**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG KHOA HỌC TỰ NHIÊN – ĐHQG TPHCM**



***Logo

Description automatically generated***

**BÁO CÁO ĐỒ ÁN**

XÂY DỰNG WEB TÌM KIẾM VÀ HIỂN THỊ CÁC NỘI DUNG   
TƯƠNG ĐỒNG VỚI 1 VĂN BẢN CẦN KIỂM TRA

BẰNG GOOGLE

**Môn: Xử lý ngôn ngữ tự nhiên**

**Giáo viên hướng dẫn: Lương An Vinh**

**Mục lục: Trang**

1. Nội dung đồ án 2
2. Thông tin thành viên, phân công và hướng dẫn 3
3. Thuật toán Textrank 4
4. Áp dụng thuật toán TextRank 5
5. Thư viện Pypeteer và sử dụng 6
6. Tài liệu tham khảo 7

# Nội dung đồ án:

Xây dựng web tìm kiếm và hiển thị các nội dung tương đồng với 1 văn bản cần kiểm tra bằng Google.

1. **Mục đích:**

Tìm kiếm các văn bản có nội dung tương đồng với một văn bản được đưa vào chương trình bằng cách trích xuất các từ khoá chính của văn bản đã cho và tìm kiếm các văn bản tương đồng trên web.

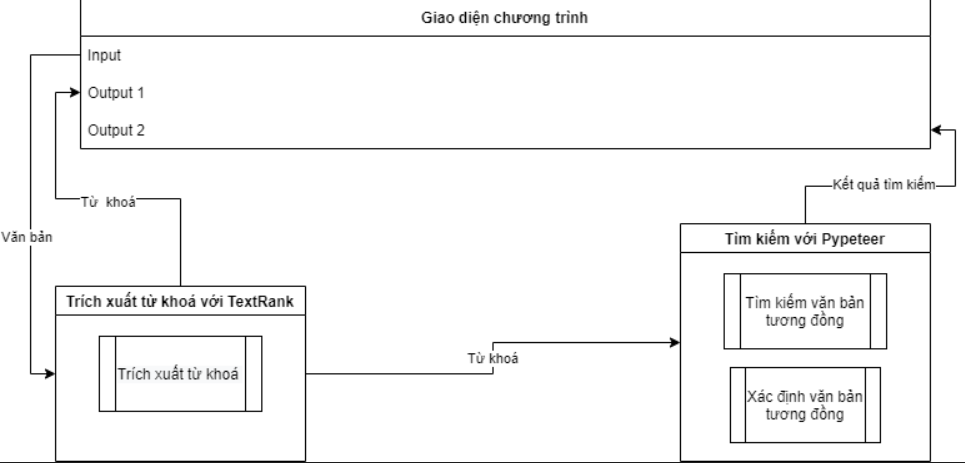
1. **Ứng dụng:**

Có thể sử dụng chương trình này để xác định đạo nhái văn bản hoặc đơn giản là tìm ra nguồn gốc của văn bản được đưa vào.

1. **Cấu trúc chương trình:**
   1. **Môi trường lập trình:**

* Lập trình bằng ngôn ngữ python.
* Các thư viện sử dụng là: tkinter, textrank, pyppeteer, …
  1. **Hướng dẫn sử dụng chương tình:**
* Bước 1: Nhập đầu vào tại Input 1 đoạn text cần kiểm tra.
* Bước 2: Bấm vào TEST nếu cần kiểm tra(trường hợp nếu muốn xóa đi nhập lại thì bấm DELETE INPUT)
* Bước 3: Nếu muốn kiểm tra 1 văn bản khác thì bấm vào DELETE ALL.
* Bước 4: Và cuối cùng thoát thì ấn nút EXIT.
  1. **Flows:**

https://drive.google.com/file/d/1Zp\_gsTC8MXU42AK8C1sTignob1XIhsnf/view?usp=sharing



# Thông tin thành viên, phân công và hướng dẫn:

1. **Thông tin thành viên:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và Tên** | **MSSV** | **Phân công** |
| 1 | Hoàng Đức Quang | 19127530 | Viết chương trình GUI python, viết report, làm video demo. |
| 2 | Phan Đình Tuân | 19127613 | Tìm hiểu và biên dịch thư viện Pyppeteer, tìm hiểu thuật toán TextRank, viết report. |
| 3 | Tô Vũ Thái Hào | 19127391 | Tìm hiểu thuật toán TextRank và chỉnh sửa code để phù hợp với chương trình, viết report. |

1. **Đánh giá mức độ hoàn thành:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tiêu chí hoàn thành** | **Mức độ hoàn thành** |
| 1 | Giao diện chương trình để nhập và xuất kết quả |  |
| 2 | Trích xuất các từ khoá từ văn bản nhập vào | 100% |
| 3 | Tìm kiếm và xác định văn bản tương đồng trên web |  |
| 4 | Xây dựng trang web | 0% |
| **Tổng kết** | |  |

* *Nhận xét:*
* *Mức độ hoàn thành khoảng 75%.*
* *Chưa xây dựng được Web, đồ án này chỉ dừng ở mức lấy thông tin “ text” trên Brower để trả về.*

# Thuật toán TextRank:

1. **Lý do chọn thuật toán này:**

Thuật toán này phù hợp được với yêu cầu đồ án được giao. Thuật toán TextRank có thể tóm tắt và rút trích các Keyword để chúng ta có thể thực hiện tìm kiếm thông tin từ Keyword đó từ Web để trả về thông tin đúng như yêu cầu.

1. **TextRank là gì?**

TextRank - là một mô hình xếp hạng dựa trên biểu đồ để xử lý văn bản có thể được sử dụng để tìm các câu có liên quan nhất trong văn bản và cũng để tìm từ khóa. Nó là một kĩ thuật tóm tắt văn bản được sử dụng trong xử lý ngôn ngữ tự nhiên để tạo các văn bản tóm tắt tài liệu.

1. **Cách thức hoạt động của TextRank?**

* Trích xuất tất cả các câu từ tài liệu văn bản, bằng cách tách ở khoảng trắng hoặc điểm dừng hoàn toàn, hoặc bất kỳ cách nào khác mà bạn muốn xác định câu của mình.
* Sử dụng thuật toán **tf-idf** để tính xác xuất số lần xuất hiện của từ đó trong văn bản hay tài liệu .

Thuật toán **TF -IDF** ?

***TF-IDF*** (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Một vài biến thể của tf-idf thường được sử dụng trong các hệ thống tìm kiếm như một công cụ chính để đánh giá và sắp xếp văn bản dựa vào truy vấn của người dùng.

* Một biểu đồ được xây dựng trong đó các đỉnh của biểu đồ thể hiện mỗi câu trong tài liệu và các cạnh giữa các câu dựa trên sự trùng lặp nội dung, cụ thể là bằng cách tính số từ mà 2 câu có trong phổ thông.
* Dựa trên mạng lưới các câu này, các câu được đưa vào thuật toán **Pagerank** để xác định các câu quan trọng nhất. Khi chúng ta muốn trích xuất một bản tóm tắt của văn bản, bây giờ chúng ta có thể chỉ lấy những câu quan trọng nhất.

Thuật toán **PageRank**?

***PageRank*** là thuật toán đời đầu của Google, sử dụng chủ yếu cho web page, khi mà chúng có thể “link” được đến nhau. Idea của PageRank là “Page nào càng được nhiều link tới, và được link tới bởi các page càng quan trọng, thì score càng cao”. Để tính toán được PageRank, thì chúng ta chỉ cần sử dụng WebCrawler để crawl được mối quan hệ “link” giữa tất cả các trang web, và tạo được một Directed Graph của chúng.

* Để tìm các từ khóa có liên quan, thuật toán textrank xây dựng một mạng từ. Mạng này được xây dựng bằng cách xem những từ nào nối tiếp nhau. Một liên kết được thiết lập giữa hai từ nếu chúng theo sau nhau, liên kết có trọng số cao hơn nếu 2 từ này xuất hiện thường xuyên hơn cạnh nhau trong văn bản.
* Trên đầu mạng kết quả, thuật toán Pagerank được áp dụng để xác định tầm quan trọng của từng từ. 1/3 trên cùng của tất cả những từ này được giữ lại và được coi là có liên quan. Sau đó, một bảng từ khóa được tạo bằng cách kết hợp các từ có liên quan lại với nhau nếu chúng xuất hiện theo sau nhau trong văn bản.
* Các câu được sắp xếp theo thứ tự giảm dần dựa trên điểm của chúng. K câu đầu tiên được chọn để làm một phần của phần tóm tắt văn bản.

# Áp dụng thuật toán TextRank:

**Hàm trích xuất từ khoá:** *extract\_key\_phrases(text)*

Input: Văn bản cần trích xuất từ khoá (text).

Output: Một set các từ khoá quan trọng của văn bản (text).

1. **Thư viện đi kèm:**

* Nltk
* Networkx
* Itertools
* Editdistance

1. **Các bước hoạt động:**
   * Dùng thư viện nltk để tách văn bản thành các tokens bao gồm các: từ, cụm từ, dấu câu,… Và lưu vào biến *word\_tokens*.
   * Gắn nhãn (tag) từ loại cho các token vào biến *tagged*.
   * Lọc tìm các từ có khả năng là từ khoá bằng hàm *filter\_for\_tags(tagged)*:
     + Danh từ, cụm danh từ: có tag *NN* và *NNP*;
     + Tính từ: có tag JJ.
   * Loại bỏ các dấu câu khỏi *tagged* bằng hàm *normalize(tagged).*
   * Sắp xếp các từ, cụm từ duy nhất (loại bỏ các lần xuất hiện trùng lặp) vào một list tên *word\_set\_list* .
   * Dựng một biểu đồ bằng hàm build\_graph(). Trong đó mỗi đỉnh thể hiện một từ khoá của văn bản, mỗi cạnh thể hiện minimum edit distance của các từ.

**Hàm *build\_graph(word\_set\_list):***

* + - Tạo một biểu đồ bằng *networkx.graph()*
    - So sánh edit distance của từng cặp từ trong *word\_set\_list* và lưu thành *weight*
    - Add các cặp từ và *weight* của chúng vào *graph.*
    - Lưu *graph.*
  + Tính toán tầm quan trọng của mỗi từ khoá trong văn bản bằng hàm *networkx.pagerank(graph)* rồi sắp xếp lại theo thứ tự giảm dần và lưu vào biến *keyphrases*.
  + Chọn ra top các từ khoá có tầm quan trọng cao nhất (1/3 số đỉnh đầu tiên trong biểu đồ).
  + Xét lại các cụm từ (ghép lại từ nhiều từ) được chọn làm keyphrases trong văn bản, nếu có 2 từ nằm cạnh nhau trong văn bản mà được chọn làm từ khoá thì ghép chúng lại với nhau.
  + Cuối cùng, trả về set các từ, cụm từ khoá.

**Hàm trả về dãy từ khoá:** *return\_key\_phrases(text, delimiter)*

Input:

* Văn bản cần trích xuất từ khoá (text).
* Dấu phân cách (delimiter) giữa các từ khoá. Mặc định là dấu phẩy: “, “

Output: Một chuỗi các từ khoá quan trọng của văn bản (text) được cách nhau bởi dấu phân cách (delimiter).

# Thư viện Pyppeteer và sử dụng:

Sử dụng thư viện Pyppeteer của Python để có thể thực hiện chụp màn hình kết quả trả về của chrome.

Để sử dụng thư viện này ta cần cài đặt nó và import vào trong python. Đây là cách cài đặt:

**Cài đặt bằng pip từ PyPI:**

python3 -m pip install pyppeteer

Hoặc cài đặt phiên bản mới nhất từ [github](https://github.com/miyakogi/pyppeteer) :

python3 -m pip install -U git+https://github.com/miyakogi/pyppeteer.git@dev

**Các bước thực hiện để lấy dữ liệu từ kết quả trả về:**

1. Tạo phiên bản trình duyệt. Ở đây ta có thể thay đổi đường dẫn để có thể lấy dữ liệu từ nguồn trang thay vì Wikipedia.
2. Chúng ta cần đoạn “text” để có thể tìm kiếm thông tin bằng đoạn “text” đó. Ta sử dụng phương thức:

*await page.type ('[id = searchInput]', text)*

1. Chụp màn hình và xuất nó ra file PNG. Ta điền tên file ảnh để khi chụp màn hình kết quả thì ảnh có ở trong file PNG.
2. Đóng trình duyệt

# Tài liệu tham khảo:

* [https://techvidvan.com/tutorials/create-web-browser-python-pyqt/](https://techvidvan.com/tutorials/create-web-browser-python-pyqt/?fbclid=IwAR0jYuwLio6i6VLb3EMws-taMgR9THLrK0QIqwFgdxlAImDe5nS-yM8aOjw)
* <https://www.wikihow.com/Make-a-Web-Browser>
* <https://github.com/summanlp/textrank>
* [Xếp hạng văn bản để tóm tắt văn bản (opengenus.org)](https://iq.opengenus.org/textrank-for-text-summarization/) – ví dụ hay
* [Textrank để tóm tắt văn bản (r-project.org)](https://cran.r-project.org/web/packages/textrank/vignettes/textrank.html) – lý thuyết hay
* <https://github.com/davidadamojr/TextRank>