TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO

MÔN TRUY VẤN THÔNG TIN ĐA PHƯƠNG TIỆN

THIẾT KẾ HỆ THỐNG TRUY VẤN VĂN BẢN

TÌM KIẾM SÁCH

Giáo viên hướng dẫn: Nguyễn Vinh Tiệp

Lớp: CS336.J21

Sinh viên thực hiện: Nguyễn Đắc Phi Hùng – 16521688

Nguyễn Duy Minh – 16521735

Trần Quang Khôi – 16521703

Lời cảm ơn

Chúng em xin cảm ơn giảng viên hướng dẫn là thầy Nguyễn Vĩnh Tiệp đã tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình học tập và thực hiện đồ án. Trong quá trình làm bài tập và đồ án, do thời gian và khả năng của bản thân còn hạn chế, nên chúng em không tránh khỏi sai sót. Vì vậy chúng em mong được sự bổ sung góp ý của thầy để hoàn thiện tốt hơn.

Chúng em xin chân thành cảm ơn thầy và chúc thầy gặp nhiều thành công trong cuộc sống!

Mục lục

I. GIỚI THIỆU 3

II. PHÂN TÍCH VÀ THIẾT KẾ 3

1) Phân tích 3

2) Cấu trúc dữ liệu 3

III. THỰC HIỆN 4

1. Cài đặt công cụ cần thiết: 4

2. Crawl dữ liệu từ trang web: 4

3. Tiền xử lý và tái cấu trúc dữ liệu 4

4. Truy vấn, xếp hạng 6

5. Flowchart 8

6. Cải tiến so với đề tài trước. 10

7. Hướng phát triển 10

8. Thiết kế giao diện người dùng 11

I. GIỚI THIỆU

- Đồ án của nhóm em là phần mềm "Tìm kiếm sách" dựa trên đồ án của anh chị Lý Bảo Khang (15520343), Mai Quốc Kiệt (15520400), Lê Thiện Duy (15520158).
- Mục tiêu đồ án:

  Trả về các đầu sách đúng với câu truy vấn của người dùng nhập (Truy vấn Tiêu đề và nội dung), Ngoài ra còn cho phép người dùng chọn lựa giữa một trong hai cách tìm các văn bản liên quan: độ tương đồng theo Distance và độ tương đồng theo Cosine. Dẫn Link đọc thử của cuốn sách mở trên trình duyệt.
- Link source đồ án:
  https://github.com/minhnduy/TVTTDPT2018_2019_16521735_16521703_16521688

II. PHÂN TÍCH VÀ THIẾT KẾ

1) Phân tích

- Bắt đầu bằng việc crawl dữ liệu từ các trang web.
- Từ dữ liệu thu về, ta xây dựng inverted index sau khi bỏ các kí tự đặc biệt và các từ stopwords
- Sau đó ta tính hệ số tf – idf cho các term để biết được độ quan trọng của term đó
- Cuối cùng từ hệ số tf – idf tính được, ta tính độ tương đồng theo Distance hoặc Cosine, sau đó xếp hạng cho các văn bản và cuối cùng trả về các văn bản có độ liên quan cao nhất.

2) Cấu trúc dữ liệu

- Nhóm sử dụng một dictionary để lưu các giá trị sau khi tính inverted index với key là các term, value là một dictionary mới. Trong dictionary mới này có key là các file chứa term đó và value là số lần xuất hiện của term trong file đó.

- Để lưu cấu trúc khi tính tf – idf, nhóm sử dụng một dictionary với key là các term, value là một dictionary mới. Trong dictionary mới này có key là các file chứa term đó và value là hệ số tf – idf của từ đó trong file đó.
- Để lưu các giá trị của độ tương đồng theo Distance hoặc Cosine, ta sử dụng một dictionary, mỗi key là tên file, value là độ tương đồng của file đó. Giá trị của Cosine là từ (0,1) và giá trị của Distance là từ (0, ∞). Sau đó ta tiến hành xếp hạng trong dictionary để trả về các file có giá trị độ tương đồng cao nhất.
- Các dictionary khi ghi vào file sẽ được lưu dưới dạng file JSON

III. THỰC HIỆN

1. **Cài đặt công cụ cần thiết:**
   - Cài đặt **Python 3.7.x**.
   - Hệ điều hành Window 10/Linux Ubuntu 16.04
   - IDE: Visual Studio Code
   - Cài đặt các thư viện sau:

     Requests, Beautiful Soup4, wxPython, Codecs, Json, Os, Itertools, Math, Time, Threading.
2. **Crawl dữ liệu từ trang web:**

   Trang web mục tiêu là Waka, URL là *https://waka.vn/*
3. **Tiền xử lý và tái cấu trúc dữ liệu**
   - ***Ý tưởng:***

     Theo cấu trúc dữ liệu đã đề ra, nhóm sẽ tạo một 2 dictionary để chứa inverted index của các term. Vậy sẽ có một không gian tọa độ n chiều với mỗi chiều là một term trong inverted index. Để xây dựng inverted index, cần phải có các bước xử lý trước:
     - Chuyển file về chung một định dạng (trong đồ án này nhóm chuyển toàn bộ các file thành định dạng UTF-8).
     - Loại bỏ các kí tự đặc biệt như / * - , …

- Tách văn bản thành các token dựa vào các khoảng trắng.
- Loại bỏ các stopwords tiếng Việt (như a lô, a ha, …)

- ***Thực hiện:***

- Bước 1: Chuyển các file tin tức về chung một định dạng:

  Nhóm tạo hàm **changeEncoding()** để chuyển các file về định dạng UTF-8 bằng cách: tìm định dạng của file tin tức hiện có bằng hàm **detect()** trong thư viện chardet, sau đó đọc và in nội dung file đó ra một file mới có định dạng UTF-8 File chuyển đổi định dạng các tin tức: **change.py**

- Bước 2: Tách văn bản thành các token dựa vào khoảng trắng:

  Sau khi chuyển đổi file thành định dạng encode UTF-8, nhóm sử dụng hàm **codecs.open()** để đọc các file đã được encode chính xác. Sau đó nhóm đọc nội dung file bằng lệnh **read()** và tách các token dựa vào khoảng trắng bằng lệnh **split()**

- Bước 3: Loại bỏ stopwords và xây dựng inverted index:

  Hàm thực hiện: **build_inverted_index (đường dẫn, các từ stopwords, jsonKey, inverted_index_fileName)**. Đầu tiên nhóm tải file stopwords tiếng Việt "*vietnamesestopwords.txt*" từ trên GitHub. Sau đó nhóm đọc file stopword bằng hàm stopword (đường dẫn đến file stopwords) tự định nghĩa và dùng hàm **split()** để trả về một list các từ stopwords. Nhóm tạo biểu thức chính quy **(regex)** để xử lý chung stopwords và các kí tự đặc biệt bằng **re.compile()**. Tiếp theo nhóm duyệt qua từng file trong dataset, đọc nội dung và chuyển về chữ thường và lưu vào string split_words. Nhóm sử dụng hai lần hàm **re.sub()**: hàm re.sub() đầu tiên trong split_words để thay thế các từ stopwords thành các ký tự khoảng trắng, hàm re.sub() thứ hai chuyển các kí tự đặc biệt bên trong split_words thành khoảng trắng. Cuối cùng nhóm tách split_words thành các token dựa vào khoảng trắng và tiến hành xây dựng inverted index.

  ❖ Xây dựng inverted index:

Nhóm tạo một dictionary có chứa key là các term, mỗi term như vậy sẽ có value là một dictionary nhỏ chứa file chứa term đó và số lần xuất hiện của term đó trong file đó.

Cách xây dựng dictionary: Nếu term đó chưa có trong dictionary, thì ta thêm term đó vào và tạo một dictionary mới làm value cho term đó. Nếu đã có term đó, ta xét đến dictionary nhỏ của term. Nếu chưa có file đang duyệt trong dictionary của term, ta thêm file đó vào thành key của dictionary nhỏ và gán số lần xuất hiện là 1, ngược lại ta tăng số lần xuất hiện lên thêm 1. Cấu trúc của dictionary inverted_index theo code**: {term: {tên file chứa term đó: số lần term đó xuất hiện trong file đó}}** Sau khi xây dựng xong inverted index, nhóm lưu vào 2 file có định dạng json **inverted_index_content.txt(cho nội dung tóm tắt của sách)** và **inverted_index_title.txt(cho tiêu đề cuốn sách).**

4. Truy vấn, xếp hạng

- Tính toán **tf_idf:**
  - Ý nghĩa của tf – idf

Trọng số tf: Một term sẽ quan trọng hơn trong một văn bản nếu term đó xuất hiện nhiều lần văn bản đó

$$tf = \frac{số\ lần\ term\ đó\ xuất\ hiện\ trong\ văn\ bản}{tổng\ số\ từ\ của\ văn\ bản}$$

hoặc

$$tf = 1 + \log\left(\frac{số\ lần\ term\ đó\ xuất\ hiện\ trong\ văn\ bản}{tổng\ số\ từ\ của\ văn\ bản}\right)$$

Trọng số idf: Một term sẽ dễ dàng phân biệt hơn nếu nó xuất hiện trong ít văn bản hơn :

$$idf = \frac{tổng\ số\ văn\ bản}{số\ văn\ bản\ chứa\ term\ đó}$$

Trọng số tf – idf: Sự kết hợp giữa tf và idf, một term sẽ có giá trị tìm kiếm cao khi xuất hiện nhiều trong văn bản đó và xuất hiện ít trong bộ dữ liệu các văn bản.

$$tf.idf(term, doc) = tf(term, doc) * idf(term)$$

Ta cũng xem query như là một document và cũng tính tf_idf, sau đó áp dụng để tính độ tương đồng ở sau

- Ý tưởng:

  Theo ý tưởng đã đưa ra ban đầu, nhóm sẽ tạo hai dictionary tương tự như inverted index. Cũng tương tự như inverted index, ta sẽ có một không gian n chiều với mỗi chiều là một term trong inverted index Sau đó ta tính tf – idf của từng document và của query nhập vào theo công thức ở trên.
- Thực hiện: Nhóm chia thành 2 phần: tính tf-idf với từng văn bản và tính tf-idf cho query

  Tính tf-idf cho từng văn bản:

  Hàm thực hiện: compute_tf_idf(đường dẫn dataset, inverted index đã xây dựng, tf_idf_fileName)

  Sau khi tính toán và lưu lại inverted index, nhóm load lại inverted index. Với mỗi từ trong inverted index, ta sẽ xét từng file trong dataset. Nếu file đó có chứa term đó, ta load file đó lên, rồi bỏ các kí tự đặc biệt bằng cách sử dụng biểu thức chính quy (regex), rồi sẽ tiến hành tính tf, idf và tf * idf cho từng term, trong đó biến x chứa giá trị của tf và biến y chứa giá trị của idf. Sau đó ta lưu kết quả tính tf-idf vào một dictionary có key là term, value là một dictionary nhỏ có key là tên file chứa term đó, value là giá trị tf-idf của term đó với file đó.

  Cấu trúc của dictionary tf-idf theo code: {term: {tên file có chứa term đó: giá trị tf-idf của term đó với file chứa nó}} Sau khi tính xong tf-idf,

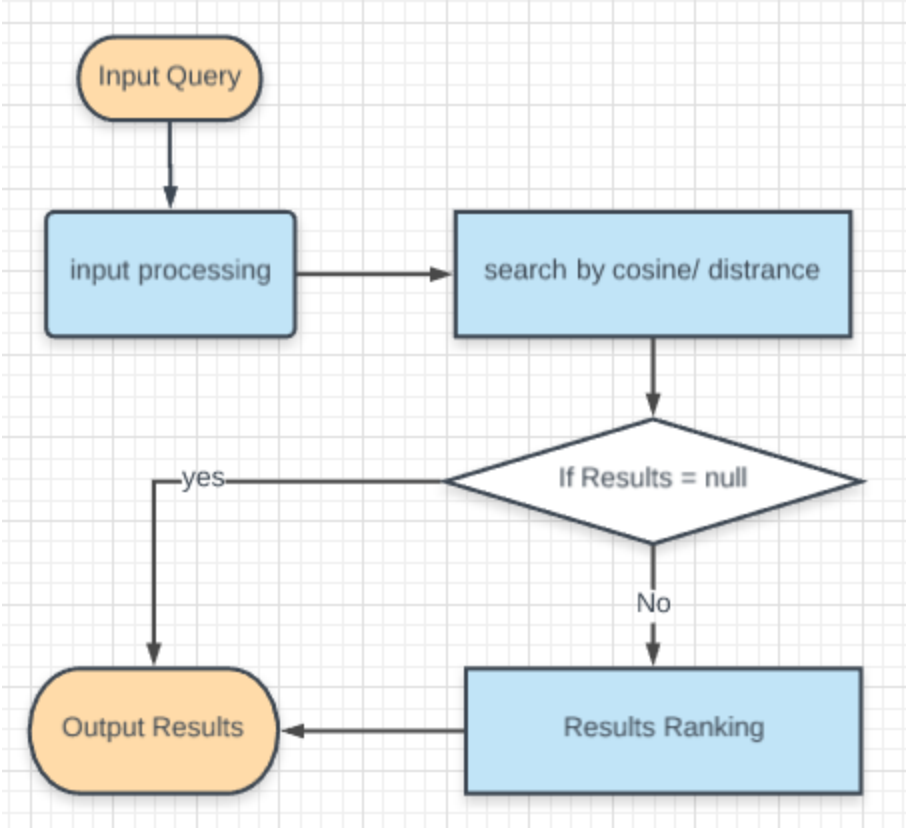
nhóm luu vào file có định dạng json tf_idf_content.txt(cho nội dung cuốn sách) và tf_idf_title.txt(cho tên sách).

Tính tf-idf cho query:

Hàm thực hiện: compute_tf_idf_query(đường dẫn dataset, dictionary tf-idf, query) Nhóm cũng tính toán tf-idf cho query tương tự như tính tf-idf cho các file tin tức, nhưng lưu trên RAM, không đánh chỉ mục. Sau đó đưa các file liên quan đến query (trong file chứa ít nhất một từ trong query) vào một list file_relevance. Với mỗi tên file liên quan đến query, tính độ tương đồng giữa tên file và query, sau đó sắp xếp lại list theo độ tương đồng giảm dần và chọn 20 file đầu.

- Tính độ tương đồng dựa trên cosine và distance:
  - Ý nghĩa: Dựa vào độ tương đồng, ta có thể xác định được các văn bản liên quan đến query. Ta có thể sắp xếp độ liên quan của các file để có thể trả về các kết quả liên quan nhất cho query.
  - Có 2 cách tính độ tương đồng: theo Distance (khoảng cách) hoặc Cosine (góc). Nếu giá trị Distance càng nhỏ thì độ tương đồng càng cao, còn giá trị Cosine càng cao thì độ tương đồng càng cao.

5. Flowchart



- Input Query

  Dữ liệu người dùng nhập vào hệ thống thông qua giao diện người dùng

- Input Processing

  Xử lý câu query ở bước Input query

  - Xóa stopwords,...
  - Tính tf_idf cho từng term của câu query

- Search by Cosine / Distrance

  -Xử lí tf-idf theo thuật toán Cosine/Distrance và xuất ra danh sách kết quả

- If Results = NULL

  -Xử lí danh sách kết quả, kiểm tra danh sách kết quả hợp lệ có rỗng hay không. Nếu danh sách kết quả trống thông báo không có kết quả nào được tìm thấy và xuất ra Output Result nếu có kết quả thì đến bước Results Ranking

- Results Ranking

-Xếp hạng danh sách kết quả và lấy ra top 20 kết quả chính xác nhất

- Output Results

-Hiển thị kết quả ra giao diện người dùng

6. Cải tiến so với đề tài trước.

- Sử dụng ajax/json để lấy dữ liệu

  Tăng tốc độ truy xuất dữ liệu, giảm thiểu internet data sử dụng, tăng tính linh hoạt trong việc xử lí dữ liệu.

- Sử dụng .json file để lưu trữ

  Tăng tốc độ truy xuất dữ liệu, tăng tính linh hoạt trong việc thao tác dữ liệu, tăng tính dễ đọc của dữ liệu.

- Sử dụng search hai bước: theo "tiêu đề" và "đầy đủ nội dung"
  - Ý tưởng: Mong muốn tăng trải nghiệm người dùng bằng việc search theo hai công đoạn là:

    Bước 1: truy vấn câu query theo tên sách

    Bước 2: truy vấn theo câu query theo nội dung tóm tắt của sách
  - Xây dựng:

    Hàm thực hiện**: searchProcess(self, phương thức tính độ tương đồng, câu query , startTime).** Lần lượt thực hiện truy vấn theo hai bước nêu trên. Sau khi truy vấn theo Tên sách xong sẽ trả về kết quả và hiển thị lên màn hình người dùng sau đó mới tiến hành truy vấn theo nội dung tóm tắt của sách và cập nhập lại danh sách trả về.

    ⇨ Tăng trải nghiệm người dùng

- Cải tiến lại thuật toán tính độ tương đồng theo Cosin và Distance

  ⇨ Tăng tốc độ truy vấn → Tăng trải nghiệm người dùng.

7. <u>Hướng phát triển</u>

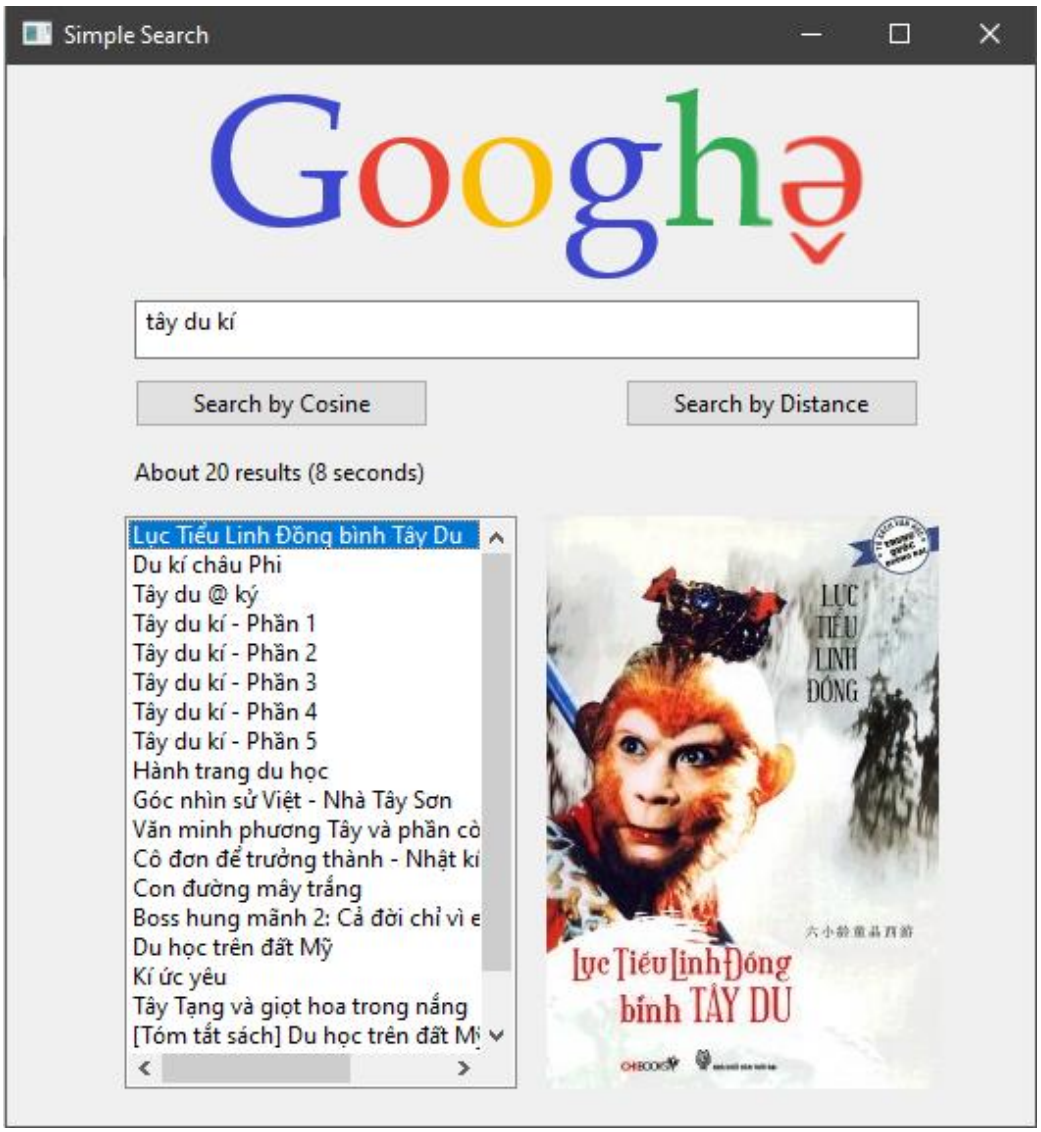
Nghiên cứu tối ưa hóa hơn cho đa luồng, tối ưa hóa việc đánh chỉ mục tăng tốc độ truy cập dữ liệu khi truy vấn.

Phát triển một số chức năng thêm.

8. Thiết kế giao diện người dùng

a. *Giao hiện chương trình*

Ví dụ tìm kiếm “tây du kí”.



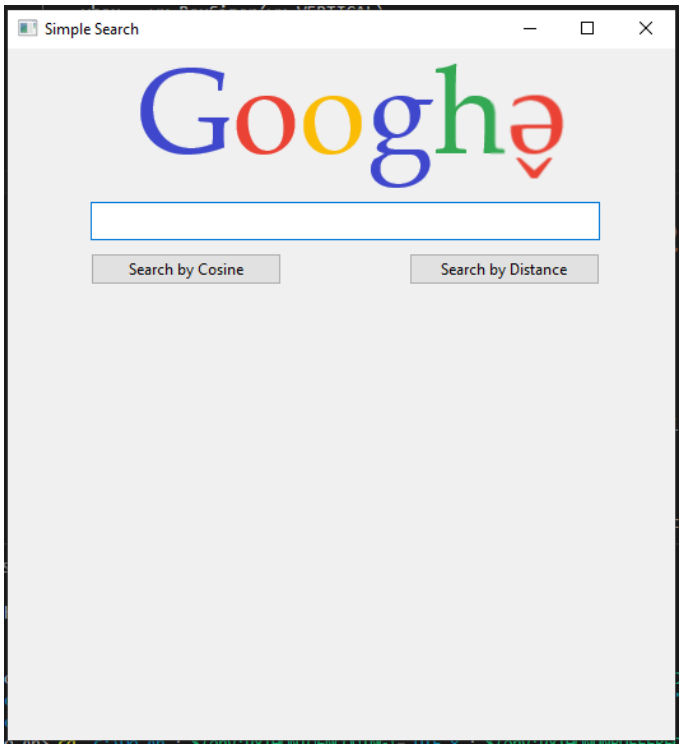
Giao diện gồm:

- Logo **GOOGHÊ**
- 1 thanh tìm kiếm để người dùng nhập vào từ khoá tìm kiếm

- Nút Search by Cosine cho phép tìm kiếm với độ tương đồng theo Cosine
- Nút Search by Distance cho phép tìm kiếm với độ tương đồng theo Distance
- Số kết quả trả về và thời gian quá trình tìm kiếm
- List tên các đầu sách bên trái
- Ảnh bìa của đầu sách được chọn bên phải

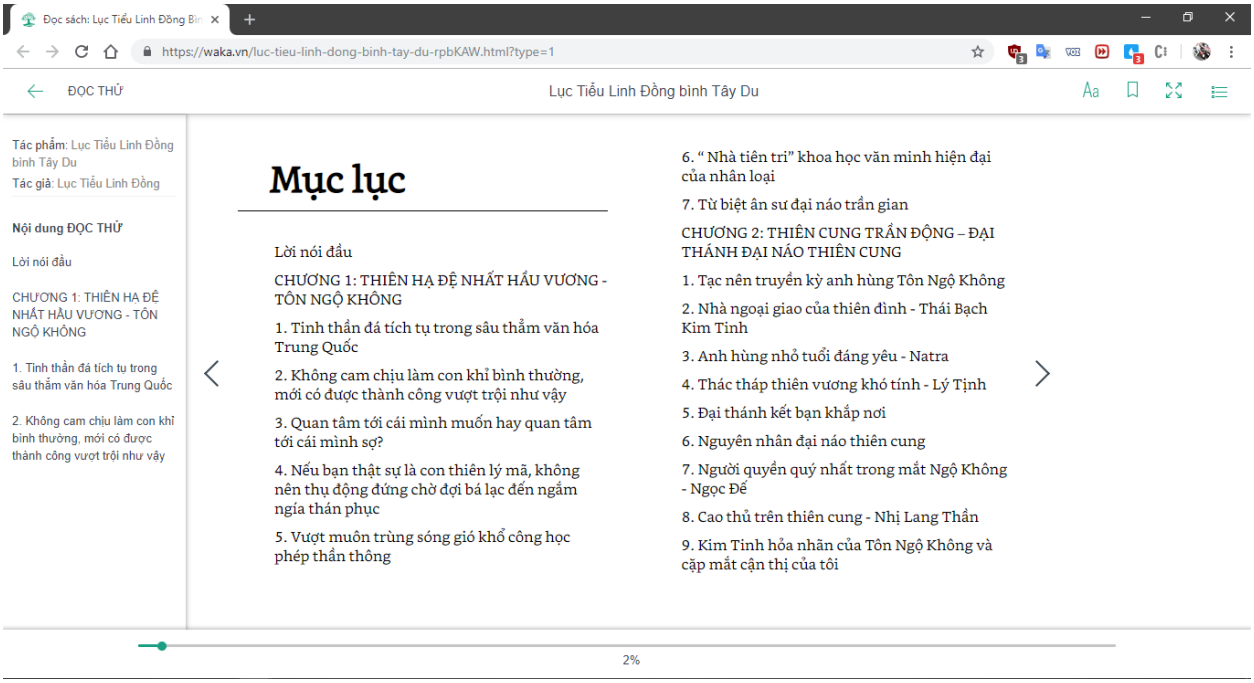
Kết quả chạy chương trình

Trước khi tìm kiếm:

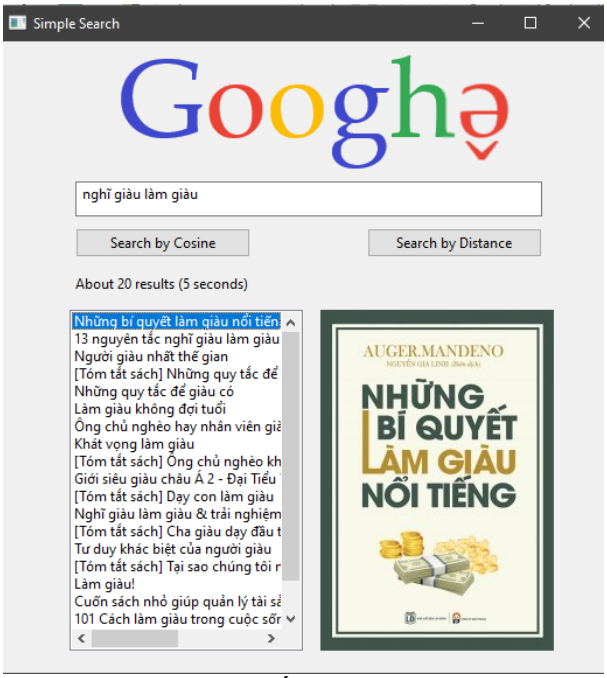


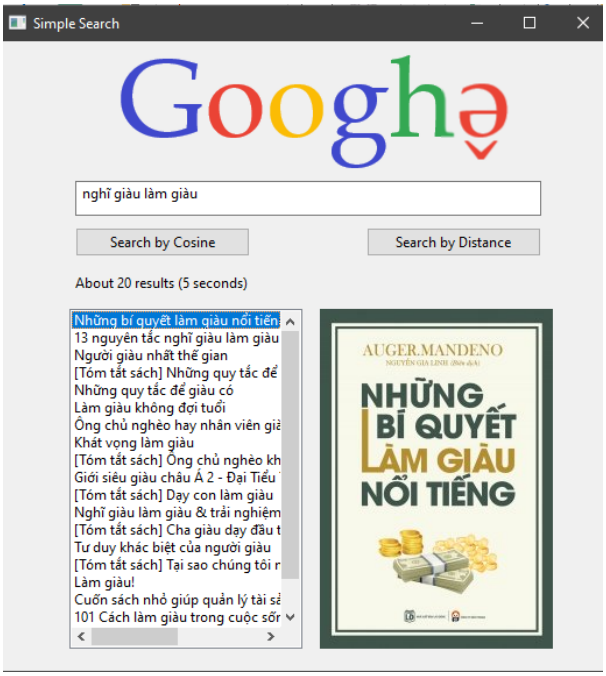
Sau khi tìm kiếm:

⇨ Bấm vào ảnh bìa cuốn sách sẽ mở trang đọc thử:



So sánh giữa 2 kiểu tìm kiếm Cosine với Distance:





⇨ Thời gian tìm kiếm tương đương nhau là 5s

⇨ Kết quả tìm kiếm có chút khác biệt