ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

---------- ☙★❧ ----------

BÁO CÁO ĐỒ ÁN MÔN HỌC
TRUY XUẤT THÔNG TIN

ĐÁNH GIÁ MÔ HÌNH KHÔNG GIAN VECTOR & LSI TRÊN TẬP DỮ LIỆU CRANFIELD

Lớp: CS419.N11

Giảng viên hướng dẫn: ThS. Nguyễn Trọng Chỉnh

Nhóm sinh viên thực hiện:

1. Phan Thanh Hải 18520705
2. Nguyễn Hoàng Long 20520239

TP. Hồ Chí Minh, tháng 03 năm 2023

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

---------- ☙★❧ ----------

BÁO CÁO ĐỒ ÁN MÔN HỌC
TRUY XUẤT THÔNG TIN

ĐÁNH GIÁ MÔ HÌNH KHÔNG GIAN VECTOR & LSI TRÊN TẬP DỮ LIỆU CRANFIELD

Lớp: CS419.N11

Giảng viên hướng dẫn: ThS. Nguyễn Trọng Chỉnh

Nhóm sinh viên thực hiện:

1. Phan Thanh Hải 18520705
2. Nguyễn Hoàng Long 20520239

TP. Hồ Chí Minh, tháng 03 năm 2023

MỤC LỤC

DANH MỤC HÌNH ẢNH

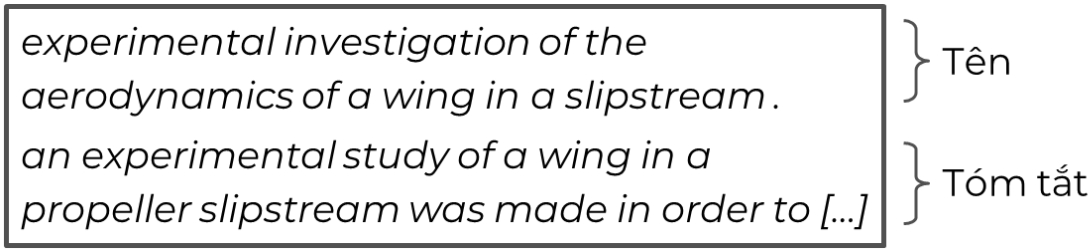
DANH MỤC BẢNG BIỂU

CHƯƠNG 1.
PHÂN TÍCH TẬP DỮ LIỆU

Tập tài liệu Cranfield được sử dụng trong đồ án này chứa toàn bộ danh sách tài liệu và câu truy vấn lấy từ trường Đại học Cranfield, Anh Quốc. Tập tài liệu Cranfield bao gồm các thông tin sau:

1.1. Kho ngữ liệu

Khu ngữ liệu gồm 1400 đoạn tóm tắt từng tài liệu liên quan đến chủ đề về khí động lực học. Mỗi tài liệu sẽ có cấu trúc gồm 2 phần: tên tài liệu và mô tả tóm tắt của tài liệu như **Hình 1.1**.



Hình 1.1. *Cấu trúc và định dạng của mỗi tài liệu*

Mỗi tài liệu đã được xử lý ở bước biến đổi về chữ thường, tức là chuyển các kí tự in hoa và kí tự thường.

Kết quả thống kê của nhóm với kho ngữ liệu.

Tài liệu 798 có kích thước lớn nhất: 655 từ (token được phân tách bởi dấu cách).

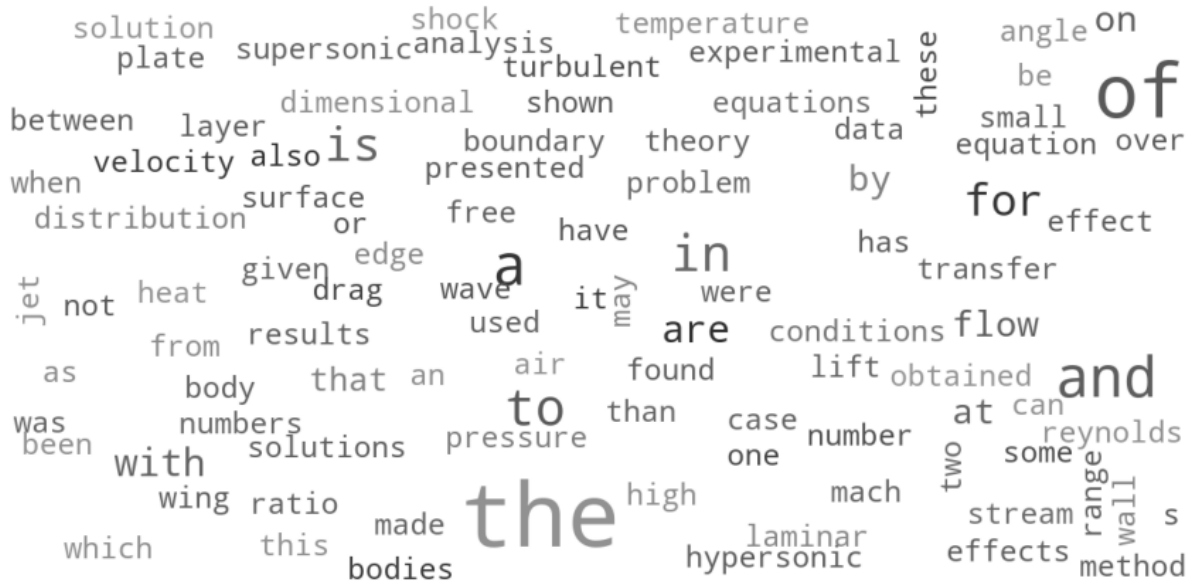
Có 2 tài liệu rỗng: 471 và 995, chiếm tỉ lệ 0,14%.

Một số điểm đặc biệt của kho ngữ liệu này với các kho ngữ liệu thông thường là xuất hiện các loại đối tượng/kí tự đặc biệt được nêu trong **Bảng 1.1**.

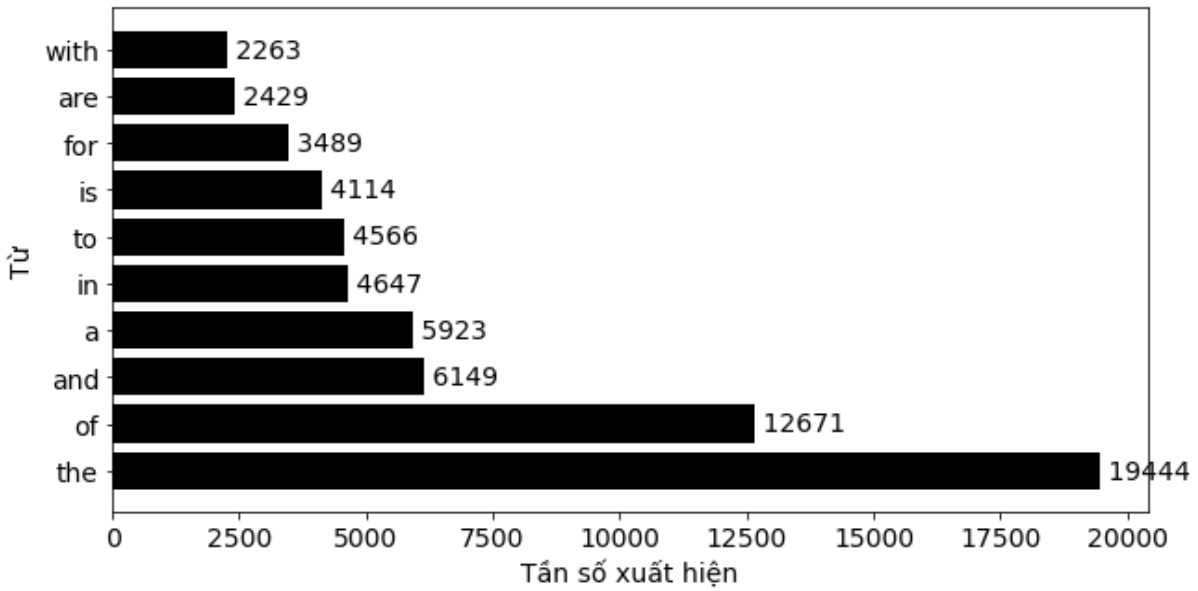
Bảng 1.1. *Các loại đối tượng đặc biệt xuất hiện trong kho ngữ liệu*

Đối tượng đặc biệt	Ví dụ
Đơn vị đo	*at speeds up to 25,000* ***ft sec***
Biến trong công thức	*the main flow exceeding* ***1 1/m*** *(where* ***m*** *is the […]*
Đánh số đối tượng	*eqs.* ***(2)*** *and* ***(3)*** *is rather confusing […] to table* ***2***
Dấu nhân	*varied from 2.35* ***x*** *10 to 2.99* ***x*** *10*

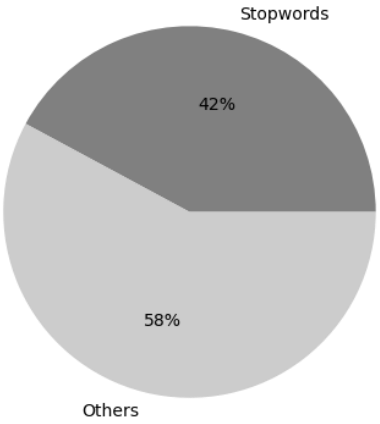
Vì đặc thù chuyên ngành nên kho ngữ liệu có xuất hiện các đơn vị đo, công thức tính toán và các biến có trong công thức. Ngoài ra, kho ngữ liệu còn có đánh số các đối tượng như đánh số công thức, đánh số bảng… Dấu nhân trong tập tài liệu thay vì được mã hóa thành ký tự × hoặc * riêng biệt thì bên xử lý đã mã hóa sang ký tự x. Dựa trên điểm đặc biệt của kho ngữ liệu, ta cần có bước tiền xử lý thích hợp.



Hình 1.2. *Sử dụng WordCloud để trực quan hóa tần số xuất hiện của top 100 từ*

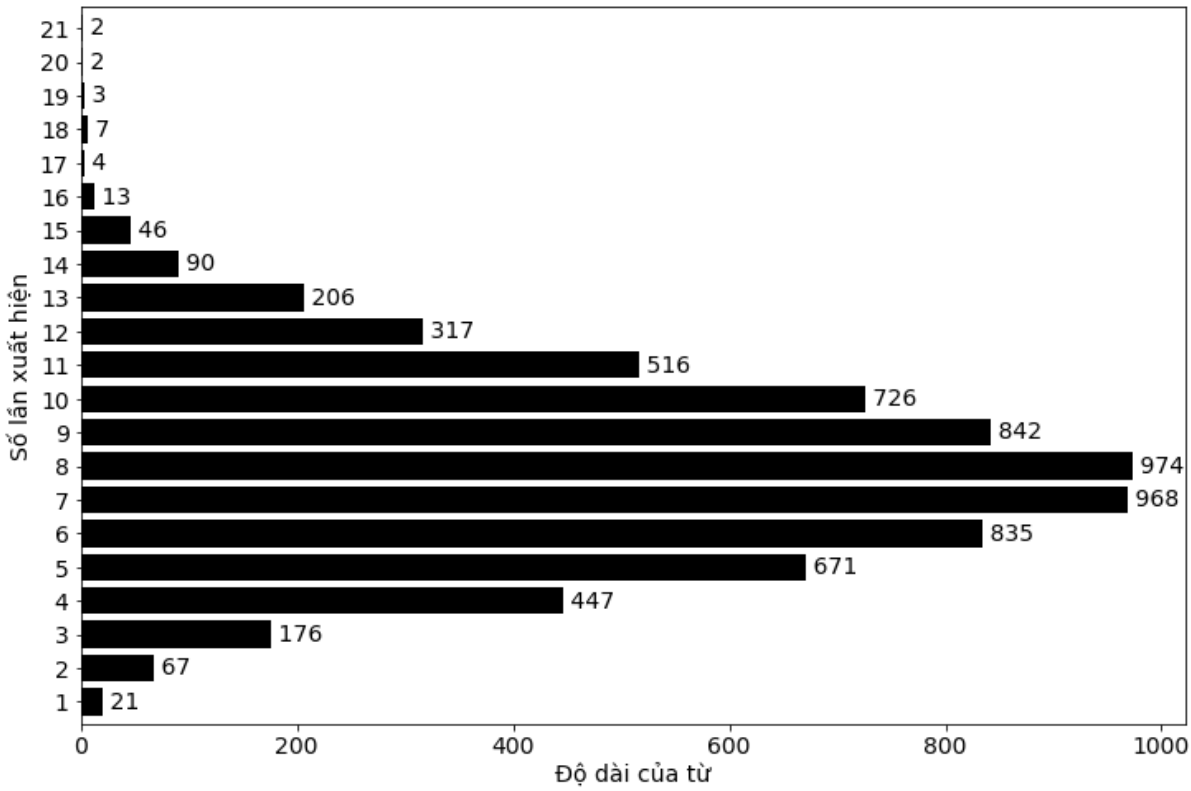


Hình 1.3. *Biểu đồ tần số xuất hiện của top 10 từ xuất hiện nhiều nhất*

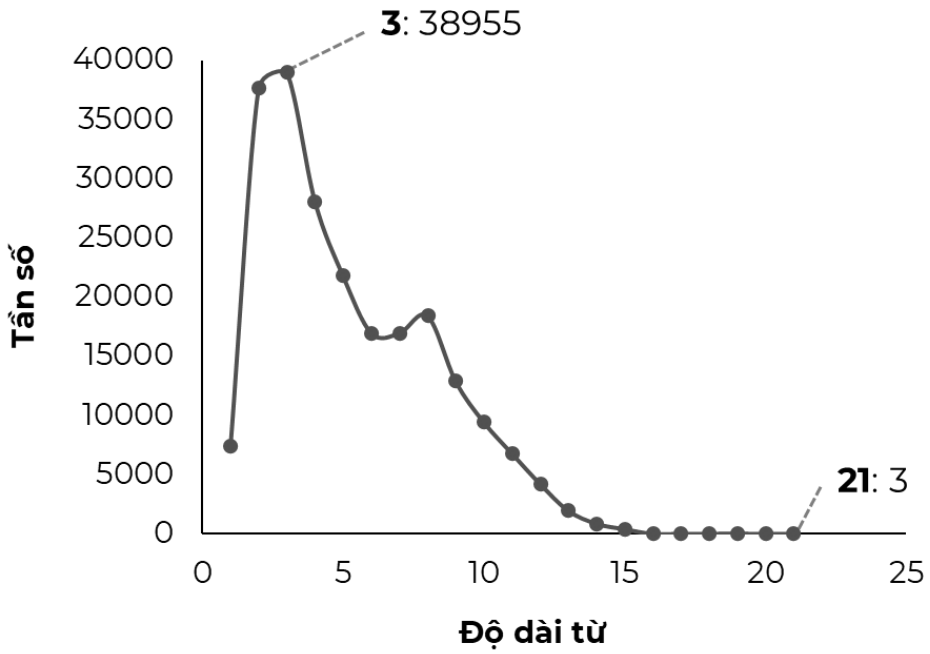


Hình 1.4. *Tỉ lệ stopwords có trong kho ngữ liệu*

Stopwords hiểu đơn giản là các từ có tần số xuất hiện nhiều như the, to... các từ này thường mang ít giá trị ý nghĩa và không khác nhau nhiều trong các văn bản khác nhau. Ví dụ từ "the" hay "to" thì ở văn bản nào nó cũng không bị thay đổi về ý nghĩa. Nhóm sử dụng danh sách các stopword tiếng Anh có trong thư viện NLTK để tiến hành thống kê. Dựa vào kết quả phân tích ở các hình, ta thấy stopword chiếm tỉ lệ cao, gần bằng một nửa số lượng từ có trong kho ngữ liệu. Ngoài ra, danh sách top 10 từ xuất hiện nhiều nhất trong kho ngữ liệu cũng lại là 10 từ trong danh sách stopword.



Hình 1.5. *Biểu đồ cột thể hiện tương quan giữa tần số xuất hiện và độ dài của từ tương ứng*



Hình 1.6. *Biểu đồ đường thể hiện tương quan giữa tần số xuất hiện và độ dài của từ tương ứng*

Dựa vào 2 biểu đồ trên, ta thấy thường những từ nào có độ dài càng ngắn thì tần số xuất hiện càng nhiều và ngược lại. Điều này kiểm chứng thực nghiệm cho luật Zipf được đề

xuất vào năm 1945 bởi nhà ngôn ngữ học George Zipf. Luật quy định rằng một từ càng thường xuyên được sử dụng trong bất kỳ ngôn ngữ nào thì nó càng có xu hướng có độ dài ngắn hơn.

1.2. Câu truy vấn

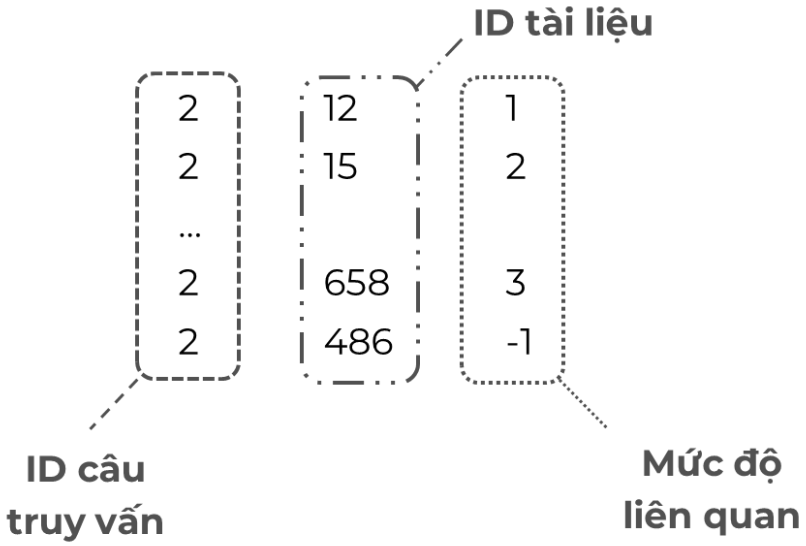
Trong tập tài liệu Cranfield có tất cả 225 câu truy vấn được lưu trữ vào trong 1 tập tin văn bản duy nhất. Trong đó, mỗi câu truy vấn sẽ nằm trên một dòng khác nhau.

Bảng 1.2. *Các loại câu hỏi truy vấn*

Loại câu truy vấn	Ví dụ
Câu hỏi	***is*** *it possible to predict when and how it [...]*
Cụm danh từ	*material* ***properties*** *of photoelastic […]*
Cụm động từ	***find*** *a calculation procedure applicable to […]*

1.3. Kết quả truy vấn đúng

Kết quả truy vấn đúng tương ứng được lưu trữ vào 225 tập tin văn bản tương ứng với mỗi câu truy vấn. Trong mỗi tập tin văn bản sẽ có 3 trường thông tin như trong



Hình 1.7. *Các trường thông tin có trong tập tin kết quả truy vấn đúng*

Trong đồ án này, nhóm chỉ sử dụng 2 trường thông tin ID đầu tiên để tiến hành đánh giá hiệu quả của các mô hình truy xuất thông tin.

CHƯƠNG 2.
XÁC ĐỊNH TERM CỦA TÀI LIỆU

Term được định nghĩa là đơn vị cơ bản của tìm kiếm. Như vậy, ta có thể xem tài liệu là tập hợp các term. Để xác định term của tài liệu, nhóm thực hiện lần lượt các bước sau:

2.1. Tách token

Tách token là quá trình tách tài liệu thành các đơn vị nhỏ hơn, gọi là token. Trong quá trình tách này, ta có thể loại bỏ một số ký tự nhất định tùy vào mục đích của người thực hiện. Với ngôn ngữ là tiếng Anh, nhóm tiến hành tách văn bản thành các từ cách nhau bởi dấu cách. Nhóm cũng tiến hành loại bỏ các ký tự là dấu câu và các con số.

Ví dụ.

Cho đoạn văn bản sau có trong tập tài liệu Cranfield:

extended up to 400,000 btu slug, corresponding to

Kết quả sau khi tách token dựa trên quy tắc trên:

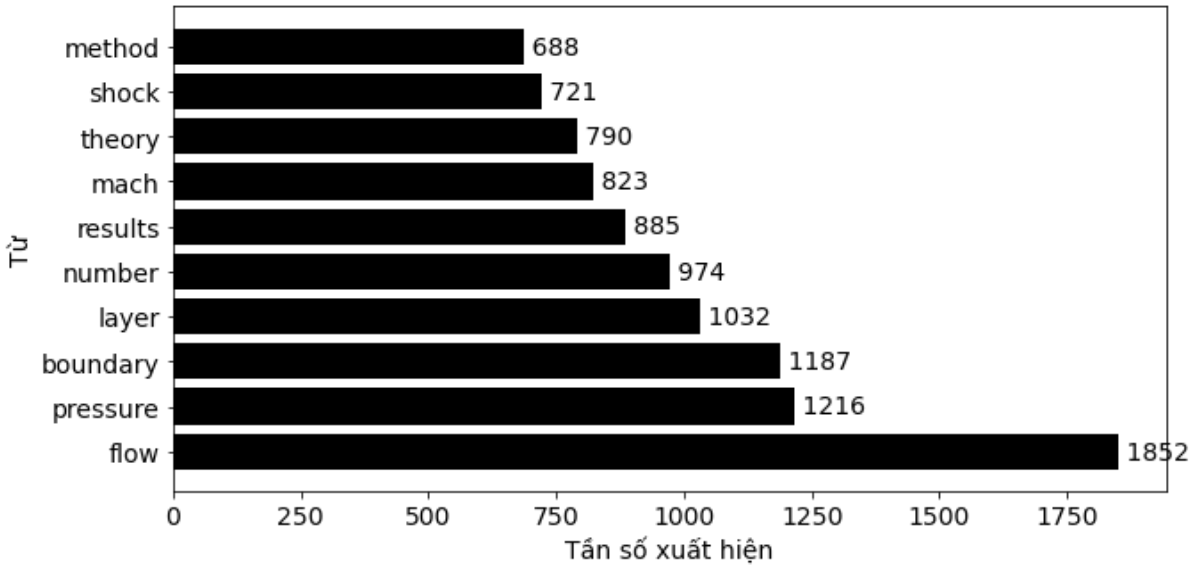
extended up to btu slug corresponding to

Nhóm đã tách được tổng là 7052 token.

2.2. Loại bỏ stop word

Ta nhận thấy các stop word mang rất ít giá trị khi chọn tài liệu phù hợp với nhu cầu của người dùng. Ngoài ra, việc loại bỏ stop word giúp giảm chi phí lưu trữ số lượng postings. Trong đa số trường hợp, việc loại bỏ stop word sẽ **không** gây ảnh hưởng tới kết quả tìm kiếm. Do đó, nhóm đã sử dụng danh sách stop word có sẵn trong thư viện NLTK để tiến hành đối chiếu với kho ngữ liệu, thực hiện việc loại bỏ stop word tương ứng.

Hình 2.1. *Sử dụng WordCloud để trực quan hóa tần số xuất hiện của những từ không phải stopwords*



Hình 2.2. *Biểu đồ cột thể hiện tương quan giữa tần số xuất hiện và độ dài của từ tương ứng (những từ không phải stopwords)*

Hình trên đây thống kê những từ hay xuất hiện trong kho ngữ liệu sau khi đã thực hiện bước loại bỏ stop word bao gồm các từ: method, shock, theory, mach, results, number, layer, boundary, pressure, flow…

2.3. Chuẩn hóa token

Quá trình này thực chất là chuẩn hóa cùng một token về dạng nguyên bản của nó.

Ví dụ.

Trong tập tài liệu Cranfield có xuất hiện 2 từ là experiment và experiments (dạng số nhiều của từ experiment). Hai từ này tuy cùng nghĩa nhưng về mặt hình thái thì được xem là 2 token khác nhau. Do đó, ta cần đưa 2 token này về dạng nguyên bản để tiện trong việc xử lý và lưu trữ term.

Có 2 cách chuẩn hóa token thường được sử dụng là stemming và lemmatization.

- *Stemming* là kỹ thuật dùng để biến đổi 1 từ về dạng gốc (được gọi là stem hoặc root form) bằng cách cực kỳ đơn giản là loại bỏ 1 số ký tự nằm ở cuối từ mà nó nghĩ rằng là biến thể của từ. Ví dụ như chúng ta thấy các từ như walked, walking, walks chỉ khác nhau là ở những ký tự cuối cùng, bằng cách bỏ đi các hậu tố -ed, -ing hoặc -s, chúng ta sẽ được từ nguyên gốc là walk. Người ta gọi các bộ xử lý stemming là Stemmer. Bởi vì nguyên tắc hoạt động của stemmer rất là đơn giản như vậy cho nên tốc độ xử lý của nó rất là nhanh, và kết quả stem đôi khi không được như chúng ta mong muốn. Chẳng hạn như từ goes sẽ được stem thành từ goe (bỏ chữ s cuối từ) trong khi đó stem của từ go vẫn là go, kết quả là 2 từ "goes" và "go" sau khi được stem thì vẫn không giống nhau. Một nhược điểm khác là nếu các từ dạng bất quy tắt như went hay spoke thì stemmer sẽ không thể đưa các từ này về dạng gốc là go hay speak.
- Tuy có các nhược điểm như trên nhưng nhóm vẫn sử dụng phương pháp để chuẩn hóa token vì nó có tốc độ xử lý nhanh và kết quả cuối cùng nhìn chung không hề tệ khi so với Lemmatization. Khác với Stemming là xử lý bằng cách loại bỏ các ký tự cuối từ một cách rất heuristic, *Lemmatization* sẽ xử lý thông minh hơn bằng một bộ từ điển hoặc một bộ ontology nào đó. Điều này sẽ đảm bảo rằng các từ như "goes", "went" và "go" sẽ chắc chắn có kết quả trả về là như nhau. Kể các từ danh từ như mouse, mice cũng đều được đưa về cùng một dạng như nhau. Người ta gọi bộ xử lý lemmatization là lemmatizer. Nhược điểm của lemmatization là tốc độ xử lý khá chậm vì phải thực hiện tra cứu từ trong cơ sở dữ liệu.

Nhóm đã sử dụng bộ **Snowball Stemmer** trong **NLTK** (dựa trên thuật toán **Porter Stemmer** cải tiến) để tiến hành chuẩn hóa token.

Ví dụ.

Bảng 2.1. *Ví dụ kết quả thực hiện stemming*

Cranfield (tài liệu 3)	the boundary layer in simple shear flow past a flat plate . the boundary layer equations are presented for steady incompressible flow with no pressure gradient .
Kết quả xử lí	boundari layer simpl shear flow past flat plate boundari layer equat present steadi incompress flow pressur gradient

Sau khi đã tiến hành 3 bước trên, nhóm thu được kết quả là 4147 term (tỉ lệ ~ 58% so với số token được thực hiện ở bước 1).

CHƯƠNG 3.
LẬP CHỈ MỤC

Chỉ mục là cấu trúc dữ liệu chuyên biệt để tối ưu hóa tốc độ thực hiện truy vấn. Với mỗi mô hình truy xuất thông tin khác nhau, ta có cấu trúc chỉ mục khác nhau.

3.1. Chỉ mục đảo ngược

Với mô hình không gian vector, nhóm sử dụng cấu trúc chỉ mục đảo ngược. Trong đó:
Với mỗi term *t* sẽ lưu trữ thông tin tài liệu có chứa *t*.
Mỗi tài liệu được biểu diễn bằng 1 con số (bắt đầu từ 1).
Ví dụ.
`'experiment' → [1, 11, 12, 16, 17, 191, 25, 29]`
Danh sách tài liệu có chứa term t được gọi là posting list.
Posting list cần được sắp xếp để truy xuất thông tin một quả hiệu quả.
Để lưu trữ chỉ mục đảo ngược hiệu quả và tra cứu nhanh phần tử trong chỉ mục đảo ngược, người ta thường sử dụng cấu trúc bảng băm và cây để xây dựng chỉ mục đảo ngược. Trong đồ án này, nhóm sử dụng cấu trúc bảng băm vì tốc độ truy xuất của bảng băm nhanh, độ phức tạp thời gian là O(1). Trong Python, ta sử dụng kiểu dữ liệu *dictionary* để xây dựng cấu trúc bảng băm.
Gọi tên biến để lưu trữ chỉ mục đảo ngược là inverted_index. Biến này sẽ có cấu trục như sau:

```
inverted_index = {
<term1>: { <chỉ số tài liệu 1>: <thông tin term>,
           <chỉ số tài liệu 2>: <thông tin term>,
     … }
… }
```

Ví dụ.

```
inverted_index = {'experimental': {1: 2, 11: 1, 12: 1, 17: 1, 19: 1}}
```

3.2. Ma trận tài liệu

Với mô hình chỉ mục ngữ nghĩa ngầm, nhóm sử dụng ma trận tài liệu để tiến hành xây dựng cấu trúc chỉ mục. Trong đó:

- Mỗi hàng tương ứng với từng term có trong tài liệu.
- Mỗi cột tương ứng với từng tài liệu.
- Giá trị của phần tử *A[i, j]* trong ma trận thể hiện mức độ quan trọng của term. Có nhiêu trọng số để biểu diễn giá trị này như giá trị nhị phân… Nhưng nhóm sử dụng trọng số TF-IDF. (dựa trên thông tin ở mục 4.1.)

CHƯƠNG 4. MÔ HÌNH TRUY XUẤT THÔNG TIN

Mỗi mô hình truy xuất thông tin có 4 đặc trưng cơ bản sau:

(D, Q, F, R)

Trong đó:

D: Cách biểu diễn văn bản

Q: Cách biểu diễn truy vấn

F: Nền tảng lí thuyết (toán học) tương thích với D và Q, giữ vai trò cơ sở để thực hiện các suy diễn xếp hạng

R(D, Q): Hàm xếp hạng, là hàm định lượng mức độ phù hợp giữa văn bản và truy vấn

4.1. Mô hình không gian vector

D: Ta xem mỗi tài liệu như là 1 vector thưa. Các phần tử trong vector thể hiện mức độ quan trọng của term có trong tài liệu.

Q: Như D.

F: Lí thuyết toán học về không gian vector.

R(D, Q): Hàm xếp hạng dựa trên độ tương đồng/giống nhau giữa 2 vector trong không gian.

Với mô hình này, các phần tử trong vector mang một trọng số term frequency *tf(t, d)*. Term frequency *tf(t,d)* xác định số lần từ *t* xuất hiện trong tài liệu *d*. Nhưng chỉ tần suất xuất hiện của một từ thôi thì chưa đủ.

Ví dụ trong một tài liệu, sự xuất hiện của một từ 10 lần thì tài liệu đó được coi là phù hợp hơn tài liệu mà từ đó chỉ xuất hiện 1 lần. Nhưng không phải là phù hợp hơn tài liệu kia 10 lần. Sự phù hợp không tỷ lệ thuận với số lần xuất hiện của từ đó trong một tài liệu. Vì thế, nhóm sử dụng phương pháp tính trọng số tần suất logarit (log-frequency). Log-frequency của một term *t* trong document *d* được tính như sau:

$$tf(t,d) = \log(1 + f_{t,d})$$

Nếu từ đó không xuất hiện trong một tài liệu, thì *tf(t,d)* bằng 0. Và bởi vì log(0) là một số âm vô cùng, cho nên chúng ta phải cộng 1.

Phương pháp tính trọng số nghịch đảo văn bản (Inverse document weighting)

Từ hiếm thì quan trọng hơn những từ có tần suất xuất hiện cao. Trong mỗi ngôn ngữ có những từ lặp đi lặp lại nhiều lần nhưng vô nghĩa (ví dụ trong tiếng Anh là a, the, to, of v.v). Đối với term frequency, thì những từ càng xuất hiện nhiều thì có điểm càng cao, còn những từ hiếm thì điểm xếp hạng lại thấp hơn. Do đó chúng ta cần một cách đánh giá khác với các từ hiếm, vì nó sẽ mang nhiều thông tin hơn là những từ phổ biến trong văn bản.

Ý tưởng là giảm trọng số của từ nào có document frequency cao, bằng cách lấy tổng số tài liệu (N) chia cho số tài liệu mà một từ xuất hiện.

Chúng ta định nghĩa trọng số idf của một từ t bởi:

$$idf(t,d) = log \frac{N}{1 + |\{d \in D, t \in d\}|}$$

Nhóm sử dụng độ đo góc/độ đo cosin để tiến hành xếp hạng tài liệu. Độ đo tương đồng được thể hiện thông qua việc tính cosin góc giữa 2 vector.

$$cos(x,y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

Ví dụ. Cho 3 tập tài liệu sau

d1: “new york times”

d2: “new york post”

d3: “los angeles times”

Trọng số IDF:

angles: log2(3/1)=1.584

los: log2(3/1)=1.584

new: log2(3/2)=0.584

post: log2(3/1)=1.584

times: log2(3/2)=0.584

york: log2(3/2)=0.584

Trọng số TF là tần số xuất hiện của term trong tài liệu (dễ tính được), ta được bảng trọng số TF-IDF như sau:

	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

Ta tính độ lớn của từng tài liệu và câu truy vấn theo trọng bộ trọng số TF-IDF.

d1 = sqrt(0.584^2+0.584^2+0.584^2)=1.011
d2 = sqrt(0.584^2+1.584^2+0.584^2)=1.786
d3 = sqrt(1.584^2+1.584^2+0.584^2)=2.316
q = sqrt(0.584^2+0.292^2)=0.652

Sau đó so khớp tập tài liệu với câu truy vấn theo công thức Cosin như sau

cosSim(d1,q) = (0*0+0*0+0.584*0.584+0*0+0.584*0.292+0.584*0) / (1.011*0.652) = 0.776
cosSim(d2,q) = (0*0+0*0+0.584*0.584+1.584*0+0*0.292+0.584*0) / (1.786*0.652) = 0.292
cosSim(d3,q) = (1.584*0+1.584*0+0*0.584+0*0+0.584*0.292+0*0) / (2.316*0.652) = 0.112

Sau cùng ta được kết quả trả về được sắp xếp như sau: d1, d2, d3.

Trên thực tế, việc tính toán như vậy không khả quan. Do đó, nhóm cộng dồn các trọng số TF-IDF để tiến hành xếp hạng tài liệu dựa trên thuật toán sau:

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5     for each pair(d, tf_{t,d}) in postings list
 6     do Scores[d] += w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

Hình 4.1. *Mã giả thuật toán xếp hạng kết quả của mô hình không gian vector*

Quá trình xử lý truy vấn.

Đối với truy vấn, nhóm tiền xử lý giống như tài liệu gồm 3 bước đã trình bày ở Chương 2. Sau đó, nhóm sẽ tokenize truy vấn đã xử lý, với mỗi token:

- Nếu token không có trong tập term, bỏ qua token này.

- Ngược lại, truy xuất vào chỉ mục và tính toán.

Mã giả thuật toán:

VECTOR SPACE MODEL (*inverted_index*)

1. ∀ *term* **in** *inverted_index*:
2. Tính $idf(t,d) = \log \frac{N}{1 + \{d \in D, t \in D\}}$
3. ∀ *doc* **in** *inverted_index*: // $f_{t,d}$ = *inverted_index[term][doc][freq]*
4. Tính $tf(t,d) = \log(1 + f_{t,d})$
5. Tính $tf\text{-}idf(t,d) = tf(t,d) \cdot idf(t,d)$
6. *inverted_index*[*term*][*doc*][*weight*] = $tf\text{-}idf(t,d)$

Hình 4.2. *Mã giả thuật toán của mô hình không gian vector*

	experimental	investigation	of	the	aerodynamics	a	wing	in	slipstream	an	study	propeller	was	made	order
0	1.584963	1.0	3.459432	4.000000	1.0	6.066089	2.0	4.523562	2.584963	3.169925	1.000000	1.0	2.321928	1.584963	1.0
1	0.000000	0.0	2.807355	4.321928	0.0	6.339850	0.0	4.321928	0.000000	3.459432	1.584963	0.0	0.000000	0.000000	0.0
2	0.000000	0.0	0.000000	1.584963	0.0	3.807355	0.0	1.584963	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0
3	0.000000	0.0	2.321928	3.169925	0.0	5.209453	0.0	3.169925	0.000000	1.584963	0.000000	0.0	0.000000	1.000000	0.0
4	0.000000	0.0	1.000000	1.000000	0.0	5.044394	0.0	3.169925	0.000000	2.321928	0.000000	0.0	0.000000	0.000000	0.0

Hình 4.3. *Kết quả tính trọng số IDF*

experimental	2.133798
investigation	2.689660
of	0.005162
the	0.008268
aerodynamics	5.807355
...	...
thermometer	9.451211
fralich	9.451211
prevented	9.451211
ing	9.451211
ob	9.451211

Hình 4.4. *Kết quả tính trọng số TF*

4.2. Mô hình chỉ mục ngữ nghĩa tiềm ẩn

D: Ma trận term-doc

Q: Ma trận 1 chiều

F: Lí thuyết toán học về ma trận

R(D, Q): Sử dụng độ đo cosin.

Phân tích ngữ nghĩa tiềm ẩn là một thuật toán trích xuất và đại diện nội dung ngữ nghĩa sử dụng tính toán thống kê với một tập văn bản lớn (Landauer và Dumais, 1997). Ý tưởng cơ bản là lấy tập hợp các từ trong tập văn bản, đưa ra từ xuất hiện hoặc không xuất hiện trong các văn bản thuộc tập văn bản đó, sau đó tính toán sự tương đồng của các từ với các từ khác hoặc của tập từ với tập từ khác.

LSA giả định rằng những từ có ngữ nghĩa gần nhau thường xuất hiện trong cùng ngữ cảnh. Xuất phát từ bảng dữ liệu A kích thước m × n, mỗi hàng tượng trưng cho một ký tự, mỗi cột tượng trưng cho một đoạn văn bản, mỗi một ô chứa tần suất mà từ ở dòng ma trận xuất hiện trong đoạn văn bản được biểu diễn tại cột của ma trận. Sau đó, LSA sử dụng kỹ thuật phân tích giá trị đơn (Singular Value Decomposition - SVD) rút trích mối tương quan ngữ nghĩa giữa các từ trong tập văn bản, giảm số cột (chiều) về *k* đặc trưng tiềm ẩn của bảng dữ liệu, thu được bảng R kích thước m × k, trong khi vẫn giữ được cấu trúc tương tự của các dòng trong bảng R

Ý tưởng chính.

Cho ma trận tài liệu A có kích thước $m \times n$. Ma trận A có thể được phân tích thành tích của ba ma trận theo dạng:

$$A = USV^T$$

Trong đó:

- U là ma trận trực giao có kích thước $m \times m$, các cột là các vector đơn bên trái của A.
- S là ma trận có kích thước $m \times n$, đường chéo chứa các giá trị đơn, không âm có thứ tự giảm dần $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_{min(m,n)} \geq 0$.
- V là ma trận trực giao có kích thước $n \times n$, các cột là các vector đơn bên phải của A.

Trong kĩ thuật LSA, ma trận A thường là ma trận thưa có kích thước rất lớn. Để giảm số chiều của ma trận, người ta thường tìm cách xấp xỉ ma trận A (có hạng *r*) bằng một ma trận A_k có hạng $k < r$. Tham số *k* này có thể được xác định thông qua trọng số Silhouette. Tuy nhiên, trong đồ án này, nhóm không có thực hiện bước giảm chiều của ma trận.

Về cách xếp hạng tài liệu thì nhóm sử dụng tương tự cách xếp hạng tài liệu ở mục 4.1.

Ví dụ. Có 9 tập tài liệu sau:

c1: Human machine interface for ABC computer applications

c2: A survey of user opinion of computer system response time

c3: The EPS user interface management system

c4: System and human system engineering testing of EPS

c5: Relation of user perceived response time to error measurement

m1: The generation of random, binary, ordered trees

m2: The intersection graph of paths in trees

m3: Graph minors IV: Widths of trees and well-quasi-ordering

m4: Graph minors: A survey

Ta có ma trận tài liệu của 9 tài liệu trên như sau:

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Ta tính toán được ma trận U:

■ {U} =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

Ma trận S:

■ {Σ} =

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

Ma trận V:

■ {V} =

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Ta tiến hành tính toán lại A_2 với tham số $k = 2$:

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Qua ví dụ trên, thấy rằng tại ma trận được xây dựng lại mối quan hệ giữa các từ trong câu đã thay đổi do việc tính toán liên quan đến các từ trong tập văn bản. Ví dụ từ “survey” với câu m4 ban đầu là 1, qua biến đổi giá trị 0.42; từ “trees” với câu m4 ban đầu là 0, qua biến đổi giá trị 0.66. Sự thay đổi này được chứng minh là chính xác hơn so với mối quan hệ ban đầu.

CHƯƠNG 5. THƯ VIỆN WHOOSH

5.1. Giới thiệu về Whoosh

Whoosh được tạo ra bởi Matt Chaput. Whoosh nhanh, nhưng chỉ sử dụng Python thuần túy, vì vậy nó sẽ chạy ở bất cứ nơi nào Python chạy mà không yêu cầu trình biên dịch.
Theo mặc định, Whoosh sử dụng xếp hạng tài liệu BM25F, nhưng giống như hầu hết các thư viện khác, ta có thể dễ dàng tùy chỉnh thông số xếp hạng tài liệu.
Whoosh tạo các số lượng chỉ mục ít hơn so với nhiều thư viện tìm kiếm khác.
Whoosh cho phép lưu trữ các đối tượng Python tùy ý với các tài liệu được lập chỉ mục.

5.2. Các tiến trình hoạt động của Whoosh

5.2.1. Xây dựng tập chỉ mục tìm kiếm

Cách mô hình hóa nội dung văn bản với Whoosh

- ***Các thủ tục cơ bản trên tập chỉ mục.*** Thêm tài liệu vào tập chỉ mục, xóa tài liệu từ tập chỉ mục, cập nhập tài liệu trong tập chỉ mục.
- ***Các thủ tục cơ bản trên tập chỉ mục.*** Có nhiều phương pháp để biểu diễn tài liệu và một phương pháp tự động cơ bản được thực hiện theo các bước:
  - Bước 1: Tách từ - Tokenization.
  - Bước 2: Loại bỏ từ thông dụng – stop word.
  - Bước 3: Quy về từ gốc – stemming
  - Bước 4: Đánh trọng số cho từ chỉ mục – term weighting.
- ***Các tuỳ chọn cho Field.*** Lựa chọn cho chỉ mục, các tùy chọn cho lưu trữ, tùy chọn cho Vector mục từ, sự kết hợp các tùy chọn.
- ***Thiết lập mức độ quan trọng cho các tài liệu và các trường.*** Không phải tất cả các tài liệu và các trường được tạo ra đều có mức độ quan trọng như nhau. Nâng cao mức độ quan trọng cho chúng có thể được thực hiện trong suốt tiến trình lập chỉ mục hoặc tiến trình tìm kiếm.
- ***Cắt giảm trường thông tin (Field truncation).*** Khi lập chỉ mục cho các tài liệu có kích thước chưa xác định, để đảm bảo cung cấp đủ bộ nhớ RAM cũng như dung lượng đĩa cứng cần thiết, thư viện Whoosh cho phép giới hạn số mục từ cần được lập chỉ mục trên các trường tương ứng.

- ***Tối ưu hóa tập chỉ mục.*** Khi tạo chỉ mục cho nhiều tài liệu, sẽ có rất nhiều segment được tạo ra. Lúc tìm kiếm hệ thống sẽ thực hiện tìm qua tất cả các segment này và kết hợp kết quả lại. Để giảm thiểu số lượng segment và tiết kiệm bộ nhớ, Whoosh hỗ trợ phương thức cho phép hợp các segment lại chỉ còn một vài segment, vừa đơn giản, tiết kiệm bộ nhớ vừa tìm kiếm nhanh hơn.

5.2.2. Tìm kiếm trên tập chỉ mục

Mục đích của hệ thống tìm kiếm là phải trả về kết quả chính xác trong thời gian nhanh nhất. Khả năng của Whoosh có thể trả về hàng trăm, hàng nghìn và có thể hàng triệu tài liệu liên quan chỉ trong thời gian ngắn với các phương thức đơn giản. Một số lớp chính sử dụng cho tìm kiếm bao gồm: IndexSearcher, Query, QueryParse.

Phân tích truy vấn: QueryParse

QueryParser của Whoosh là lớp đối tượng chính được sử dụng để tạo ra câu truy vấn từ nội dung tìm kiếm của người sử dụng.

Tìm kiếm với đối tượng IndexSearcher

Toàn bộ quá trình tìm kiếm được thực hiện dựa trên một số lớp đối tượng chính như IndexSearcher, IndexReader …

Danh sách kết quả

Khi gọi tìm kiếm với phương thức `search (query, n)` sẽ nhận được kết quả trả về, đối tượng này chứa danh sách *n* các mã tài liệu được xếp thứ tự theo mức độ liên quan.

5.2.3. Tiến trình phân tích của Whoosh

Các bước Whoosh thường sử dụng để phân tích như:

Sử dụng trình phân tích: dùng khi chuyển văn bản sang mục từ.

Từ vựng - Token: Analyzer là lớp cơ sở, nó thực hiện chuyển dữ liệu văn bản sang một dòng các từ vựng (Token) trong lớp TokenStream.

Loại bỏ các từ thường dùng - Stopword: StopAnalyzer thực hiện loại bỏ một số từ đặc biệt gọi là các stop word.

Chuẩn hoá từ - Stemming: Tiến trình xử lý cuối cùng là loại bỏ các hình thái khác nhau của từ để đưa nó về dạng gốc.

Phân tích ngôn ngữ: Chúng ta phải giải quyết nhiều vấn đề khi phân tích các văn bản với những ngôn ngữ khác nhau

CHƯƠNG 6. KẾT QUẢ THỬ NGHIỆM

Nhóm sẽ sử dụng độ chính xác trung bình, độ phủ trung bình, độ đo F trung bình để thực hiện đánh giá kết quả truy xuất thông tin với hai mô hình VSM, LSI và khi sử dụng thư viện Whoosh (trọng số sử dụng là trọng số TF-IDF).

Bảng 6.1. *Kết quả thử nghiệm*

	Vector Space Model	**Latent Semantic Indexing**	**Whoosh**
Precision	**0.16**	0.09	0.01
Recall	**0.51**	0.22	0.37
F1-score	**0.24**	0.12	0.01

Nếu thuật toán VSM thực hiện nhanh không tốn nhiều thời gian để thực thi thì thuật toán LSI lại mất rất nhiều thời gian để thực hiện truy xuất văn bản nhưng độ chính xác vẫn chưa được cải thiện hơn so với VSM.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Đỗ Đăng Hưng, "Tiền xử lí dữ liệu văn bản với NLTK", 2021. [Trực tuyến]. Địa chỉ: http://it.ou.edu.vn/asset/ckfinder/userfiles/5/files/TrichDanKieu_IEEE.pdf [Truy cập 24/02/2023]

[2] Trần Xuân Chiến, "NLP – Stemming và Lemmatization", 2016. [Trực tuyến]. Địa chỉ: https://chienuit.wordpress.com/2016/05/03/nlp-stemming-va-lemmatization/ [Truy cập 24/02/2023]

[3] "Tìm hiểu về mô hình không gian vector", 2013. [Trực tuyến]. Địa chỉ: https://butchiso.com/2013/10/tim-hieu-ve-mo-hinh-khong-gian-vector.html [Truy cập 26/02/2023]

[4] Phương pháp tìm kiếm Boolean. [PowerPoint]. Viện công nghệ Thông tin và Truyền thông. Địa chỉ: https://users.soict.hust.edu.vn/hieunk/courses/IT4853/L01.pdf [Truy cập 26/02/2023]

[5] "4. Term Weighting and Vector Space Model". [PowerPoint]. Universität Mannheim

[6] "8. Latent and Semantic Retrieval" . [PowerPoint]. Universität Mannheim

[7] Trần Cao Đệ. "Đo độ tương tự ngữ nghĩa tiềm ẩn để phát hiện việc sao chép tài liệu". *Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông.*

[8] Trần Thanh Tùng. "Tự động phân tích các nội dung giống nhau trong hệ thống tổng hợp ý kiến góp ý trong hội nghị". Luận văn thạc sĩ. Trường Đại học Công Nghệ. Đại học Quốc gia Hà Nội

TỈ LỆ ĐÓNG GÓP BÁO CÁO

<u>Phan Thanh Hải = Nguyễn Hoàng Long = 50%</u>, trong đó:

Phan Thanh Hải code phần tiền xử lý dữ liệu + xác định term của tài liệu; soạn thảo nội dung báo cáo từ chương 1 đến chương 3.

Nguyễn Hoàng Long code phần chạy mô hình và đánh giá kết quả; soạn thảo nội dung báo cáo từ chương 4 trở đi.