Trường đại học Khoa học Tự nhiên

Khoa Công nghệ thông tin

----- ๛ 🕮 ๛ -----

ĐỒ ÁN CUỐI KỲ

Đề tài: SEARCH ENGINE

Giảng viên môn học: Võ Hoài Việt

Nguyễn Hoàng Lâm – 20120316

Trần Kiều Minh Lâm – 20120018

2020-2021

Đề bài: Rút trích đơn giản 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 đến từ khóa từ cao đến thấp.

Phần I: Phân tích đề:

Search engine là công cụ tìm kiếm trong văn bản. Đồ án yêu cầu tạo chương trình người dùng nhập vào từ khóa và tìm kiếm trên tập dữ liệu có sẵn (train data). Ngoài ra còn có thể thêm và xóa dữ liệu. Nếu ta truy vấn dữ liệu trực tiếp trong tập dữ liệu gốc thì thời gian sẽ rất lâu. Do đó ta cần:

- Đọc dữ liệu trong thư mục train và làm sạch dữ liệu (xóa dấu, viết thường, xóa các dấu câu thừa, xóa stopword).
- Rút trích văn bản và lưu vào file metadata để có thể truy xuất nhanh hơn.
- Thực hiện rút trích văn bản theo 3 dạng monogram, digram, trigram
  Ví dụ: đại học quốc gia: “đại”, “học”, “quốc”, “gia”, “đại học”, “học quốc”, “quốc gia”, “đại học quốc”, “học quốc gia”.
- Trả về độ xuất hiện của từ khóa quan trọng theo trọng số.

Phần II: Các kiểu dữ liệu tự định nghĩa

- File: kiểu dữ liệu tự định nghĩa lưu các giá trị của 1 file.
  + name: tên file, path: đường dẫn file. isDel: đánh dấu file đã bị xóa chưa, dùng khi truy vấn xóa file từ người dùng.
  + appear[3]: khi có truy vấn Search, dùng để lưu lại số lần xuất hiện trong file của các từ khóa người dùng nhập vào
  + grams[3]: lưu lại các từ monogram (grams[0]), digram (grams[1]), trigram (grams[2]).
  + rates[3]: lưu lại phần trăm số lần xuất hiện các từ grams tương ứng.

```
struct File {
     string name;
     string path;
     bool isDel = false;
     int appear[3] = { 0 };
     VectorStr grams[3];
     VectorInt rate[3];
     void del();
};
```

- Topic: kiểu dữ liệu tự định nghĩa lưu các giá trị của một topic (gồm nhiều file)
    + numFiles: số lượng file trong topic, cntDel: số lượng file đã đánh dấu xóa (dùng cho việc thêm và xóa file).
    + files: là 1 mảng gồm nhiều File
    + del: giải phóng vùng nhớ.

```
struct Topic {
     int numFiles = 0;
     int cntDel = 0;
     string path;
     string name;
     File* files;
     void del();
};
```

- Data: kiểu dữ liệu tự định nghĩa lưu các giá trị của một tập train (gồm nhiều topic).
    + numTopic: số lượng topic có trong data.
    + path: đường dẫn tới data.

```
struct Data {
     int numTopic;
     Topic* topics = nullptr;
     string path;
     void del();
};
```

- VectorInt: lưu mảng động có kiểu là Int, và các thông tin liên quan.
    + pushBack: thêm phần tử.
    + del: giải phóng vùng nhớ.
    + init: khởi tạo.
    + quickSort: sắp xếp nhanh.

```
struct VectorInt{
    int cap = 0;
    int size = 0;
    int* a = nullptr;
    void pushBack(int x);
    void del();
    void init(int sz, int val = 0)
```

```
    void quickSort(int left, int right);
    VectorInt& operator =(const VectorInt& cur);
```

- VectorStr: lưu mảng động có kiểu là String, và các thông tin liên quan.
    + pushBack: thêm phần tử.
    + del: giải phóng vùng nhớ.
    + init: khởi tạo.
    + quickSort: sắp xếp nhanh.
    + lowerBound: trả chỉ số của phần tử lớn hơn hoặc bằng một chuỗi nhập vào, dùng thuật toán tìm kiếm nhị phân.
    + unique: xóa các phần tử giống nhau.

```
struct VectorStr {
    int cap = 0;
    int size = 0;
    string *s = nullptr;
    void del();
    void init(int sz, string val = "");
    void pushBack(string x);
    void quickSort(int left, int right);
    int lowerBound(string x);
    void unique();
    VectorStr& operator =(const VectorStr& cur);
};
```

- PairSI: dùng để lưu một cặp giá trị <string, int>
    + cmpPSI: so sánh 2 cặp giá trị <string, int> theo biến int

```
struct PairSI {
    string s;
    int a;
    int cmpPSI(PairSI y);
};
```

- VectorPSI: lưu mảng nhiều PairSI (định nghĩa ở trên) và các hàm liên quan đến nó.
    + del: giải phóng bộ nhớ.
    + init: khởi tạo.
    + pushBack: thêm phần tử.
    + quickSort: sắp xếp.

```
struct VectorPSI {
    int cap = 0;
    int size = 0;
    PairSI* p = nullptr;
    void del();
    void init(int sz, PairSI val = { "", 0 });
    void pushBack(PairSI x);
    void quickSort(int left, int right);
};
```

Phần III: Tiền xử lý file văn bản

1. Đọc file văn bản tiếng việt

Sử dụng các thư viện: <locale>, <codecvt>, <fstream>

Dùng hàm đọc hai định dạng file khác nhau (UTF16 và UTF8). Lưu dạng văn bản sau khi đọc ở dạng wstring.

```
std::wstring ReadFileUTF16(string path) {
     std::wstring line, res;
     std::wifstream fin(path);
     fin.imbue(std::locale(fin.getloc(), new
std::codecvt_utf16<wchar_t, 0x10ffff, std::consume_header>));
     while (getline(fin, line))
          res += line + L"\n";
     fin.close();
     return res;
}
std::wstring ReadFileUTF8(string path) \\ Tương tự
```

Nguồn tham khảo:
ReadFileUTF16: https://stackoverflow.com/questions/50696864/reading-utf-16-file-in-c
ReadFileUTF8:
https://github.com/tntxtnt/VietnameseIO/blob/master/VietnameseIO/umain.cpp

- Kiểm tra file là UTF8 hay UTF16 để chọn cách đọc phù hợp:

File UTF16 sẽ luôn có 3 byte đầu tiên chứa BOM, còn UTF8 thì không chắc. Nên ta sẽ đọc vào BOM, nếu là UTF16 thì ReadFileUTF16 ngược lại ReadFileUTF8.
Tham khảo:
https://vicidi.wordpress.com/2015/03/09/reading-utf-file-with-bom-to-utf-8-encoded-stdstring-in-c11-on-windows/

2. Xóa dấu, lowercase, fix word.

- Xóa dấu: chuyển từ wstring về string: Khi gặp các nguyên âm có dấu thì sẽ chuyển thành ko dấu

Ví dụ: àáạảãâầấ -> a.
Thực hiên, tạo một mảng tĩnh các từ cần xóa dấu, dùng vòng lặp lần lượt thay thế từng từ với từ tương ứng không dấu.

```
wstring idVowels = L"aAeEiIoOuUyYdD";
wstring vowels[14] = {
L"àáạảãâầấậẩẫăằắặẳẵ", L"ÀÁẠẢÃÂẦẤẬẨẪĂẰẮẶẲẴ",
L"èéẹẻẽêềếệểễ", L"ÈÉẸẺẼÊỀẾỆỂỄ",
L"ìíịỉĩ", L"ÌÍỊỈĨ",
L"òóọỏõôồốộổỗơờớợởỡ", L"ÒÓỌỎÕÔỒỐỘỔỖƠỜỚỢỞỠ",
L"ùúụủũưừứựửữ", L"ÙÚỤỦŨƯỪỨỰỬỮ",
L"ỳýỵỷỹ", L"ỲÝỴỶỸ",
L"đ", L"Đ" };
```

- Lowercase: chuyển các từ viết hoa thành viết thường

Kết hợp vòng lặp for và hàm **`tolower`**`()` cho từng ký tự.

- Fix word: xóa dấu ko cần thiết *“.,/;:-_)(~!@#$%^&*.\n”*

Dùng vòng lặp và thay thế các dấu không cần thiết bằng một khoảng trắng.

3. Xóa stopword trong file văn bản:

- Đọc file stopword vào wstring và tiền xử lý như trên. Tách từng dòng stopword ra thành từng token (một token tương ứng với một cụm từ stopword) và lưu vào struct VectorStr do nhóm tự định nghĩa . Sắp xếp các token stopword tăng dần theo thứ tự từ điển.
- Tiến hành xóa stopword: duyệt qua các các từ trong văn bản, sử dụng chặt nhị phân để tìm những vị trí xuất hiện của từ trong VectorStr

stopword. So khớp từ trong văn bản với từ trong stopword, nếu giống nhau thì xóa đi.

- Chặt nhị phân: do mảng stopword đã sắp xếp theo thứ tự từ điển, sử dụng thuật toán này để tìm vị trí có thể giống nhau.
- Xem thêm hàm `deleteStopWord(string& content)`trong file stringFunction.cpp.

Phần IV: Rút trích văn bản

Sau khi tiền xử lý văn bản xong, bây giờ văn bản sẽ chỉ còn lại những từ quan trọng mang ý nghĩa. Do đó ta sẽ rút trích văn bản thành file metadata để dùng trong việc truy vấn.

- Hàm `TrainData(Data& metadata, string trainPath, int x, int y);` dùng vòng lặp để rút trích văn bản từng file một trong thư mục train data.
- Dùng hàm `extractKeyWord(string path, VectorStr grams[3], VectorInt rate[3]);` để trích xuất lấy keyword từng từ in vào file metadata. Cách hoạt động:
  + Tiền xử lý văn bản.
  + Tách văn bản đã qua xử lý thành từng token, sử dụng hàm `countAppearance(VectorStr& words, VectorStr& grams, VectorInt& rate, int type)` đếm phần trăn số lần xuất hiện của một `gram (mono, di, tri)` sau đó lưu vào `rate.`
  + Cách tính phần trăm như sau: Tính số lần xuất hiện của từ/cụm từ đó chia cho tổng số từ/cụm từ trong văn bản.
  + Nếu văn bản dưới 50 từ/cụm từ thì sẽ lấy toàn bộ kết quả vào metadata. Nếu trên 50 từ/cụm từ thì sẽ lấy từ 1% - 10% lưu vào metadata.
  + Chọn mốc là 50 từ và 1%-10%: trải qua nhiều lần thử nghiệm, nhóm đã thống nhất chọn mốc này vì cho ra kết quả hợp lý.
  + In ra kết quả theo cặp vào file metadata.
- Cấu trúc file metadata:
  + Dòng 1: đường dẫn tới thư mục để train ra metadata.
  + Dòng 2: số lượng topic có trong metadata.
  + Tiếp theo là nội dùng các topic, có định dang là:
    - Dòng 1: tên topic
    - Dòng 2: số lượng file có trong topic đó.

        - Tiếp theo là nội dung các file trong topic đó, có định dạng là:
            - Dòng 1: tên file
            - Số lượng từ trong grams[i], với i = 0, 1, 2
            - Các từ của grams[i] trên từng dòng, với i = 0, 1, 2
- Thời gian train ước lượng tầm 12 phút (với tập train data).
- Hàm `ReadData(Data& metadata, string dataPath, int x, int y);`
    + Đọc file metadata vào chương trình để chạy.
- Hàm `PrintFile(File& file, ofstream& fout);`
    + In các thay đổi ra file metadata khi kết thúc chương trình.

Phần V:Tìm kiếm, thêm và xóa file trên struct Data

1. Search
    - Đọc vào các từ khóa người dùng nhập → **string** input;
    - Tiền xử lý xâu input về định dạng cơ bản.
    - Duyệt qua các grams của input, sau đó tìm trong metadata. Duyệt qua từng File, dùng chặt nhị phân trên các mảng grams[3], nếu khớp thì thêm rate của grams đó vào appear[3] tương ứng.
    - Duyệt qua từng File, tính toán điểm của từng file bằng sum(appear[i] * weight[i]) và thêm vào VectorPSI res.
    - Với phép nhân trên được hiểu là tổng của các tích appear[i]*weight[i] tương ứng. Với weight[0] = 0,75 (monogram), weight[1] = 1,25 (digram), weight[2] = 1,5 (trigram)
    - Sắp xếp mảng res theo số điểm và lấy 25 kết quả cao nhất.
    - In ra màn hình 5 trang, mỗi trang có 5 kết quả là tên file và số điểm tương ứng.
2. Add file
    - Người dùng nhập đường dẫn file cần thêm vào.
    - Kiểm tra đường dẫn có đúng hay không.
    - Kiểm tra file đó đã có trong metadata hay chưa. Sử dụng hàm `FindFile`().
    - Sau cùng, `TrainFile`() và thêm vào topic "ảo" tên là topic nằm cuối "Other" trong struct metadata.
3. Delete file
    - Người dùng nhập tên file cần xóa.

- Kiểm tra file đó đã có trong metadata hay chưa. Sử dụng hàm **FindFile**() trả về vị trí của file đó. Đánh dấu **isDel** của file đó là true.

Phần VI: Giao diện người dùng

Giao diện của chương trình giúp người dùng dễ dàng thao tác sử dụng các chức năng. Bao gồm các chức năng:

- *1: Search → Nhập vào các từ khóa cần tìm kiếm. Sau đó có các câu lệnh
  + -1 → quay trở về menu
  + 0 → chuyển về trang trước.
  + 1 - 5 → Sẽ có tối đa 5 trang, mỗi trang gồm 5 kết quả. VD: nhập vào 2 sẽ xem chi tiết file kết quả thứ 2 trong trang đó.
  + 6 → chuyển đến trang sau
- *2: Add file → Nhập vào đường dẫn đến file cần thêm
- *3: Delete file → Nhập vào tên file cần xóa
- *4: Train Folder → Nhập vào đường dẫn đến thư mục cần train (lưu ý: thư mục vừa nhập đúng định dạng là thư mục lưu các thư mục topic)

  VD: source\Test\new test → đúng định dạng

- *5: Exit → Thoát chương trình, đồng thời lưu lại dữ liệu.

Lưu ý: khi tắt chương trình phải chờ lưu dữ liệu rồi mới có thể tắt, nếu tắt ngang sẽ bị mất dữ liệu.

- Các hàm cần dùng trong xử lý giao diện người dùng:

```
void goToXy(SHORT posX, SHORT posY);
```

→ Di chuyển con trỏ đến vị trí (x; y)

```
void SetColor(int backgound_color, int text_color);
```

→ Thay đổi màu chữ

```
void SetWindowSize(SHORT width, SHORT height);
```

→ Thay đổi Console Window Size

```
void SetScreenBufferSize(SHORT width, SHORT height);
```

→ Thay đổi Screen Buffer Size

```
void DisableResizeWindow();
```

→ Vô hiệu hóa thay đổi kích thước màn hình

```
void DisableCtrButton(bool Close, bool Min, bool Max);
```

→ Vô hiệu hóa các nút Minimize, Maximize và Close

```
BOOL DirectoryExists(const char* dirName);
```

→ Kiểm tra đường dẫn có tồn tại hay không

```
BOOL WINAPI SetConsoleTitle(_In_ LPCTSTR lpConsoleTitle);
```

→ Thay đổi Console Title

```
void DisableSelection();
```

→ Vô hiệu hóa Select (bôi đen text)

Tham khảo: https://codelearn.io/

Phần VII: Các thư viện và nguồn tham khảo

1. Các thư viện sử dụng trong chương trình.

- Dùng trong việc đọc file UTF8, UTF16

<fcntl.h>

<io.h>

<locale>

<codecvt>

- Thư viện cơ bản

<iostream>

<string>

<filesystem>

<fstream>

<Windows.h>

<conio.h>

2. Các nguồn tham khảo

- https://stackoverflow.com/
- https://codelearn.io/
- https://www.geeksforgeeks.org/
- https://github.com/
- https://www.cplusplus.com/reference/