Đại học quốc gia TP Hồ Chí Minh

Trường Đại học Khoa Học Tự Nhiên

Khoa Công nghệ thông tin

Báo cáo thực hành

Đồ án cuối kì

SEARCH ENGINE TIẾNG VIỆT

Nhóm 14

1. Lê Quang Trí – 20120022

2. Nguyễn Kông Đại – 20120448

Giảng viên: Võ Hoài Việt

Môn học: Kỹ thuật lập trình

Tp.HCM, ngày 15 tháng 07 năm 2021

Các thuật ngữ được dùng trong báo cáo

- Corpus: nguồn chứa các tệp văn bản.
- Document/Doc: một văn bản.
- Query: câu truy vấn của người dùng.
- Token: một từ được phân tách bằng các khoảng trống hay còn gọi là tiếng.
- Word: tương tự với token.
- Term: một từ có nghĩa trong tiếng Việt. Term có thể gồm nhiều token hoặc word.
- Stop word: từ không có nghĩa trong tiếng Việt. Trái ngược với term.
- Indexing: giai đoạn tạo các siêu văn bản.
- Minimal Interval: đoạn tối thiểu chứa một lượng token nhất định (sẽ được định nghĩa cụ thể ở trong báo cáo)
- Searching: giai đoạn tìm kiếm các văn bản liên quan đến câu truy vấn của người dùng.
- Lexicon: một cấu trúc hoặc tệp chứa các từ vựng được trích xuất sau quá trình indexing.

I. Giới thiệu đồ án

- Đồ án cuối kì
- Nội dung đồ án:

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.

- Yêu cầu:

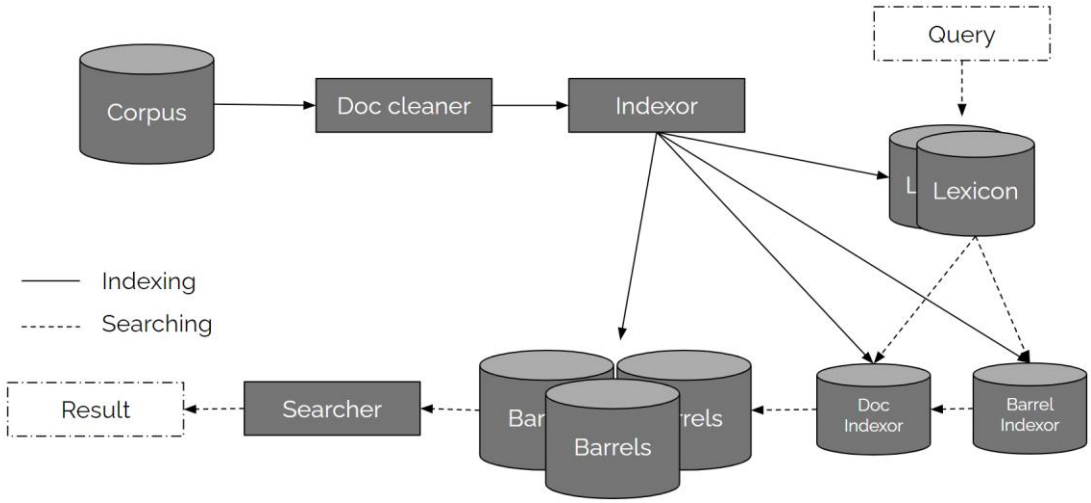
Các sinh viên trong lớp cùng tập hợp dữ liệu thống nhất bao gồm các văn bản tiếng Việt. Các tập tin văn bản được đặt trong cùng một thư mục với tên tập tin tương ứng với tựa đề và phần mở rộng .txt. Dựa trên các văn bản nguồn, sinh viên tự tạo ra tập tin siêu dữ liệu (metadata) ở dạng nhị phân hoặc văn 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.

- Yêu cầu lập trình:
  + Hiển thị menu cho phép người dùng chọn các 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.
- Hình thức và quá trình thực hiện:
  + Được viết bằng ngôn ngữ C/C++
  + Thực hiện trong thời gian nghỉ dịch học online
  + Có sự đoàn kết trong công việc
  + Làm việc có tổ chức và khoa học
- Chương trình được chia làm 3 phần chính:
  + Indexing
  + Searching
  + Chỉnh sửa dữ liệu

II. Các phần của chương trình

1. Cấu trúc chung của chương trình

Cấu trúc của nhóm chúng em lựa chọn được tham khảo theo [2] bởi sự linh hoạt của cấu trúc này. Sự linh hoạt sẽ được giải thích thông qua các phần sau.



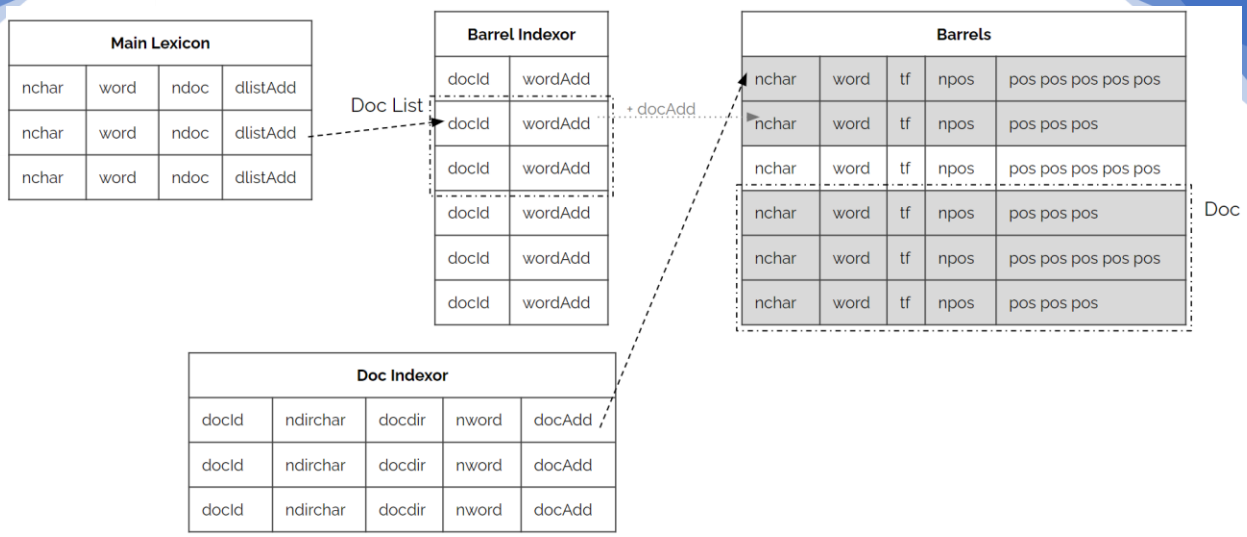
Hình 1: Cấu trúc chung của chương trình

- Quá trình indexing được thực hiện theo trình tự các mũi nét liền. Theo trình tự từ corpus, các văn bản sẽ lần lượt đi qua bộ phận làm sạch văn bản Doc Cleaner (bỏ các ký tự dấu câu, khoảng trắng thừa, dấu xuống dòng,…). Sau đó được thực hiện index bằng bộ phận Indexor và lưu trữ vào các Lexicon, Barrel, Barrel Indexor và Doc Indexor.
- Quá trình searching được thực hiện theo trình tự các mũi tên nét đứt. Từ query, chương trình sẽ tìm các từ vựng trong Lexicon và lấy địa chỉ đến Barrel Indexor và Doc Indexor. Sau đó, lấy các thông tin của các document tìm được trong Barrels để thực hiện tính điểm cho các document ở bộ phận Searcher. Cuối cùng là sắp xếp và đưa ra kết quả.

Để việc thêm/xóa document trong corpus, chúng ta phải có cách lưu trữ linh hoạt. Do đó, chương trình đã tuân thủ theo nguyên tắc thiết kế như sau: “Các document phải có docId duy nhất và được lưu trữ bằng các con trỏ”.

Khi lưu trữ bằng các con trỏ (địa chỉ) chúng ta có thể linh động thay đổi một dữ liệu sang vùng nhớ khác mà không cần tác động đến các vùng nhớ lưu trữ dữ liệu khác. Trong khi đó, để xóa “thật” vùng nhớ cũ, chúng ta có thể thực hiện việc ghi lại tệp dựa trên dữ liệu mà con trỏ trỏ tới.

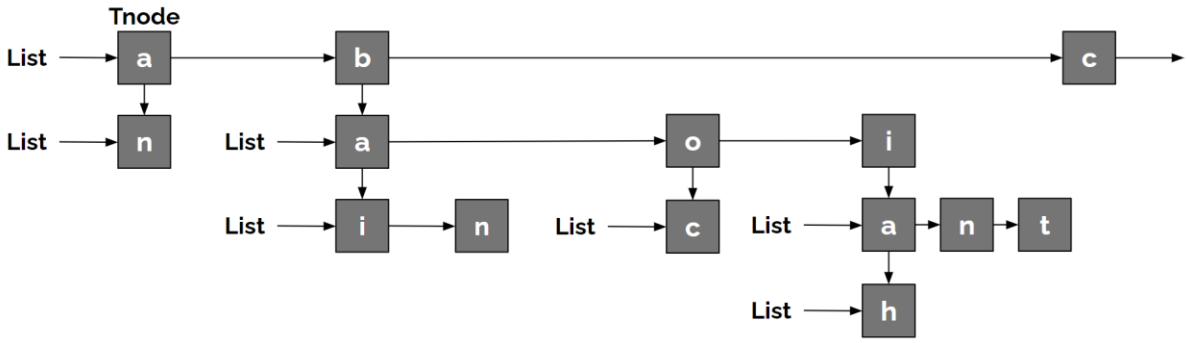
Hình 2 là mô tả cấu trúc cụ thể cho Lexicon, Barrel Indexor, Doc Indexor và Barrels. Trong đó, Lexicon sẽ chứa địa chỉ trỏ tới vùng nhớ trong Barrel Indexor; Doc Indexor sẽ chứa địa chỉ trỏ tới vùng nhớ trong Barrels. Khi thay đổi corpus, chúng ta chỉ cần thêm vùng nhớ và gán thêm địa chỉ con một con trỏ mới; hoặc xóa “giả” vùng nhớ cũ bằng cách xóa con trỏ trỏ tới vùng nhớ đó. Việc ghi lại tệp dữ liệu sẽ giúp chúng ta xóa “thật” vùng nhớ đó. Các phần sau sẽ tiếp tục làm rõ các ý tưởng này.



Hình 2: Cấu trúc chi tiết

2. Lexicon

a. Cấu trúc lưu trữ từ vựng Lexicon



Hình 3: Ví dụ về cấu trúc Lexicon

Để việc tìm kiếm một chuỗi được dễ dàng và nhanh chóng, nhóm chúng em đã thiết kế một cấu trúc dùng Linked List (gọi tắt là List). Phần data của node trong List là một struct Tnode có các loại dữ liệu sau:

- wchar_t chr: Lưu ký tự.
- bool isWord: Đánh dấu kết thúc của một từ/chuỗi.
- SList branch: Nhánh (Ký tự tiếp theo).
- void* adapter: Phần dữ liệu đính kèm theo cho một từ.

Tổ hợp các Linked List với data của các node là Tnode được gọi là **Lexicon**. Tnode có branch là một List khác được gọi là Tnode cha. Các ký tự trong List của branch là các ký tự liền sau ký tự trong Tnode cha. Các ký tự trong cùng một List (có thể hiểu là một tầng) là các ký tự riêng biệt nhau, không tạo thành một từ.

Ví dụ: Hình 3 là ví dụ về một Lexicon. Các từ mà Lexicon này có thể đang lưu trữ là “a”, “an”, “bai”, “ban”, “boc”.

Việc lưu trữ và tìm kiếm từ trong Lexicon diễn ra dễ dàng và nhanh hơn. Dưới đây là bảng so sánh lợi thế của Lexicon so với mảng và List.

	Lexicon	**Mảng**	**Linked List**
Bộ nhớ	Nhiều	Ít	Nhiều
Thêm từ vẫn giữ thứ tự	Nhanh	Chậm	Nhanh
Xóa từ vẫn giữ thứ tự	Nhanh	Chậm	Nhanh
Tìm kiếm	Nhanh	Nhanh (tìm kiếm nhị phân)	Chậm hơn

Ưu điểm của Lexicon là chỉ cần lưu ký tự thay vì cả một từ và việc tìm kiếm từ diễn ra với tốc độ gần như tuyến tính với độ dài của từ cần tìm. Tuy nhiên, cấu trúc Lexicon sẽ phức tạp hơn và tốn thêm bộ nhớ để lưu các con trỏ đến các Linked List khác.

b. Siêu văn bản Lexicon

Siêu văn bản Lexicon sẽ chứa các từ vựng đựng trích lọc từ các văn bản trong corpus. Tại đây, mỗi từ sẽ chứa địa chỉ dlistAdd trỏ tới vị trí của DocList của từ vựng đó trong Barrel Indexor.

Thông tin lưu trữ trong tệp Lexicon gồm:

- short nchar: Số ký tự của từ vựng.
- wchar_t* word: Mảng ký tự của từ vựng (bao gồm '\0').
- int ndoc: Số văn bản chứa từ vựng đó.
- int dlistAdd: Địa chỉ tới vị trí DocList.

Thông tin lưu trữ trong tệp Lexicon sau khi đưa lên RAM sẽ lưu trong cấu trúc Lexicon với adapter là các dữ liệu struct WLData gồm các dữ liệu tương tự lưu trữ trong tệp Lexicon.

Để việc lưu trữ dữ liệu Lexicon trên RAM được nhỏ gọn, nhóm em sau này có thể tách tệp Lexicon thành hai tệp riêng biệt dựa trên số lần xuất hiện của từ vựng. Khi không tìm thấy từ vựng trên Lexicon chính thì mới tìm ở tệp Lexicon phụ.

3. Barrel Indexor

Chứa các DocList lưu trữ thông tin các document gồm:

- int docId: ID của document.
- int wordAdd: Địa chỉ đến vị trí chứa thông tin của từ đó trong document.

Trong đó, wordAdd cần cộng thêm docAdd để có được địa chỉ đầy đủ đến vị trí chứa thông tin của từ đó trong Barrels. Do tính chất của việc phân tách này chúng ta có thể linh hoạt thay đổi địa chỉ vùng nhớ của document mà không cần thay đổi địa chỉ vùng nhớ của từ trong Barrel Indexor.

Ngoài ra, việc tách rời phần Barrel Indexor và Barrels sẽ giúp chúng ta có thêm một bước để chọn lọc các document có một số lượng token khớp với query trước khi tiến hành truy xuất

vào tệp Barrels khổng lồ. Từ đó sẽ hạn chế tối thiểu dung lượng bộ nhớ lưu trữ trên RAM trong quá trình searching. Tuy nhiên, hiện tại dự án chúng em chưa áp dụng bộ lọc này vì chưa thực sự cần thiết với lượng dữ liệu nhỏ như yêu cầu đồ án.

4. Doc Indexor

Chứa thông tin của các document đã được index, bao gồm:

- int docId: ID của document.
- int ndirchar: Độ dài đường dẫn đến document.
- char* docdir: Mảng ký tự đường dẫn đến document (bao gồm '\0').
- int nword: Số từ vựng trong document (sau khi lọc bỏ stop words)
- int docAdd: Địa chỉ tới vị trí chứa lưu trữ thông tin của document đó.

DocIndexor và Barrel Indexor đóng vai trò trung gian giữa Lexicon và Barrels. Tuy nhiên, hai bộ phận này lại không trực tiếp kết nối với nhau. Điều đó tạo ra sự linh hoạt cho việc thực thi lập trình, không bị ràng buộc thành một chuỗi kết nối liên tục mà có thể phát triển song song nhau và dễ dàng mở rộng/thay đổi cấu trúc. Trong các quy trình sau này, chúng ta sẽ thấy rõ sự hoạt động song hành giữa cặp Lexicon – Barrels Indexor và cặp Barrels – Doc Indexor.

5. Barrels

Chứa thông tin các từ trong các doc bao gồm:

- short nchar: Số ký tự của một từ.
- wchar_t* word: Mảng ký tự của từ (bao gồm '\0')
- float tf: Tần số xuất hiện của từ trong document.
- short npos: Số vị trí của từ trong document.
- short* posArray: Mảng các vị trí của từ trong document.

Barrels sẽ chứa hầu hết các thông tin dành cho quá trình searching, tính toán điểm số cho các document. Tuy việc lưu trữ từ trong barrels trông dư thừa vì Lexicon đã lưu trữ từ vựng rất đầy đủ, nhưng điều đó rất cần thiết cho quá trình tìm kiếm ngược lại từ Barrels vào Lexicon khi muốn xóa một tệp ra khỏi corpus.

Ví dụ:

- Document 1 có ID là 0 và có nội dung là:
  "Trịnh Công Sơn là một người có nhiều công trình tiêu biểu".
- Document 2 có ID là 1 và có nội dung là:
  "Công an trinh sát rất tài ba"
  (Các từ gạch chân là stop words sẽ được loại bỏ).
  Main Lexicon.

2	an	1	0
2	ba	1	8
4	bieu	1	8

4	cong	2	16
3	sat	1	32
3	son	1	40
3	tai	1	48
4	tieu	1	56
5	trinh	2	64

Barrel Indexor.

1	an	0
1	ba	14
0	bieu	0
0	cong	18
1	cong	28
1	sat	46
0	son	38
1	tai	62
0	tieu	54
0	trinh	72
1	trinh	78

Doc Indexor.

0	15	corpus/doc1.txt	0
1	15	corpus/doc2.txt	94

Barrel.

4	bieu	0.1429	1	11
4	cong	0.2857	2	1 8
3	son	0.1429	1	2
4	tieu	0.1429	1	10
5	trinh	0.2857	2	0 9
2	an	0.1667	1	1
2	ba	0.1667	1	6
4	cong	0.1667	1	0
3	sat	0.1667	1	3
3	tai	0.1667	1	5
5	trinh	0.1667	1	2

6. Quá trình Indexing

Quá trình này sẽ trích xuất dữ liệu từ các văn bản thông qua các bước làm sạch văn bản, xóa bỏ stop word, các vị trí xuất hiện của các từ. Các dữ liệu trích xuất được sẽ được tổ chức lưu trữ trong các siêu văn bản. Các siêu văn bản sẽ là nguồn để quá trình tìm kiếm dựa vào mà không phải trực tiếp truy vấn đến các văn bản gốc.

a) Làm sạch tệp văn bản

Đọc văn bản và xóa bỏ các ký tự dư thừa như: các dấu câu, các khoảng trắng dư thừa, ký tự xuống dòng,…

b) Tạo Word Array cho một văn bản

Tạo một List tokenData chứa các từ của document được phân tách bằng khoảng trắng và vị trí của từ đó trong document.

Việc kiểm tra stop word sẽ được dựa vào một danh sách các stop words đã được nghiên cứu sẵn, tham khảo [1]. Nhóm em chỉ xét stop word cho những từ ghép 2 tiếng vì trong danh sách stop words và trong từ điển tiếng Việt, từ ghép 2 tiếng chiếm gần 70%, theo nghiên cứu [3].

Đối với các stop word có một tiếng thì rất khó xác định có thực sự là từ không có nghĩa trong văn bản vì trong nhiều văn bản, stop word một tiếng sẽ đóng vai trò là từ quan trọng ví dụ như "người dùng", "game hay",… Do đó, phải lọc stop word một tiếng bằng tỷ lệ xuất hiện của cặp từ đó và cặp từ tương tự trong văn bản. Dưới đây là thuật toán thực hiện việc lọc bỏ stop word dựa trên ý tưởng chính này.

Thuật toán.

Bước 1: Đi qua từng từ trong List
Bước 2: Nếu cặp từ đó và từ liền sau là một stop word 2 tiếng thì đánh dấu xóa cả hai từ. Di chuyển tới từ tiếp theo và quay lại bước 1.
Bước 3: Nếu cặp từ đó và từ liền sau có lặp lại trong văn bản thì tính các tỷ lệ:
    Bước 3.1: Nếu token2 là stop word thì:
    Fm = [số lần xuất hiện "token1 token2"] / [số lần xuất hiện "token1 ***"]
    Bước 3.2: Nếu token1 là stop word thì:
    Sm = [số lần xuất hiện "token1 token2"] / [số lần xuất hiện "*** token2"]
    Với *** là một token.
Bước 4: Nếu Fm < [Threshold one stop word] và Sm = 0 thì quay lại bước 1. Nếu không thì di chuyển tới từ tiếp theo và quay lại bước 1.
Bước 5: Nếu Sm < [Threshold one stop word] và Fm = 0 thì từ đó là stop word. Đánh dấu xóa và quay lại bước 1. Nếu không thì di chuyển tới từ tiếp theo và quay lại bước 1.
Bước 6: Nếu Fm < [Threshold two stop word] và Sm < [Threshold two stop word] thì đánh dấu xóa cho hai từ. Di chuyển tới từ tiếp theo và quay lại bước 1.

Sau khi loại bỏ stop word, chúng ta tiến hành đếm số lần xuất hiện của một từ và chia cho tổng số từ để có được tần số xuất hiện cho một từ tf. Sau đó đưa các tf và các vị trí xuất hiện của từ vào mảng Word Array.

Việc xử lý stop word là xử lý các token chưa bỏ dấu. Nhưng khi lưu trữ vào các siêu văn bản ở các bước sau được thực hiện khi token đã được bỏ dấu.

c) Ghi và lưu văn bản vào Barrels và cập nhật Doc Indexor

Tiến hành ghi từng phần tử trong Word Array vào tệp Barrels và ghi lại vị trí bắt đầu của văn bản và các từ và trả về Word Address Array để lưu vào Lexicon trên RAM và cập nhật Doc Indexor.

d) Ghi dữ liệu vào Lexicon và Barrels Indexor

- Phương thức 1: Lưu trữ vào cấu trúc Lexicon trên RAM – Lưu vào Barrels Indexor – Cập nhật cấu trúc Lexicon trên RAM và lưu vào tệp Lexicon.

  Lưu trữ tất cả mảng Word Address Array trả về từ bước trên vào adapter của Lexicon. Sau đó, ghi dữ liệu của từng từ vào Barrels Indexor thành các DocList và thay đổi adapter của Lexicon thành struct WLData.

- Ưu điểm: Tốc độ index rất nhanh vì dữ liệu của tất cả từ được lưu trên RAM nên việc truy xuất, cập nhật dữ liệu rất nhanh. Thời gian thực hiện index 14000 văn bản là 1200 giây.
- Nhược điểm: Tốn nhiều bộ nhớ RAM để lưu trữ tạm thời các DocList của các từ.

o Phương thức 2: Thực hiện song song việc lưu trữ Barrels Indexor và cập nhật cấu trúc Lexicon.

Đối từng Word Address Array trả về từ bước trên, truy xuất đến DocList đã có trong Barrels Indexor để cập nhật chúng và lưu mới vào Barrels Indexor. Song, cập nhật địa chỉ dlistAdd mới của của DocList vào cấu trúc Lexicon. Sau cùng là lưu cấu trúc Lexicon vào tệp Lexicon.

- Ưu điểm: Tiết kiệm bộ nhớ RAM trong quá trình indexing.
- Nhược điểm: Tốc độ index chậm hơn vì việc truy xuất và ghi liên tục vào tệp Barrels Indexor. Cần thêm bước ghi lại Barrels Indexor để loại bỏ các DocList cũ.

7. Quá trình Searching

Tìm kiếm văn bản phù hợp với dữ liệu người dùng nhập vào. Quá trình này sẽ tính toán điểm số cho các văn bản, các điểm số này cho biết mức độ thích hợp của document với query của người dùng. Các điểm số sẽ được tính toán dựa trên dữ liệu đã được trích xuất trong các siêu văn bản đã được tạo ra từ bước indexing.

a) Lấy dữ liệu của từ trong Lexicon và Barrels Indexor

Đối với từng token của query, chúng ta tìm dữ liệu của token đó trong Lexicon. Thuật toán tìm một từ trong Lexicon được trình bày như sau:

Thuật toán.

Bước 1: list = Lexicon.List (List đầu tiên chúng ta xét là List chứa các ký tự đầu tiên, có thể hiểu là tầng 1).
Bước 2: Đi qua từng ký tự trong token.
Bước 3: Tìm ký tự trong list. Trả về con trỏ Tnode chứa ký tự đó.
Bước 4: Nếu Tnode tồn tại thì list = Tnode→Branch.
    Bước 4.1: Nếu ký tự đang kiểm tra là ký tự cuối thì:
    - Nếu Tnode được đánh dấu isWord là true thì trả về Tnode. Đến bước 7.
    - Nếu không thì đến bước 6.
    Bước 4.2: Nếu không thì quay lại bước 2.
Bước 5: Nếu Tnode không tồn tại thì đến bước 6.
Bước 6: Trả về NULL. Đến bước 7.
Bước 7: Kết thúc.

Nhờ vào ưu điểm của Lexicon nên việc tìm kiếm từ trở nên dễ dàng và nhanh hơn.

Sau khi lấy thông tin của các token, chúng ta lấy ra các DocList tương ứng trong Barrel Indexor và tổng hợp chúng thành QDocList. Phần data của node trong QDocList là SDData. SDData sẽ chứa dữ liệu bao gồm:

- int docId: ID của document chứa token khớp với query.
- int nword: Số token mà document đó chứa khớp với query.

- Word Address Array: Mảng các địa chỉ của các token mà document đó chứa khớp với query.
- float score: Điểm số của document đó (Ban đầu giá trị này bằng 0).

b) Lấy dữ liệu chi tiết của từ trong Barrels và Doc Indexor.

Đối với từng SDData trong QDocList, chúng ta lấy địa chỉ đến vùng nhớ chứa dữ liệu của document trong Barrels thông qua Doc Indexor. Để lấy địa chỉ chính xác của từ trong Barrels, chúng ta cộng địa chỉ document và địa chỉ từ. Từ đó lấy ra các thông tin gồm: tf, mảng các vị trí của từ đó trong document.

c) Xác định đoạn tối thiểu (Minimal Interval)

Người dùng có thể nhập vào query không dấu nên một document có thể có nhiều token, bao gồm những token không liên quan đến query như ví dụ 2, đặc biệt là khi query có chứa stop word.

Để xử lý các token của document khớp với token của query, bao gồm cả document không khớp đủ số token của query, nhóm em nghiên cứu khái niệm Minimal Interval (đoạn tối thiểu) được xuất phát từ nghiên cứu [4]. Một đoạn (interval) là một đoạn chứa đủ số token mà document đó khớp.

Định nghĩa Minimal Interval: Một đoạn được gọi là Minimal Interval là đoạn không chứa các đoạn nhỏ hơn cũng là Minimal Interval.

Ví dụ: Một document có các token khớp với query "token1 token2 token3 token4" (document này chứa token4) như sau:

Token	token2	token2	token3	token1	token 2	token3	token2
Vị trí	2	10	11	19	20	21	90

Đoạn từ 2 đến 19 là không phải là Minimal Interval vì đoạn này chứa một đoạn nhỏ hơn, từ 10 đến 19, là Minimal Interval.

Việc xác định các đoạn Minimal Interval có thể giúp chúng ta đánh giá sự liên quan giữa các token mà document đó khớp và các token trong query.

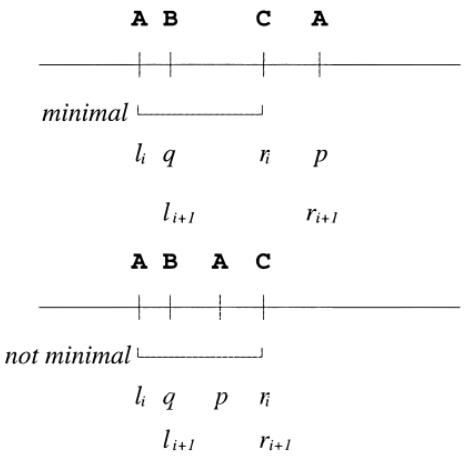
Để làm rõ thuật toán xác định Minimal Interval, chúng ta gọi tên cho các số liệu:

- $P_1, P_2, \ldots, P_k$ là các token mà document đó khớp. Mỗi token sẽ có một mảng các vị trí của token đó trong document.
- p_{ij} là vị trí của lần xuất hiện thứ j của token i trong document.
- [l..r] là mảng các token nằm trong đoạn từ vị trí l đến r trong văn bản.
- l_p, r_p là các token tương ứng nằm ở vị trí l và r của đoạn [l..r].

Thuật toán. (Dựa theo nghiên cứu [4])

Bước 1: Sắp xếp mảng tăng dần các vị trí của các token.
Bước 2: Lấy và xóa bỏ phần tử đầu tiên của các token p_{i1} (i = 1,.., k). Sắp xếp chúng và xác định $[l_1..r_1]$ từ các phần tử đã lấy ra. Đặt i = 1.
Bước 3: Nếu mảng vị trí của token l_p trong $[l_i..r_i]$ không còn phần tử thì đến bước 6. Nếu không thì gán cho p giá trị của phần tử đầu tiên của mảng vị trí của token l_p, xóa phần tử đó. Gán cho q là vị trí tiếp theo trong $[l_i..r_i]$.
Bước 4: Nếu $p > r_i$ thì $[l_i..r_i]$ là một minimal interval. Gán cho $l_{i+1} = q$, $r_{i+1} = p$.
Nếu không, gán cho $l_{i+1} = \min\{p, q\}$ và $r_{i+1} = r_i$ và cập nhật lại thứ tự của $[l_{i+1}..r_{i+1}]$.
Bước 5: Gán i = i + 1. Đến bước 3.

Bước 6: $[l_i..r_i]$ cũng là một minimal interval.



Hình 4: Ví dụ về thuật toán xác định minimal interval.

d) Tính điểm cho document
   - Tính điểm cho một minimal interval
     Điểm của một minimal interval được đánh giá dựa vào:
     - Điểm số token khớp với token của query

[Điểm số token khớp] = ([Số token khớp với query] / [Số token của query])$^{\text{Exponential matched token}}$

Trong đó, thông số Exponential matched token sẽ làm chênh lệch điểm giữa các document số lượng token khớp khác nhau. Thông số này càng cao thì các văn bản có số lượng token khớp với query nhiều sẽ có điểm này cao. Tuy nhiên, khi câu query chứa stop word thì không phải document nào chứa càng nhiều token khớp với query thì cũng tốt.

- Điểm số cặp token đúng thứ tự so với query

Một cặp token đúng thứ tự so với query là cặp token thỏa $0 \leq p_{i+1} - p_i - 1 \leq$ [Distance ordered pair].

[Điểm số token đúng thứ tự] = [Số cặp token đúng thứ tự so với query]$^{\text{Exponential Proximity}}$

Trong đó, p_i là vị trí của token thứ i trong minimal interval, sau khi đã sắp xếp lại theo đúng thứ tự token của query. Hiệu số này lớn hơn 0 tức là cặp từ đã đúng thứ tự. Tuy nhiên, để đảm bảo vẫn tìm được những trường hợp có stop word xen giữa các token trong query; có những từ khác ở giữa… Ví dụ: query "sinh viên đại học" thì vẫn đánh giá cao document chứa "sinh viên các trường đại học". Vì thế chúng ta vẫn cho phép cách xa nhau nhưng phải "đủ gần nhau". Thông số Distance ordered pair là giới hạn cho khoảng cách giữa hai token kế tiếp. Do đó, thông số này không được quá lớn để tránh việc đánh giá cao cho document không liên quan và cũng không quá nhỏ để tránh việc loại bỏ những document có khả năng liên quan. Ngoài ra, thông số [Exponential order pair] cũng rất quan trọng vì nó đóng

vai trò làm tăng mức độ ảnh hưởng của số cặp từ đúng thứ tự, mức độ quan trọng hơn tần số xuất hiện của từ.

- Khoảng cách giữa token bên trái cùng và token bên phải cùng

[Điểm khoảng cách] = 0 nếu $p_r - p_l >$ ([Distance minimal interval] + số token - 1)

[Điểm khoảng cách] = (Số token – 1) / $p_r - p_l$ nếu ngược lại

Trong đó, p_r và p_l là vị trí của token bên phải cùng và bên trái cùng của minimal interval đã được sắp xếp theo thứ tự vị trí tăng dần. Để không bỏ sót các trường hợp các token nằm cùng một câu nhưng không sát nhau nên thông số Distance minimal interval đã cho phép một khoảng cách tối đa giữa các token đó. Ví dụ: query "sinh viên đại học" thì vẫn phải đánh giá cao hơn cho document chứa "đại học sẽ tổ chức thi trực tuyến (online) cho các sinh viên sắp tốt nghiệp".

Cuối cùng, điểm cho một minimal interval được tính như sau.

[Điểm số token khớp] + [Điểm số token đúng thứ tự]

+ [Weight distance Minimal Interval] x [Điểm khoảng cách]

Tuy vậy, chúng ta thấy [Điểm khoảng cách] dường như chỉ chiếm một số điểm nhỏ vì hiện tại quá trình indexing vẫn chưa có phương pháp trích xuất phù hợp để phân biệt token trong một câu. Do đó, [Điểm khoảng cách] vẫn chưa là điểm ưu tiên cao khi cho điểm document, nhưng cũng có thể xem như là một điểm cộng để sắp xếp các document có các số điểm các bằng nhau.

- Tính điểm proximity của văn bản
  Điểm proximity của văn bản được đánh giá bằng cách tính trung bình điểm của các minimal interval.
- Tính điểm tf của văn bản
  Điểm tf của văn bản được tính bằng cách tính tổng tf của các token khớp với token của query.
- Tính điểm chung của văn bản
  Điểm của văn bản được tính theo công thức sau:

[Điểm của một văn bản] = [Điểm tf] x [Điểm proximity].

e) Sắp xếp
Sau khi tính điểm của các document, chúng ta có thể sắp xếp document theo thứ tự giảm dần. Với số lượng document lớn, nhóm em đã dùng thuật toán Merge Sort thông dụng để sắp xếp.

8. Chỉnh sửa corpus

Do tính chất linh hoạt của cấu trúc và sự kết nối giữa các siêu văn bản, việc thêm/xóa tệp văn bản vào corpus cũng trở nên dễ dàng hơn. Có thể có nhiều cách để thực hiện việc chỉnh sửa siêu văn bản, nhóm em đã chọn phương pháp chung sau:

- Những dữ liệu cần được cập nhật, chúng ta sẽ lấy dữ liệu của chúng từ vùng nhớ cũ. Cập nhật dữ liệu đó và ghi toàn bộ dữ liệu sau khi cập nhật vào vùng nhớ mới.
- Để xóa toàn bộ vùng nhớ cũ, chúng ta cần ghi lại tệp bằng cách tạo một tệp cache. Chúng ta sẽ đi qua các vùng nhớ đang được sử dụng để ghi chúng vào tệp cache. Tuy nhiên, địa chỉ cũ sẽ cần được điều chỉnh lại. Cuối cùng, chúng ta có thể đổi vai trò của tệp cache thành tệp chính và xóa hoàn toàn tệp chính cũ.

Việc thêm/xóa file của nhóm em đều tuân theo phương pháp này. Khi ghi lại Barrels thì phải cập nhật địa chỉ trong Doc Indexor. Khi ghi lại Barrel Indexor thì cập nhật địa chỉ trong Lexicon. Thực vậy, Doc Indexor và Lexicon có dung lượng rất nhỏ so với Barrels và Barrel Indexor nên khi cần cập nhật, chúng ta có thể đưa lên bộ nhớ RAM để xử lý và ghi lại hai tệp này một cách đơn giản.

III. Thử nghiệm các thông số

Trên thực tế có nhiều phương pháp đánh giá Search Engine, bao gồm nhiều phương pháp như đánh giá thủ công hoặc theo các công thức định sẵn. Theo nghiên cứu [2], phương pháp điều chỉnh các thông số, đánh giá Search Engine thủ công (thử từng query và đọc document để đánh giá sự liên quan của chúng). Vì vậy, nhóm em sẽ tự đánh giá thủ công với từng thông số quyết định đến sự hiệu quả của chương trình, rồi đưa ra con số hiệu quả nhất cho từng thông số.

1. Thử nghiệm thông số Threshold-stop-word:

Đây là thông số để loại bỏ các stop word một tiếng đi kèm với một tiếng có nghĩa khác. Theo công thức chung của quá trình xóa bỏ stop word, thông số này càng cao thì càng nhiều tiếng được bỏ, bao gồm các tiếng là stop word nhưng có tiềm năng mang lại ý nghĩa.

Nhóm chúng em lựa chọn 3 văn bản ngẫu nhiên có số lượng từ khác nhau và ít nhất một văn bản chứa từ có nghĩa đi chung với stop word một tiếng. Đó là các văn bản: MS_VNE_ (30), GT_VNE_ (27), GTTH_TT_ (214) với số lượng từ ban đầu tương ứng là 643, 2436, 252.

Văn bản	Giá trị của thông số Threshold-stop-word*				
	0	0.25	0.5	0.75	1
MS_VNE_ (30)	601	505	493	483	477
GT_VNE_ (27)	2210	1938	1918	1851	1811
GTTH_TT_ (214)	220	191	184	176	176

Bảng 1: Số lượng token còn lại sau quá trình bỏ stop word.

* Giá trị của Threshold two stop word bằng 0, tức là không loại bỏ 2 stop word một tiếng đi cùng nhau.

Nhận xét: Khi giá trị của thông số Threshold bằng 0, tức là chỉ loại bỏ các stop word 2 tiếng, văn bản vẫn chưa được lọc tốt vì còn chứa stop word một tiếng. Khi tăng Threshold lên 0,25 thì số lượng stop word một tiếng đã được loại bỏ khoảng 20%. Tuy loại bỏ nhiều như thế, các từ có nghĩa như “thiết bị”, “đun nước”, “sự nghiệp” vẫn được giữ lại. Số lượng từ được lược bỏ dần ít đi khi tăng giá trị thông số lên. Khi lên đến 1 thì các từ stop word một tiếng đã được loại bỏ hẳn, dẫn đến mất nghĩa của nhiều từ quan trọng. Điều tương tự cũng diễn ra với giá trị 0.75 khi cụm từ “không dây” trong văn bản 3 đã chuyển thành “dây”, tức là mất đi ý nghĩa ban đầu. Do đó, giá trị phù hợp nhất để lọc được nhiều stop word một tiếng và giữ được ý nghĩa của các từ là 0.5.

2. Thử nghiệm thông số Threshold-two-stop-word

Sau khi đã loại bỏ các stop word một tiếng đi chung với một tiếng có nghĩa khác. Chúng ta cần phải loại bỏ trường hợp 2 stop word một tiếng đi chung với nhau nhưng không tạo ý nghĩa trong văn bản.

Do giá trị của thông số Threshold stop word là 0.5 nên ban đầu, số lượng từ của các văn bản lần lượt là 493, 1918, 184.

Văn bản	Giá trị của thông số Threshold-two-stop-word				
	0.2	0.4	0.6	0.8	1
MS_VNE_ (30)	420	413	410	403	403
GT_VNE_ (27)	1605	1427	1333	1267	1267
GTTH_TT_ (214)	154	150	146	138	138

Bảng 2: Số lượng token còn lại sau quá trình bỏ stop word.

Nhận xét: Tương tự như thông số Threshold stop word, khi tăng dần giá trị thông số của Threshold two stop word thì số lượng từ bị loại bỏ tăng lên ít hơn. Tuy nhiên, khi chuyển từ 0.6 lên 0.8 thì số lượng từ bị loại bỏ tăng nhiều hơn bất thường, đó là do các từ có nghĩa được tạo thành từ 2 stop word một tiếng như "người dùng" đã bị loại bỏ trong khi nó đã đóng vai trò quan trọng trong văn bản 3. Do đó, giá trị phù hợp nhất cho thông số này là 0.6.

3. Thử nghiệm với weight-distance-minimal-interval

Để tự đánh giá được hiệu quả hơn, nhóm em chọn các query điển hình cho từng thông số, việc này sẽ giúp ta thấy được rõ ràng hơn sự ảnh hưởng của từng thông số. Dưới đây là các query thử nghiệm:

Query 1: hoa ky viet nam → Query chứa 2 từ ghép có thể tách xa nhau.

Query 2: truong pho thong → Query chứa cụm từ có thể có từ xen giữa.

Query 3: trinh cong son la ai → Query chứa từ có nghĩa và ít stop words.

Query 4: ngay mai la ngay em dep nhat → Query chứa chủ yếu stop words.

Chúng ta sẽ bắt đầu thử nghiệm với thông số weight-distance-minimal-interval trước với các giá trị cơ bản của các thông số còn lại (chuyển các thông số còn lại về giá trị cơ bản sao cho nó không ảnh hưởng đến điểm tổng của 1 document), sau đó sẽ đánh giá và giữ lại giá trị tốt nhất của thông số vừa thử nghiệm. Sau bước thử nghiệm thông số cuối cùng thì chúng ta sẽ chốt lại bộ thông số hiệu quả nhất vừa được thử nghiệm và đánh giá và lấy bộ giá trị đó đưa vào chạy chương trình tổng.

	Giá trị của weight distance minimal interval			
	1	2	5	10
query 1	Du lich\DL_VNE_ (219) Giao duc\GDu_TN_ (465) Chung khoan\CK_VNE_ (171)	Du lich\DL_VNE_ (219) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)	Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473) Bong da\BD_VNE_ (487)	Giao duc\GDu_TN_ (473) Du lich\DL_TN_ (282) Bong da\BD_VNE_ (487)
query 2	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Giao duc\GDu_NLD_ (682)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)
query 3	Am nhac\AN_VNE_ (256) Khong gian song\KGS_VNE_ (46) Am nhac\AN_VNE_ (601)	Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (256) Am nhac\AN_VNE_ (601)	Am nhac\AN_VNE_ (610) Chung khoan\CK_VNE_ (30) Am nhac\AN_VNE_ (256)	Chung khoan\CK_VNE_ (30) Hinh su\HS_VNE_ (129) Chung khoan\CK_VNE_ (171)
query 4	Am nhac\AN_VNE_ (232)	Am nhac\AN_VNE_ (232)	Am nhac\AN_VNE_ (232)	Am nhac\AN_VNE_ (232)

	Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)

Bảng 3. Ba document phù hợp nhất được trả về.

Nhận xét: Qua kết quả các document trả về cho từng query cũng như nội dung của các document đó, nhóm em nhận thấy giá trị của thông số weight distance minimal interval này bằng 5 là hiệu quả nhất, xuất hiện đầy đủ các từ trọng yếu của query, riêng query 4 không thay đổi bởi vì query 4 chứa nhiều stop words nằm cách xa nhau. Vì thế giá trị cần chọn cho thông số này là 5.

4. Thử nghiệm với distance minimal interval

	Giá trị của distance minimal interval			
	2	5	10	20
query 1	Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473) Bong da\BD_VNE_ (487)	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)
query 2	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)
query 3	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)
query 4	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)

Bảng 4. Ba document phù hợp nhất được trả về.

Nhận xét: Tuỳ vào giá trị của distance-minimal-interval mà các document tốt nhất trả về là khác nhau, nhưng dựa vào bảng trên ta thấy được, các document tốt nhất chỉ thay đổi vị trí cho nhau, chỉ có vài trường hợp bị thay thế. Và nội dung của các document thuộc khoảng thông số từ 5-10 trở về sau là khá sát với query, đáp ứng đầy đủ nội dung của query và tốt hơn các document ở các khoảng thông số còn lại. Và qua đó ta suy ra số liệu phù hợp nhất cho thông số này là nằm trong khoảng từ 5-10 trở về sau, ta có thể lấy 10 làm giá trị cố định để đưa vào chạy chương trình.

5. Thử nghiệm với distance order pair

	Giá trị của distance order pair			
	1	2	4	10
query 1	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Giao duc\GDu_TN_ (473)	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Du lich\DL_VNE_ (219)	The gioi tre\TGT_TN_ (169) Du lich\DL_VNE_ (219) Du lich\DL_TN_ (282)
query 2	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Giao duc\GDu_NLD_ (812)
query 3	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)

query 4	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)

Bảng 5. Ba document phù hợp nhất được trả về.

Nhận xét: Qua bảng kết quả cũng như nội dung của các document tiêu biểu trong bảng, ta thấy mặc dù kết quả các document trả về không nhiều nhưng sự thay đổi nhỏ nhưng hiệu quả hơn hết ứng với giá trị 4 của thông số distance order pair này. Qua đó chúng ta có thể chọn 4 là giá trị cho thông số này.

6. Thử nghiệm với exponential-order-pair

	Giá trị exponential order pair			
	1	5	10	20
query 1	The gioi tre\TGT_TN_ (169) Du lich\DL_TN_ (282) Du lich\DL_VNE_ (219)	Du lich\DL_VNE_ (219) Giai tri tin hoc\GTTH_TT_ (332) Giao duc\GDu_NLD_ (607)	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)
query 2	Giao duc\GDu_NLD_ (802) Giao duc\GDu_VNE_ (171) Am thuc\AT_TT_ (247)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)
query 3	Am nhac\AN_VNE_ (256) Chung khoan\CK_VNE_ (30) Cuoc song do day\CSDD_VNE_ (218)	Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (256)	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)
query 4	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)

Bảng 6. Ba document phù hợp nhất được trả về.

Nhận xét: Ta thấy được, khi thay đổi thông số này thì các document trả về thay đổi rất nhiều, nhưng nội dung của các document đó rất tốt, đáp ứng tốt hơn nội dung của query. Và thứ tự các document tốt nhất được sắp xếp chuẩn nhất là ở giá trị 10 của thông số exponential order pair này. Vì thông số này quan trọng hơn hết nên sự thay đổi của các document tốt nhất là rất lớn qua các khoảng giá trị của thông số nhưng bù lại rất hiệu quả. Vì thế, nhóm em suy ra được giá trị cần chọn cho thông số này là 10.

7. Thử nghiệm với exponential-match-token

	Giá trị của exponential match token			
	1	5	10	20
query 1	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)	Giao duc\GDu_NLD_ (607) The gioi tre\TGT_TT_ (178) Giai tri tin hoc\GTTH_TT_ (332)
query 2	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)	Giao duc\GDu_VNE_ (313) Giao duc\GDu_NLD_ (682) Mua sam\MS_VNE_ (146)
query 3	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)	Am nhac\AN_VNE_ (46) Am nhac\AN_VNE_ (610) Am nhac\AN_VNE_ (698)
query 4	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46) Bong da\BD_TN_ (1618)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46)	Am nhac\AN_VNE_ (232) Bat dong san\BDS_VNE_ (46)

		Bong da\BD_TN_ (1618)	Bong da\BD_TN_ (1618)	Bong da\BD_TN_ (1618)

Bảng 7. Ba document phù hợp nhất được trả về.

Nhận xét: Từ bảng 5 và dữ liệu của các document trong bảng ta thấy các document tốt nhất được trả về không thay đổi hoặc có thay đổi rất ít, nhưng các document trả về tiếp theo đó thì vẫn có sự thay đổi nhẹ, chúng hướng tới bản hoàn thiện nhất ứng với giá trị 5 của thông số cuối cùng này. Qua đó ta suy ra số liệu cần cho thông số này là 5.

Cuối cùng ta có một bộ thông số hữu hiệu nhất để chạy chương trình là

+ Threshold-stop-word 0.5
+ Thrshold-two-stop-word 0.6
+ Weight-distance-minimal-interval: 5
+ Distance-minimal-interva: 10
+ Distance-order-pair: 4
+ Exponential-order-pair: 10
+ Exponential-match-token: 5

Các thông số này chúng bổ sung và hoàn thiện nhau, đưa tới một kết quả chạy rất thích hợp với nội dung tìm kiếm.

IV. Tham khảo:

Trang web:

[1] https://github.com/stopwords/vietnamese-stopwords

Nghiên cứu:

[2] Sergey Brin and Lawrence Page (1998). "The Anatomy of a Large-Scale Hypertextual Web Search Engine". Computer Science Department, Standford University.

[3] Hung Nguyen Thanh and Khanh Bui Doan (2006). "Word Segmentation for Vietnamese Text Categorization – An internet-based statistic and genetic algorithm approach". VNU-HCM, University of Paris.

[4] Hiroshi IMAI and Kunihiko SADAKANE (2001). "Fast Algorithms for k-Word Proximity Search". IEICE TRANS. University of Tokyo.