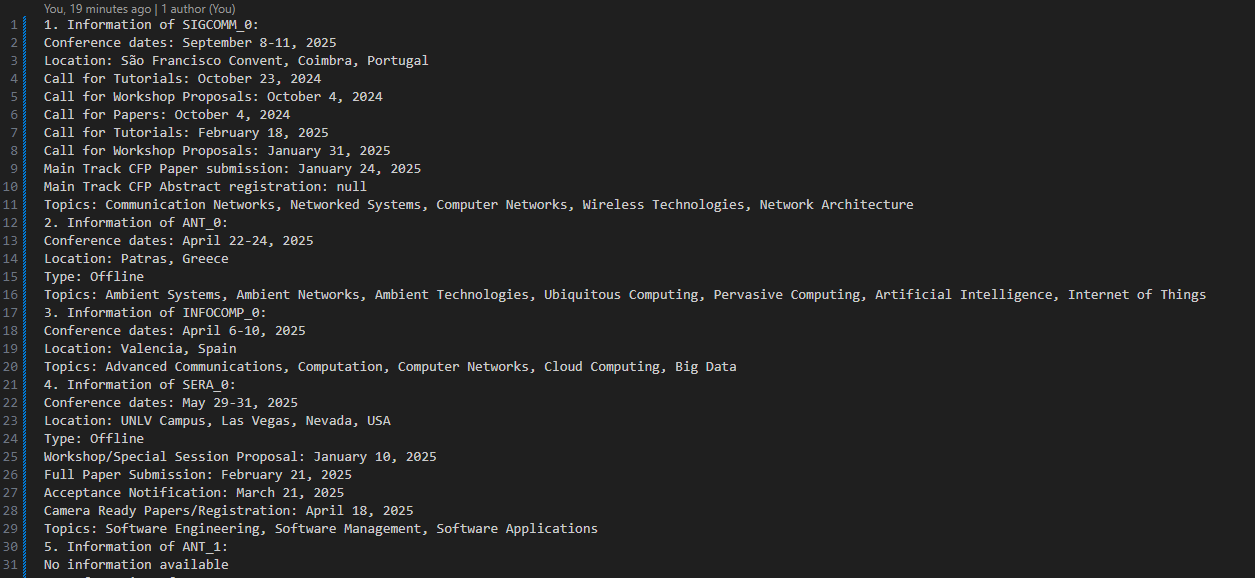




1. Đến khi crawl được một số lượng đường link nhất định, tạo một batch gồm toàn bộ thông tin crawl được từ các đường link để call Gemini API:



1. Nhận kết quả trả về từ Gemini API



1. **Hạn chế trong các phương pháp của nhóm trước trong phần crawl và rút trích thông tin:**
2. **Crawl:**

* Sử dụng thư viện Puppeteer để crawl dữ liệu:

-> dù Puppeteer vẫn hỗ trợ rất tốt nhưng việc crawl chỉ đang được thực hiện trên 1 luồng chạy

-> crawl lâu nếu thực hiện crawl toàn bộ tất cả các hội nghị

-> giải pháp: crawl song song nhiều luồng

* Khi truy cập một đường link để crawl thông tin thì thực hiện tải toàn bộ trang:

-> chuyển hướng và tải trang lâu

-> giải pháp: tắt các tài nguyên không cần thiết khi truy cập trang web

1. **Rút trích thông tin:**

* Các thông tin của hội nghị được rút trích dựa trên các từ điển và bộ luật:

-> việc tìm và xây dựng các từ khóa và bộ luật rất mất thời gian

-> không tổng quát cho tất cả các trường hợp

-> dễ bị thất thoát và nhận dạng sai thông tin trong nhiều trường hợp

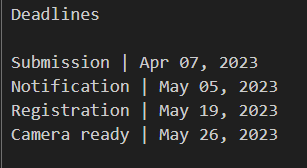
1. **Các công cụ, thư viện, và kỹ thuật tối ưu đã sử dụng:**
2. Crawl **song song** đồng thời **nhiều luồng** thay vì chỉ crawl 1 luồng duy nhất giúp tăng tốc độ crawl
3. Sử dụng thư viện crawl mới là **Playwright** hỗ trợ việc crawl song song nhiều luồng tốt hơn thư viện **Puppeteer** cũ
4. **Tắt các thành phần không cần thiết** khi tải một trang web như image, javascript, font, ads, … giúp tăng thời gian tải trang **thay vì** phải **tải toàn bộ tất cả**
5. Xây dựng các hàm format để **định dạng văn bản** crawl được từ các trang web theo một **cấu trúc rõ ràng**, giúp việc nhận diện và trích xuất các thông tin quan trọng dễ dàng hơn, đặc biệt hữu ích đối với các thẻ table hoặc list (thường chứa thông tin về ngày tháng quan trọng của hội nghị), giúp giữ lại đúng cấu trúc table hoặc list gốc trong trang web.

Ví dụ:

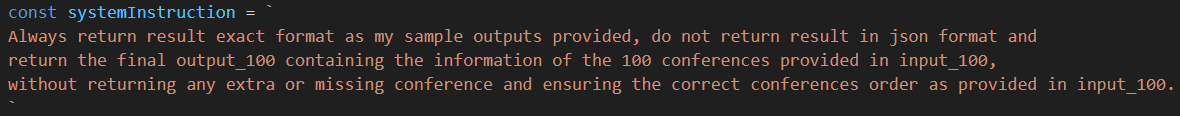
* Nếu chỉ đơn giản tách các thẻ HTML và thay thế thẻ bằng dấu cách:



* Sử dụng các hàm format dữ liệu:

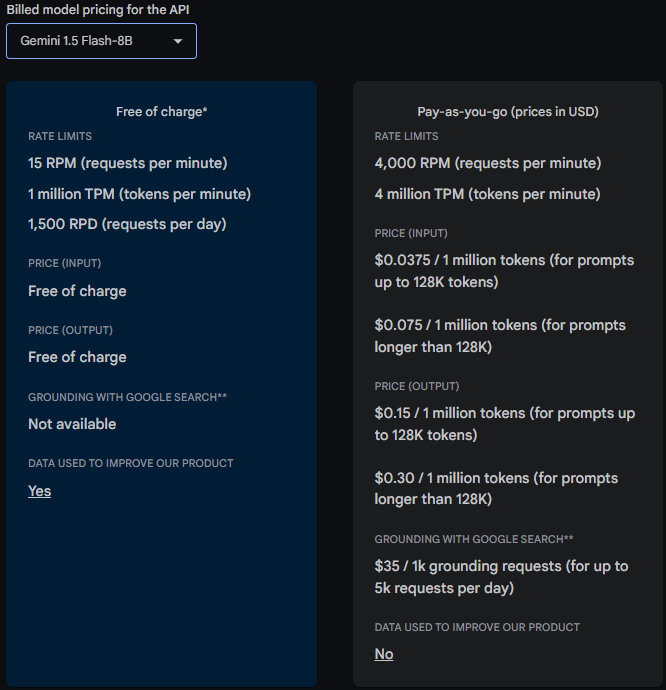
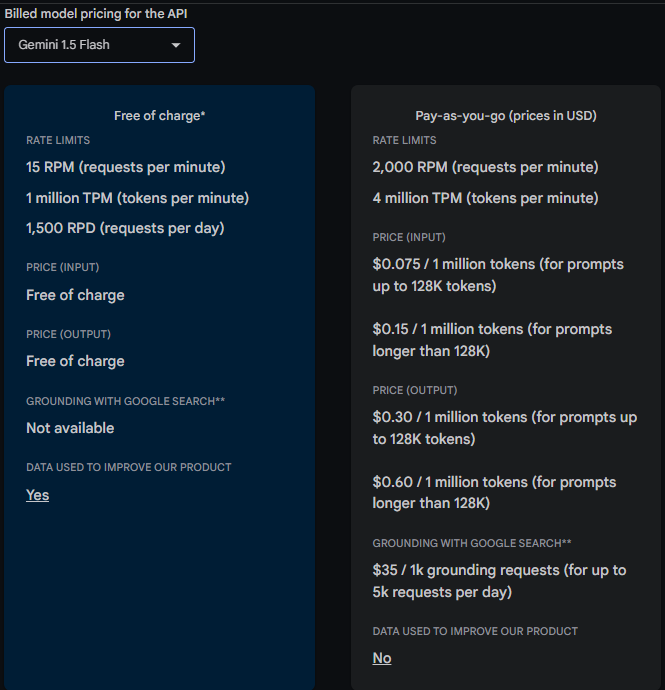
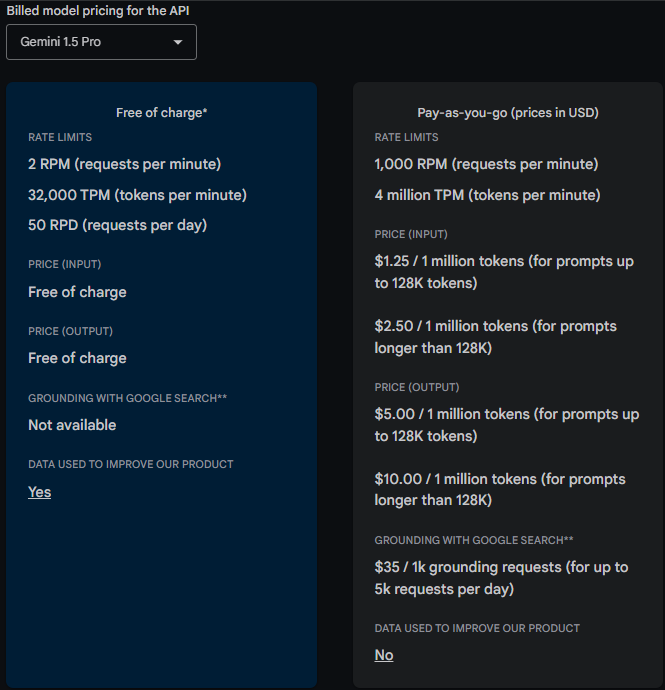


1. **Crawl dữ liệu và gửi dữ liệu** để call API được thực hiện **tách biệt và song song** nhau, khi crawl đủ dữ liệu và call API để gửi dữ liệu đi thì việc crawl vẫn được tiếp tục thực hiện, giúp **không tốn thời gian chờ đợi API** trả về kết quả
2. Thiết lập **systemInstruction** khi thiết lập kết nối tới API để đảm bảo model sẽ trả về kết quả theo đúng cấu trúc đã được cung cấp trong các ví dụ mẫu (few shot prompting)

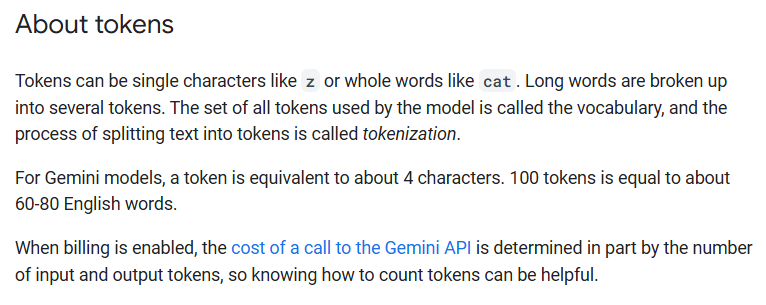


1. Sử dụng phiên bản LLM **mới nhất và tối ưu nhất về các điều kiện sử dụng** mà Google cung cấp API là **Gemini-1.5-flash** giúp các câu trả lời có độ chính xác cao nhất.

* So sánh giữa các model mới nhất của Google Gemini API:



* Giải thích các thông số:
* RPM: Số lượng requests tối đa 1 phút
* TPM: Số lượng tokens đối đa 1 phút
* RPD: Số lượng requests tối đa 1 ngày
* Tokens của văn bản đầu vào sẽ được tính bằng tokenizer của model và sẽ không bằng số lượng từ hay số lượng kí tự của văn bản đầu vào



* Dựa trên các giới hạn mà Google cung cấp đối với tài khoản miễn phí thì model Gemini-1.5 flash là hợp lý nhất vì:
* Số tokens tối đa 1 phút là 1.000.000 tokens -> do crawl toàn bộ nội dung trang web của hội nghị nên thông tin đầu vào thường khác dài và tốn tokens, trung bình sẽ là vài ngàn tokens một đường link -> chỉ có Gemini-1.5-flash hỗ trợ input có tokens lớn (tối đa 1.000.000 tokens) -> chọn Gemini-1.5-flash thay vì

Gemini-1.5-pro dù là mô hình mạnh hơn nhưng chỉ hỗ trợ input tối đa 32.000 tokens) còn Gemini-1.5-flash-8b là phiên bản nhẹ hơn hơn của Gemini-1.5-flash (ít tham số hơn – 8 tỉ tham số) nên kết quả trả về có thể không chính xác bằng nhưng có thể dùng để backup cho Gemini-1.5-flash trong nhiều trường hợp.

* Số request tối đa 1 phút là 15 -> tuy nhiên ta sẽ không lo về giới hạn này vì dữ liệu sẽ chỉ được gửi khi đủ 1 batch với một số lượng thông tin từ các đường link nhất định
* 1 batch hiện tại đang được thiết lập đủ thông tin từ 100 đường links (tokens của input sẽ dao động từ khoảng 400.000 tokens đến 800.000 tokens, tuy nhiên cần phải xem xét thêm output đầu ra vì các model chỉ hỗ trợ output tối đa là 8192 tokens, từ đó có thể điều chỉnh nếu cần thiết để gửi nhiều đường link hơn hoặc ít hơn vào 1 request) thì mới gửi request và thời gian để crawl đủ thông tin từ 100 đường link sẽ lớn hơn 1 phút và vì lúc này đã qua 1 phút nên số lượng request trên 1 phút đã được reset về 0.
* Kết quả trả về đối với 1 batch gồm 100 đường links sẽ thường từ 30s đến 60s:

- Đối với CORE2023 đang có khoảng 958 hội nghị, mỗi hội nghị 4 đường link (đối với trường hợp tất cả các đường link đều truy cập được và không bị chặn):

958 x 4 = 3832 đường link

- 1 batch sẽ gồm 100 đường link:

3832 / 100 = 39 batch

- Thời gian nhận phản hồi từ 1 batch trung bình khoảng 30s đến 60s:

39 x (30 – 60s) = 20 – 40 phút

* Do việc call API sẽ được thực hiện song song với việc crawl dữ liệu nên sẽ không phải tốn thời gian cho việc nhận kết quả trả về từ API trên.

1. **Một số hạn chế và đề xuất giải pháp khắc phục và cải tiến:**

* Crawl song song đồng thời nhiều luồng sẽ tiêu tốn nhiều tài nguyên CPU và RAM hơn 1 luồng

-> cần có cấu hình máy chủ crawl ổn

-> giải pháp: thuê VPS để làm máy chủ crawl (có thể thuê trên Azure, Google Cloud)

* Tất cả thông tin của toàn bộ các hội nghị đang được tìm kiếm bằng Google

-> thường bị Google chặn với lý do phát hiện lưu lượng truy cập bất thường

-> dẫn đến việc truy cập một số đường link có thể bị lỗi và không crawl được thông tin

-> các giải pháp có thể xem xét:

* Có thể xem xét giảm số luồng crawl để giảm thiểu khả năng bị Google chặn

-> tuy nhiên sẽ làm tăng thời gian crawl

* Có thể sử dụng đồng thời Google và Bing để chia sẻ khối lượng tìm kiếm

-> Bing có thể trả về kết quả không chính xác bằng Google

* Tắt các tài nguyên không cần thiết khi truy cập một trang web

-> có thể gây lỗi khi truy cập trang vì một số trang yêu cầu phải có tài nguyên nhất định thì mới có thể truy cập, ví dụ như trang có thể yêu cầu bật javascript, …

-> giải pháp: bổ sung cơ chế kiểm tra các tài nguyên cần thiết khi truy cập thay vì áp dụng tắt các tài nguyên đối với tất cả các đường link

* Cải thiện systemInstruction để đảm bảo model sẽ trả về đúng và đầy đủ kết quả theo định dạng mong muốn
* Việc call API đôi khi có thể xảy ra một số lỗi không mong muốn như lỗi 503 Service Unavailable do model bị overloaded và không thể nhận phản hồi

Bị giới hạn số lượng tokens của input đầu vào, output đầu ra và request trong 1 phút và 1 ngày

* Giải pháp:
  + - Có một model riêng để có thể sử dụng rất kỳ lúc nào và không bị giới hạn như call API
    - Sử dụng các model đã được pre-trained có thể tải về cục bộ để train và

fine-tuned theo dữ liệu crawl được.

* + - Model nhóm đang có dự định sử dụng:
      * Tên model: **allenai/led-base-16384**
      * Chuyên cho tác vụ Summarization
      * Hỗ trợ input đầu vào lớn ( vài nghìn tokens ) so với các model khác
      * Không quá nặng và yêu cầu tài nguyên để train ở mức trung bình
      * Link tham khảo thông tin model:

[**allenai/led-base-16384 · Hugging Face**](https://huggingface.co/allenai/led-base-16384)