ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Thực hành Kĩ thuật lập trình

ĐỒ ÁN MÔN HỌC

Search Engine

Sinh viên thực hiện

Hồ Văn Sơn
Trần Hữu Thiên

Tháng 7 năm 2021

Mục Lục

1. Mô tả Đồ án Search Engine

1.1. Mục tiêu

Rút trích nội dung chính của văn bản tiếng Việt. Tìm kiếm những văn bản có nội dung tương ứng với từ khóa do người dùng nhập vào, xếp hạng kết quả tìm kiếm theo mức độ liên quan từ khóa từ cao đến thấp.

1.2. Yêu cầu

Các sinh viên trong lớp cùng tập hợp dữ liệu thống nhất bao gồm các văn bản tiếng Việt. Các tập tin văn bản được đặt trong cùng một thư mục với tập tin tương ứng với tựa đề và phần mở rộng .txt. Dựa trên các văn bản nguồn, sinh viên tự tạo ra tập tin siêu dữ liệu (metadata) ở dạng nhị phân hoặc văn bản; thông tin về các văn bản đã tiền xử lý và nội dung chính của từng văn bản theo cấu trúc tự định nghĩa. Khi thêm/xóa tập tin văn bản, chương trình tự động cập nhật dữ liệu của tập tin siêu dữ liệu.

1.3. Yêu cầu lập trình

- Hiển thị menu cho phép người dùng chọn chức năng.
- Tận dụng các cấu trúc dữ liệu đơn giản đã học.
- Phải truy vấn dựa trên tập tin siêu văn bản, không truy vấn trực tiếp trên các văn bản.
- Tự cài đặt thuật toán sắp xếp, tìm kiếm (không sử dụng thư viện có sẵn) với mảng hoặc danh sách liên kết. Tổng quát hóa bằng cách sử dụng con trỏ hàm (tùy chọn).
- Chỉ sử dụng thư viện `string.h` của C để xử lý chuỗi.

2. Định hướng cách xử lý và tổ chức dữ liệu

Sau thời gian nghiên cứu về đồ án, nhóm đề xuất cách giải quyết cho bài toán này gồm các bước như sau:

1. Rút trích đặc trưng nội dung các tập tin văn bản

   Bước này sẽ thực hiện các thao tác xử lý, chuẩn hóa dữ liệu trong các file văn bản và tính toán đưa ra các từ quan trọng (keyword) là đặc trưng của mỗi văn bản.

2. Xây dựng tập tin siêu dữ liệu (metadata)

   Đây là tập tin được tạo nên từ các đặc trưng của các văn bản trong hệ thống dữ liệu.

3. Một số chức năng thao tác trên metadata như: thêm file vào metadata, xóa file khỏi metadata, thêm các file trong 1 folder vào metadata, chỉnh sửa 1 file trong metadata.

Các bước này sẽ được làm rõ hơn trong các mục bên dưới.

3. Rút trích đặc trưng nội dung các tập tin văn bản

Đầu tiên, ta đọc 1 file văn bản bằng cách sử dụng bộ ký tự mở rộng (thư viện `wchar.h`). Tiếp theo ta tiến hành các bước xử lý nội dung cụ thể như sau:

3.1. Tiền xử lý văn bản

3.1.1. Chuẩn bị

Để thuận tiện cho việc xử lý cũng như tăng tốc thời gian tìm kiếm các ký tự, ta tạo mảng ánh xạ ký tự in hoa sang kí tự in thường (`MAP_UPPER`) để dùng trong bước *chuẩn hóa văn bản*, và mảng ánh xạ ký tự in thường có dấu sang ký tự in thường không dấu (`MAP_LOWER`) để dùng trong bước *xóa dấu câu*.

Vì bộ nhớ của mỗi ký tự unicode là 2 bytes, nên khoảng giá trị sẽ từ 0 đến $2^{16} - 1$. Do đó độ dài mảng `MAP_UPPER` và `MAP_LOWER` bằng 2^{16}, là khá nhỏ nên việc khởi tạo và lưu trữ trên RAM là khả thi.

3.1.2. Chuẩn hóa văn bản

Bước này nhằm đưa dữ liệu văn bản về dạng chuẩn, tức là chỉ giữ lại các thông tin là các ký tự chữ cái, các chữ cái in hoa được đưa về dưới dạng chữ thường, giữa mỗi 2 từ cách nhau bởi duy nhất 1 khoảng trắng và loại bỏ tất cả khoảng trắng ở 2 đầu.

Ví dụ: (trích trong file `.\train\Du hoc\DH_TN_ (601).txt`)

Văn bản trước khi chuẩn hóa:
`Trong các ngày 22 và 26/4 tại Hà Nội và TP.HCM, cty VINECO sẽ tổ chức các buổi hội thảo du học tìm hiểu thông tin về trường đại học Fontys (Hà Lan).`

Văn bản sau khi chuẩn hóa:
`trong các ngày 22 và 26 4 tại hà nội và tp hcm cty vineco sẽ tổ chức các buổi hội thảo du học tìm hiểu thông tin về trường đại học fontys hà lan`

3.1.3. Loại bỏ stopword

Trước hết, ta sẽ đọc dữ liệu là các stopword từ file `stopword.txt` được tạo từ trước với bộ ký tự mở rộng, sau đó đưa toàn bộ các stopwords vào 1 mảng `wchar_t*`. Sau đó, dựa vào mảng này ta loại bỏ các stopwords có trong văn bản đã được chuẩn hóa ở bước trên. Từ đó, ta tạo được dữ liệu có tính đặc trưng cao hơn cho văn bản.

Tuy nhiên nếu cứ với mỗi từ hoặc cụm từ ta duyệt tất cả stopword có trong tập để kiểm tra thì thời gian loại bỏ stopword của cả tập dữ liệu sẽ rất lâu. Để tối ưu thời gian thực hiện thao tác này ta sắp xếp các stopword tăng dần theo thứ tự từ điển sau đó với mỗi từ hoặc cụm từ cần kiểm tra ta tìm kiếm nhị phân trên tập stopword đã được sắp xếp.

Ví dụ:

Văn bản trước khi bỏ stopword:
`trong các ngày 22 và 26 4 tại hà nội và tp hcm cty vineco sẽ tổ chức các buổi hội thảo du học tìm hiểu thông tin về trường đại học fontys hà lan`

Văn bản sau khi bỏ stopword:
`22 26 4 hà nội tp hcm cty vineco tổ chức hội thảo du học thông trường đại học fontys hà lan`

3.1.4. Xóa dấu câu

Mục đích của việc xóa dấu câu là nhằm đặc trưng hóa dữ liệu văn bản, giúp cho việc xử lí lượng lớn văn bản cũng như việc tìm kiếm bằng chuỗi từ không dấu trở nên dễ dàng hơn.

Ví dụ:

Văn bản trước khi bỏ dấu:
`22 26 4 hà nội tp hcm cty vineco tổ chức hội thảo du học thông trường đại học fontys hà lan`

Văn bản sau khi bỏ dấu:
`22 26 4 ha noi tp hcm cty vineco to chuc hoi thao du hoc thong truong dai hoc fontys ha lan`

3.2. Tìm keyword của văn bản

Nhóm tiến hành theo các bước như sau:

1. Tách văn bản thành các chuỗi từ có độ dài là 1, 2 và 3.

   Điều này nhằm mục đích kiểm soát các từ hay đi cạnh nhau, ví dụ như: `ha noi, du hoc, truong dai hoc,...`

   Với mỗi giá trị `len` $\in \{1, 2, 3\}$, ta sẽ có được một dãy các chuỗi `char** words[len]` tương ứng, với tổng số lượng phần tử là `total_words[len]`. Các bước dưới đây sẽ được *thực hiện 3 lần*, tương ứng với mỗi giá trị của `len`.

2. Sắp xếp dãy các chuỗi từ tăng dần theo thứ tự từ điển:

   Để hiệu quả, nhóm dùng *Sắp xếp trộn (Merge sort)* để thực hiện sắp xếp. Lợi ích của việc sắp xếp dãy các chuỗi từ giảm dần là: dễ dàng thống kê số lần xuất hiện của mỗi chuỗi từ, dễ dàng loại bỏ các chuỗi từ trùng nhau, tìm kiếm các chuỗi từ được nhanh hơn bằng tìm kiếm nhị phân.

3. Loại bỏ các từ trùng nhau, đếm số lần xuất hiện của từ đó:

   Sau khi sắp xếp các chuỗi từ theo thứ tự tăng dần thì các từ, cụm từ giống nhau sẽ là các đoạn con liên tiếp trong dãy. Từ đây ta dễ dàng loại bỏ ra các từ, cụm từ trùng nhau bằng cách duyệt từng từ, cụm từ chưa xóa và tiến hành xóa các từ, cụm từ kế tiếp trong dãy cho đến khi khác từ, cụm từ hiện tại.

   Trong quá trình loại bỏ các chuỗi từ trùng nhau, ta đồng thời cũng đếm được số lần xuất hiện của mỗi chuỗi từ trong dãy các chuỗi từ đó. Từ đó, ta có được mảng `frequency[len]` lưu lại tần suất xuất hiện của mỗi chuỗi từ trong dãy ban đầu (trước khi bỏ các từ trùng nhau) và cập nhật lại mảng `words[len]` chỉ chứa các chuỗi từ khác nhau.

4. Tính toán và đưa ra các keyword cho văn bản:

   Đối với mỗi chuỗi từ, nó sẽ có một giá trị gọi là trọng số thể hiện độ quan trọng của chuỗi từ đó trong văn bản. Nhóm đề xuất một cách tính trọng số đó là:

$$\boxed{\texttt{scale[len][i]} = \frac{\texttt{frequency[len][i]}}{\texttt{total\_words[len]}}}$$

Trong đó:

- `scale[len][i]` là trọng số của chuỗi từ `words[len][i]` trong văn bản.
- `frequency[len][i]` là số lần xuất hiện của `words[len][i]` trong văn bản.
- `total_words[len]` là tổng số các chuỗi từ có trong `word[len]`.

Sau khi tham khảo và tính toán dựa trên một số văn bản có sẵn, nhóm lọc ra các keyword nếu từ, cụm từ đó thỏa mãn điều kiện:

$$\boxed{\frac{3\%}{\texttt{len}} \leq \texttt{scale[len][i]} \leq 1}.$$

Điều này nói lên rằng:

- Với một từ được xem là keyword nếu trung bình mỗi 34 từ thì từ đó lại xuất hiện, nói cách khác nếu ta giả sử mỗi câu trung bình chứa từ 15 - 20 từ thì cứ mỗi 2 câu thì lại có 1 câu chứa từ đấy.
- Đối với cụm từ (số từ bằng 2 hoặc bằng 3) thì tần suất xuất hiện của cụm từ sẽ càng giảm đi nếu số từ càng tăng lên do đó nhóm giảm tần suất xuất hiện thấp nhất của một keyword là cụm từ xuống tương ứng là 1.5% đối với cụm từ có 2 từ và 1% đối với cụm từ có 3 từ.

Kết quả ta sẽ có được một lượng keyword có trong văn bản là `keyword_count[len]`.

4. Xây dựng tập tin siêu dữ liệu (metadata)

1. Đầu tiên, ta truyền vào đường dẫn tập `train` (là thư mục chứa các file văn bản làm dữ liệu) vào chương trình. Nếu đường dẫn không tồn tại thì thông báo ra màn hình và kết thúc tại bước này.

2. Tiếp theo, ta tiến hành tổng hợp tất cả tên của toàn bộ file văn bản làm dữ liệu (là các đường dẫn tuyệt đối tới file) vào file `index.inp`, tên của mỗi file nằm trên 1 dòng.

3. Ta tạo 1 file nhị phân `position.txt` để lưu vị trí của dữ liệu của các văn bản gồm tên file trong file `index.inp` và đặc trưng của nó trong file metadata.

4. Bước tiếp theo, duyệt từng văn bản có trong `index.inp`. Với mỗi file văn bản, ta thực hiện rút trích đặc trưng nội dung chính cho văn bản đó, đồng thời cập nhật file `position.txt` vị trí của con trỏ trong file `index.inp` và file metadata. Sau khi tính toán được các keyword, mỗi file văn bản ở trong metadata sẽ có dạng như sau:

   - Dòng đầu là `keyword_count[len]`.
   - Trong `keyword_count[len]` dòng tiếp theo, mỗi dòng sẽ có cú pháp như sau: ‘‘`frequency[len][i] words[len][i]`’’, với $0 \leq$ `i` $<$ `keyword_count[len]`

   Các giá trị `keyword_count[len], words[len], frequency[len]` được định nghĩa như ở phần *trên*. Với lưu ý rằng thao tác trên cũng được *lặp lại 3 lần* đối với độ dài chuỗi từ của `words[len]` là `len` $\in \{1, 2, 3\}$.

5. Cuối cùng, sau khi duyệt hết tất cả các file trong `index.inp`, ta có được một giá trị `nFiles` là tổng số file có tập dữ liệu. Và ta ghi giá trị `nFiles` này vào cuối file `position.txt`. Vậy file `position.txt` ta tạo ra gồm `(nFiles` $\times$ `2 + 1)` blocks số nguyên, mỗi block có độ lớn là 4 bytes.

5. Các chức năng chính có trong menu của chương trình

5.1. Chức năng tìm kiếm một chuỗi từ khóa

Khi nhập vào, người dùng sẽ được hướng dẫn nhập vào 1 chuỗi từ khóa, có thể có dấu hoặc không dấu. Sau đó chương trình sẽ có nhiệm vụ như sau:

1. Tiền xử lý chuỗi ký tự nhập vào. Sau đó thực hiện thao tác tách chuỗi vừa nhập thành 3 dãy gồm các chuỗi có độ dài `len` $\in \{1, 2, 3\}$. Các dãy vừa tách đó được biểu diễn bởi `char** word[len]`, và mỗi dãy `word[len]` chứa các chuỗi có độ dài `len` sẽ có độ dài là `count[len]`.

2. Duyệt từng file văn bản trong metadata, thực hiện thao tác tìm kiếm nhị phân chuỗi các keyword ứng với độ dài chuỗi `len` tương ứng, đưa ra đánh giá độ liên quan của các văn bản đối với chuỗi ký tự nhập vào.

   Một cách tổng quát: Gọi `score` và `occur` là 2 mảng lần lượt biểu diễn cho *điểm đánh giá* dựa trên các từ khóa trong chuỗi ký tự mà mỗi văn bản chứa và *số lần xuất hiện các từ khóa* trong văn bản đó. Văn bản thứ `i` được gọi là có độ liên quan lớn hơn văn bản thứ `j` nếu như biểu thức sau đây xảy ra:

   ```
   score[i] > score[j] || (score[i] == score[j] && occur[i] > occur[j])
   ```

   Trong đó, gọi `num[len][indx]` là số lần xuất hiện của chuỗi từ `word[len][indx]`. Khi đó:

   $$\texttt{score[i]} = \sum_{\texttt{len=1}}^{3} \left(\sum_{\texttt{indx=0}}^{\texttt{count[len]-1}} \texttt{f[len][indx]} \times (2 \times \texttt{len - 1}) \right)$$

   $$\texttt{occur[i]} = \sum_{\texttt{len=1}}^{3} \left(\sum_{\texttt{indx=0}}^{\texttt{count[len]-1}} \texttt{num[len][indx]} \right)$$

   Trong đó `f[len][indx] = 1` nếu `num[len][indx] > 0` và ngược lại `f[len][indx] = 0`.

   Trong công thức tính `score` trên, lý giải cho việc sử dụng giá trị `(2×len - 1)` là bởi vì: độ quan trọng của các chuỗi keyword dài hơn sẽ dược quy ước là lớn hơn. Vì vậy nhóm tính điểm cho keyword có độ dài 1, 2, 3 lần lượt là 1, 3, 5. Như vậy sẽ phần nào đánh giá tốt hơn độ quan trọng của keyword đó trong văn bản.

3. Sắp xếp các văn bản tìm được theo thứ tự độ liên quan giảm dần (ưu tiên giảm dần theo điểm đánh giá của văn bản, sau đó là số lần xuất hiện của các từ khóa trong văn bản). Nếu như không tìm thấy văn bản nào thì thông báo ra màn hình và kết thúc tại bước này.

4. In ra màn hình các thông tin sau: số file văn bản tìm được, thời gian thực hiện tìm kiếm, top 5 văn bản có keyword liên quan và các thông số thể hiện mức độ liên quan của văn bản đó đối với chuỗi từ cần tìm kiếm. Sau đó, cho người dùng tùy chọn tiếp tục với các lựa chọn sau:

   - In ra thêm 5 văn bản liên quan nữa.
   - In 1 văn bản trong số các văn bản tìm được ra cửa sổ console.

5.2. Chức năng thêm 1 file vào metadata

Chức năng này được thực hiện dựa trên các bước như sau:

1. Đầu tiên người dùng nhập vào đường dẫn tuyệt đối của file cần thêm vào. Kiểm tra sự tồn tại file trong máy, nếu file không tồn tại thì thông báo ra màn hình và kết thúc chương trình tại bước này.
2. Kiểm tra xem file đã tồn tại trong metadata hay chưa. Nếu đã tồn tại rồi thì thông báo ra màn hình và kết thúc tại bước này.
3. Rút trích đặc trưng nội dung của file đó rồi thêm vào cuối metadata.
4. Cập nhật file `index.inp` và `position.txt`, kết thúc chương trình.

5.3. Chức năng thêm tất cả các file text từ 1 folder vào metadata

Thực hiện lần lượt theo các bước sau:

1. Đầu tiên, người dùng nhập vào đường dẫn tuyệt đối tới thư mục chứa các file văn bản cần thêm. Nếu thư mục không tồn tại thì thông báo ra màn hình và kết thúc chương trình tại đây.
2. Tổng hợp tất cả các file có đuôi là `.txt` vào trong 1 file tạm, ở đây sẽ là `temp_index.inp`.
3. Duyệt các file trong `temp_index.inp`, lần lượt thực hiện rút trích đặc trưng nội dung file đó và thêm chúng vào cuối file metadata.
4. Xóa file tạm `temp_index.inp`, cập nhật file `index.inp` và file `position.txt`, kết thúc chương trình.

5.4. Chức năng xóa 1 file khỏi metadata

Chức năng này được thực hiện dưa trên các bước như sau:

1. Đầu tiên người dùng nhập vào đường dẫn tuyệt đối của file cần xóa đi
2. Kiểm tra xem file đó có tồn tại trong metadata hay không. Nếu không thì kết thúc chương trình tại bước này.
3. Xóa file bằng cách xây dựng lại file metadata theo phương pháp đó là: chia metadata thành 2 phần, lần lượt là các file nằm trước và các file nằm sau file cần xóa. Sau đó ghép 2 phần đó lại với nhau mà không có file cần xóa. Khi đó, file đã được xóa thành công khỏi metadata.
4. Cập nhật file `index.inp` và `position.txt`, kết thúc chương trình.

5.5. Chức năng sửa 1 file trong metadata

Các bước được thực hiện như sau:

1. Đầu tiên người dùng nhập đường dẫn tuyệt đối của file muốn chỉnh sửa.
2. Kiểm tra xem file đó có tồn tại trong metadata hay không. Nếu không, thông báo ra màn hình và kết thúc tại bước này.
3. Thực hiện thao tác xóa file đó khỏi metadata.
4. Mở file đó bằng notepad, cho phép người dùng tự do chỉnh sửa trên file đó. Kết thúc chỉnh sửa bằng cách tắt cửa số notepad.
5. Rút trích đặc trưng nội dung file đó và thêm nó vào cuối metadata.
6. Cập nhật `index.inp` và `position.txt`, kết thúc chương trình.

6. Tài liệu tham khảo

- StackOverFlow
- GeeksForGeeks
- Stopwords