ĐẠI HỌC QUỐC GIA TP.HCM

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

----------o0o----------

ĐỒ ÁN MÔN HỌC:

XÂY DỰNG MÔ HÌNH TRUY VẤN KHÔNG GIAN VECTOR

Giảng viên hướng dẫn: Nguyễn Trọng Chỉnh

Sinh viên thực hiện:

Hà Thúc Đăng Khoa (18520914)

Dương Ngọc Hùng (18520792)

Nguyễn Tuấn Lộc (18521011)

THÀNH PHỐ HỒ CHÍ MINH - 2021

Mục lục

Danh sách hình minh hoạ 3
1. GIỚI THIỆU 4
1.1. Tổng quan 4
1.2. Nhiệm vụ đề tài 4
2. CƠ SỞ LÝ THUYẾT 4
2.1. Tổng quan về truy vấn thông tin 4
2.2. Hệ thông truy vấn thông tin theo mô hình không gian vector 6
2.2.1. Khái quan về mô hình không gian vector 6
2.2.2. Term và tiền xử lí 7
2.2.3. Stemming và lemmatizing 8
2.2.4. Inverted index 10
2.2.5. Trọng số từ 13
2.2.6. Thực hiện truy vấn 14
2.2.7. Mức độ tương đồng 15
2.2.8. Đánh giá một hệ thống truy vấn 16
3. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM 17
4. KẾT QUẢ THỰC NGHIỆM 21
5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 23
1. Kết luận 23
2. Hướng phát triển 23
6. TÀI LIỆU THAM KHẢO 23
7. PHỤ LỤC 24
1. Tập stop word sử dụng trong quá trình thực nghiệm 24
2. Các nhãn POS phổ biến 24
3. Ảnh kết quả thực hiện đánh giá 26

Danh sách hình minh hoạ

1. Khái quát quy trình truy vấn thông tin 5
2. Sơ đồ các mô hình đánh giá mức độ liên quan 6
3. Ví dụ stemming. 8
4. Một công thức tf và idf 13
5. Minh hoạ cosine giữa 2 vector 15
6. Biểu đồ biểu diễn recall và precision 17

1. GIỚI THIỆU

1.1. Tổng quan

Truy vấn thông tin là một phần quan trọng trong thời đại số hiện nay. Sự xuất hiện của internet dẫn đến việc bùng nổ thông tin. Lượng thông tin lớn và không ngừng tăng lên đòi hỏi cần phải có phương pháp để có thể truy xuất được thông tin cần thiết trong vô số thông tin đang rải rác trên mạng.

1.2. Nhiệm vụ đề tài

Vận dụng kiến thức đã học, xây dựng hệ thống truy vấn thông tin sử dụng mô hình không gian vector có các chức năng:

- Lập chỉ mục
- Xử lí văn bản
- Thực hiện truy vấn

Lập chỉ mục trên tập dữ liệu Cranfield, đánh giá kết quả và so sánh dựa trên tập test gồm 100 câu truy vấn và chuẩn vàng kèm theo mỗi truy vấn.

2. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về truy vấn thông tin

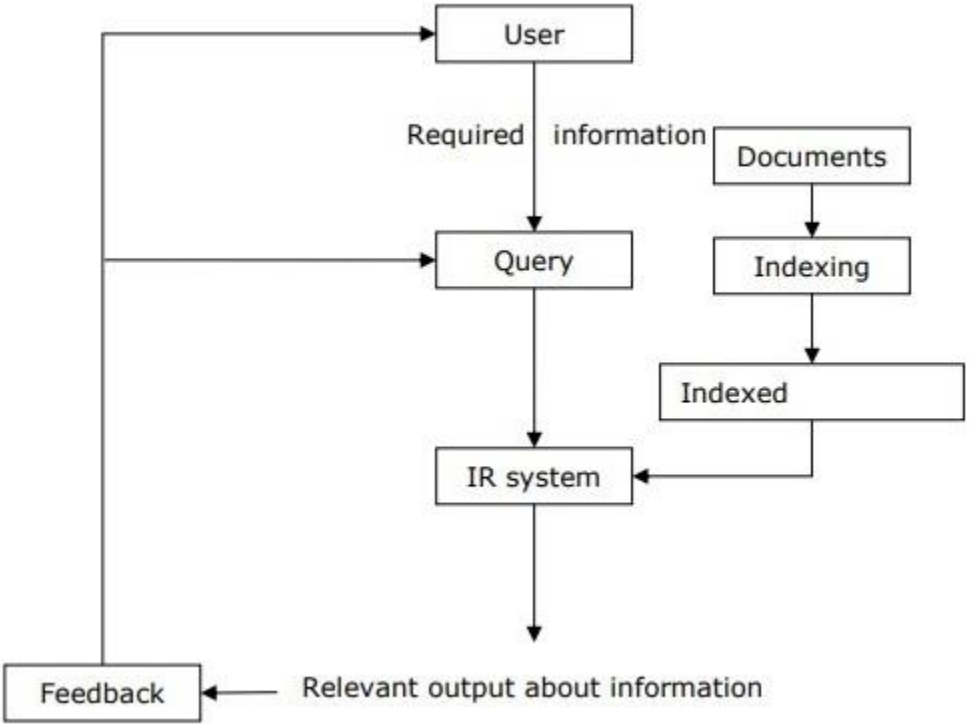
Truy vấn thông tin là quá trình thu thập tài nguyên của một hệ thống thông tin mà có liên quan đến thông tin cần tìm kiếm[4]. Việc thu thập thông tin này có thể được tự động hoá bằng các sử dụng một hệ thống truy vấn thông tin. Sử dụng các hệ thống truy vấn thông tin tự động còn nhằm hạn chế tình trạng quá tải thông tin[1], đặc biệt trong thời đại công nghệ thông tin hiện nay, khi mà có nhiều người truy cập hơn bao giờ hết.

Để giải quyết tình trạng quá tải thông tin, các hệ thống truy vấn thông tin được sử dụng để hỗ trợ việc truy xuất thông tin cần thiết. Một hệ thống truy vấn thông tin là một

[1] Quá tải thông tin (hoặc bùng nổ thông tin) là tình trạng tồn tại quá nhiều thông tin về một chủ đề, dẫn đến việc khó tiếp thu và đưa ra những quyết định về chủ đề đó.

chương trình cung cấp truy cập đến sách, báo và các tài liệu khác; lưu trữ và quản lý những tài liệu này[4].

Quy trình truy vấn thông tin có thể được thể hiện qua biểu đồ sau:



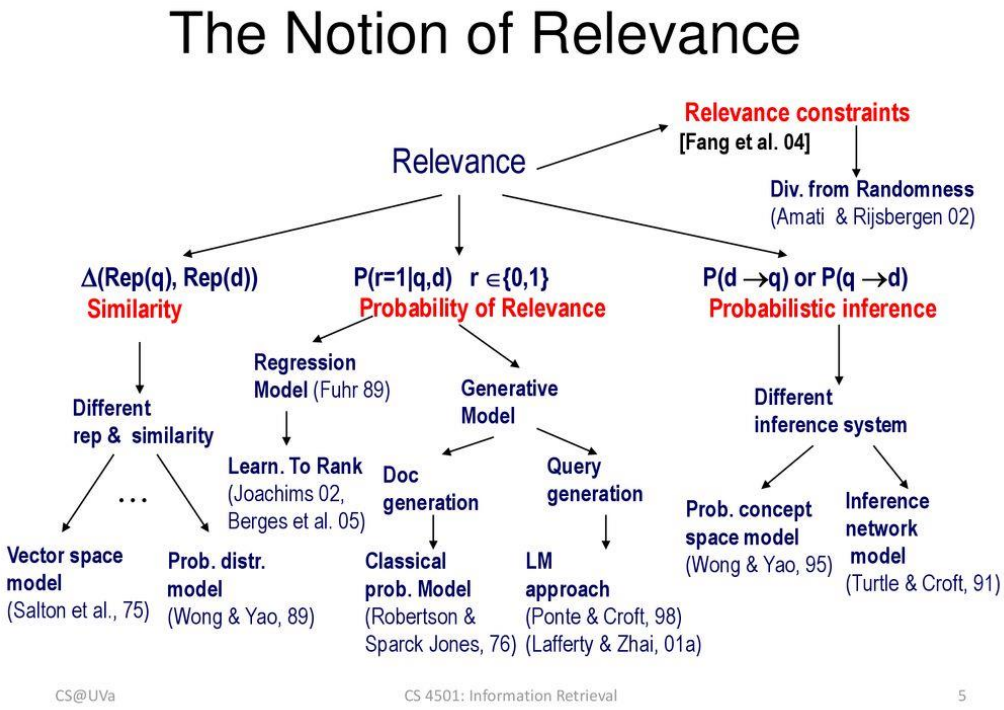
1. Khái quát quy trình truy vấn thông tin

Theo biểu đồ trên, hệ thống truy vấn sẽ nhận thông tin dưới dạng chỉ mục (index). Khi người dùng nhập truy vấn , hệ thống sẽ trả về kết quả phù hợp với câu truy vấn. Kết quả trả về này sẽ được đánh giá và phản hồi về phía người dùng.

Đánh giá mức độ phù hợp của một văn bản đối với câu truy vấn là một vấn đề nan giải do hai nguyên do chính:

- Ngôn ngữ con người phong phú, đa dạng, dẫn đến việc nhiều câu truy vấn có thể đề cập đến cùng một văn bản.
- Không phải câu truy vấn nào cũng đủ tốt để biểu diễn văn bản nó muốn đề cập đến.

Từ đây, ta nhận thấy cần có cách để biểu diễn mức độ phù hợp giữa câu truy vấn và văn bản trong tập tài liệu. Trong quá khứ, đã có nhiều nghiên cứu về vấn đề này, nhiều giải pháp đã được đề xuất; tuy nhiên, những đề xuất này cũng chi mang tính heuristic.



2. Sơ đồ các mô hình đánh giá mức độ liên quan

Một trong những các này chính là đánh giá dựa trên mức độ tương đồng, mà cụ thể là mô hình không gian vector. Ta sẽ bàn cụ thể hơn về mô hình này trong phần sau của bài báo cáo.

2.2. Hệ thông truy vấn thông tin theo mô hình không gian vector

2.2.1. Khái quan về mô hình không gian vector

Mô hình không gian vector là một mô hình toán học được sử dụng trong việc biểu diễn văn bản chữ bằng một vector định danh[7]. Dựa vào điểm này, ta có thể thấy

mô hình không gian vector phù hợp với bài toán truy vấn thông tin, mà nó được sử dụng lần đầu tiên trong hệ thống truy vấn thông tin SMART[2].

Trong một mô hình không gian vector, văn bản thứ *i* có độ dài *n* từ và câu truy vấn có độ dài *t* từ được biểu diễn bằng các vector định danh tương ứng:

- $doc_i = (w_{1i}, w_{2i}, \dots, w_{ni})$
- $q = (w_1, w_2\ , \dots, w_t)$

Mỗi chiều trong vector định danh của một văn bản là tương ứng với một term, và giá trị của nó sẽ khác không nếu term này tồn tại trong văn bản đó. Có nhiều cách để tính những giá trị đã được phát triển, và giá trị được gọi là trọng số của term. Ta sẽ làm rõ thêm về vector định danh và trọng số của term trong phần 2.2.5.

2.2.2. Term và tiền xử lí

Định nghĩa của term sẽ phụ thuộc vào ứng dụng của mô hình. Thông thường, term sẽ là từng từ riêng lẻ, các từ khoá hoặc các cụm từ. Quy trình tiền xử lí là bước đầu tiên của quy trình lập chỉ mục cũng như thực hiện truy vấn trên hệ thống. Quy trình này bảo gồm các bước:

1. **Loại bỏ các kí tự không mong muốn:** Các kí tự không mong muốn này là các dấu câu,các kí tự đặc biệt không mong muốn. Nếu không loại các kí tự này sẽ làm tăng kích thước tập token không cần thiết.
2. **Tách văn bản thành các token:** Các token này có thể từng từ trong câu, hoặc một cụm n từ
3. **Stemming/Lemmatizing các token về dạng "gốc" của nó:** Phần này khác rộng, sẽ được làm rõ hơn ở phần sau của bài báo cáo.

[2] SMART(System for the Mechanical Analysis and Retrieval of Text): hệ thống truy vấn thông tin được phát triển bởi đại học Cornell, nhiều khái niệm quan trọng trong lĩnh vực truy vấn thông tin được phát triển trong quá trình nghiên cứu hệ thống này.

4. **Loại bỏ các token nằm trong tập stop word:** Stop word là những từ xuất hiện quá thường xuyên trong tập văn bản dẫn đến việc chúng mang lại ít giá trị trong việc phân biệt giữa các văn bản.

Sau khi được tiền xử lí, ta sẽ có được một tập các term sẵn sàng cho quá trình lập chỉ mục hoặc truy vấn.

2.2.3. Stemming và lemmatizing

Đối với một số ngôn ngữ, cụ thể là tiếng anh, một từ có thể mang nhiều dạng cho phù hợp về mặt ngữ pháp. Chẳng hạn:

- talk – talks – talking – talked
- take – took – taken
- go – going – went – gone

Ngoài ra, ngôn ngữ còn tồn tại nhiều nhóm các từ có liên quan với nhau mang nét tương đồng về mặt ngữ nghĩa. Ví dụ:

- democracy – democratic – democratization
- illiterate – illiteracy

Trong nhiều trường hợp, ta cần đưa những từ này về gốc từ của nó để có thể truy vấn được các tài liệu phù hợp. Để làm điều này, ta có hai hướng tiếp cận là stemming và lemmatize.

Stemming là phương pháp mang tính heuristic. Những gì mà stemming làm đơn thuần là cắt bỏ đuôi hoặc biến đổi của từ theo một số nguyên tắc nhất định với hi vọng đem lại kết quả phù hợp.

(F)	**Rule**		
	SSES	→	SS
	IES	→	I
	SS	→	SS
	S	→	

3. Ví dụ stemming.

Dựa theo các nguyên tắc trong hình 3, ta có một số ví dụ về việc stemming từ như sau

- caresses -> caress
- quantities -> quantiti
- success -> sucess
- talks -> talk

Không tồn tại một phương pháp stemming hoàn hảo cho mọi văn bản. Tuy nhiên, đã có các nguyên cứu được đề xuất một số phương pháp stemming bao gồm Lovins Stemmer, Porter Stemmer và Paice Stemmer. Trong số các phương pháp này, Porter Stemmer, trải qua quá trình đánh giá và thực nghiệm, cho kết quả tốt giữa ba phương pháp nêu trên[3].

Mặt khác, một hướng khác trong bài toán đưa từ về gốc từ chính là lemmatize. Thay vì sử dụng một bộ quy tắc được quy định từ trước, các lemmatizer sẽ phân tích từ để biến đổi từ về gốc từ một các chính xác, gốc từ này được gọi là một lemma. Để làm được điều này, các lemmatizer phải làm một bước gọi là phân tích hình thái từ.Phân tích hình thái từ là việc phân loại, phân tích và mô tả cấu trúc của các morpheme, ví dụ như từ gốc, tiền tố, hậu tố và part-of-speech(tạm dịch: vị trí trong câu văn). WordNet Lemmatizer là một lemmatizer phổ biến, sử dụng part-of-speech cửa từ để tìm ra lemma phù hợp.

Một số ví dụ của lemmatizer sử dụng một số part-of-speech phổ biến có thể được tham khảo ở phụ lục 3 của bài báo cáo:

- loving/NN-> loving
- loving/VBG -> love
- is/VBP -> be

Trong ví dụ trên, ta thấy được cùng một từ “loving” nhưng với part-of-speech khác nhau, ta có được hai lemma khác nhau.

2.2.4. Inverted index

Quay lại vấn đề ban đầu, sau khi tiền xử lí văn bản, ta vẫn cần cách để biểu diễn chúng sao cho thuận tiện cho quá trình truy vấn. Giả sử trong corpus của chúng ta có ba văn bản sau:

- Doc_1: Computer architecture is hard.
- Doc_2: It's a hard knock life.
- Doc_3: The computer is broken.

Từ ba văn bản này,sau khi trải qua các bước tiền xử lí, ngoại trừ bước stem /lemmatize và loại bỏ stop word, ta có thể lập được một ma trận I hai chiều như sau:

	Doc_1	Doc_2	Doc_3
a	0	1	0
architecture	1	0	0
broken	0	0	1
computer	1	0	1
hard	1	1	0
is	1	0	1
it's	0	1	0
knock	0	1	0
life	0	1	0
the	0	0	1

Trong ma trận nêu trên, mỗi cột đại diện cho một văn bản và mỗi hàng là một từ. Ô I_{ij} bằng 1 nếu từ thứ i tồn tại trong văn bản thứ j và bằng 0 nếu ngược lại. Như vậy, khi truy vấn ta có thể sử dụng mỗi cột như vector đặc trưng của mỗi văn bản. Truy vấn theo cách này được gọi là truy vấn theo mô hình boolean. Hạn chế của mô hình này chính là sự lãng phí bộ nhớ. Chẳng hạn ta có 3 văn bản, mỗi văn bản dài 1000 từ; tuy nhiên, ba văn bản này lại có có từ chung, tức là từ tồn tại trong văn bản thứ nhất sẽ không tồn tại trong hai văn bản còn lại. Điều này dẫn đến ma trận I có kích

thước 3000x3 sẽ có đến 6000 ô mang giá trị 0 và chỉ có 1/3 ma trận thực sự mang giá trị thông tin. Khi này, ma trận I sẽ được gọi là ma trận thưa. Để giải quyết tình trạng này, ta sẽ sử dụng một cấu trúc dữ liệu gọi là inverted posting list(IPL).

Ý tưởng của cấu trúc dữ liệu này chính là map các term với danh sách sách các văn bản chứa nó, được gọi là posting list

$term_i$: doc_1, doc_2,…,doc_n

Trong IPL, các văn bản được thể hiện bằng id thay vì tên đầy đủ. Trong trường hợp này, ta sử dụng ba văn bản doc_1, doc_2, và doc_3 để làm ví dụ và chúng có id là 1,2 và 3 tương ứng. Việc lập một IPL trải qua các bước như sau:

1. Lập từ điển các từ và ghi nhận văn bản chứa nó

Term	Id
computer	1
architecture	1
is	1
hard	1
it’s	2
a	2
hard	2
knock	2
life	2
the	3
computer	3
is	3
broken	3

2. Sắp xếp các từ theo thứ tự bảng chữ cái

Term	Id

a	2
architecture	1
broken	3
computer	1
computer	3
hard	1
hard	2
is	1
is	2
it’s	2
knock	2
life	2
the	3

3. Hợp nhất các từ trùng lặp tạo thành posting list

Term	Posting list
a	2
architecture	1
broken	3
computer	1, 3
hard	1, 2
is	1, 3
it’s	2
knock	2
life	2
the	3

Như vậy, ta đã có được một IPL cho tập văn bản gồm ba văn bản doc_1, doc_2, và doc_3. Quy trình xây dựng IPL cho mọi corpus là như nhau.

2.2.5. Trọng số từ

Sau khi có được IPL, ta tiếp đến bước kế tiếp chính là đánh trọng số cho term trong văn bản. Như đã nói ở phần 2.2.1, một văn bản được biểu diễn một vector định danh, có dạng $doc_i = (w_{1i}, w_{2i}, \dots, w_{ki}, \dots, w_{ni})$, trong đó:

- doc_i là văn bản thứ i trong corpus, có độ dài tương ứng với n term.
- w_{ki} là trọng số của term thứ k trong văn bản thứ i.

Trọng số w_{ki} được được tính bằng phương pháp heuristic tf-idf, với tf là từ viết tắ cho term frequency và idf là inverse document frequency. Một số hệ thống còn trải qua bước chuẩn hoá trọng số này để hỗ trợ cho quá trình truy vấn. Tuy nhiên, hệ thống mà nhóm xây dựng sẽ không có bước chuẩn hoá trọng số. Để tính tf-idf của một từ, ta đơn thuần chỉ cần nhân tf với idf. Về phần tf và idf, 2 khái niệm này có khá nhiều cách tính khác nhau:

Term frequency		Document frequency	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$		

4. Một công thức tf và idf

Xét các công thức trong ảnh 4, ta có:

- $\mathrm{tf}_{t,d}$ là số lần xuất hiện của term t trong văn bản d.
- df_t là số văn bản có chứa term t.
- N là số văn bản trong corpus.

Hệ thống mà nhóm xây dựng sử dụng công thức $l * t = \left(1 + \log\left(tf_{t,d}\right)\right) * \left(\log\left(\frac{N}{df_t}\right)\right)$ cho cả văn bản trong corpus lẫn câu truy vấn. . Tuy nhiên, trong ví dụ, bài báo cáo sẽ chỉ sử dụng công thức $tf * idf = tf_{t,d} * \frac{N}{df_t}$ cho ví dụ. Sau khi tính trong số các từ trong văn bản tương ứng, ta sẽ ghi nhận trọng số vào posting list.

Term	Posting list	idf
a	(2, 3)	3
architecture	(1, 3)	3
broken	(3, 3)	3
computer	(1,3/2), (3,3/2)	3/2
hard	(1,3/2), (2,3/2)	3/2
is	(1,3/2), (3,3/2)	3/2
it's	(2,3)	3
knock	(2,3)	3
life	(2,3)	3
the	(3,3)	3

Lưu ý, ta cần lưu lại giá trị idf của mỗi term để có thể tính trọng số của term trong câu truy vấn. Một số posting list sẽ được sắp xếp theo thứ tự tăng dần của giá trị tf-idf, tuy nhiên, do hạn chế của cấu trúc dữ liệu sử dụng trong quá trình xây dựng hệ thống, nhóm sẽ bỏ qua bước này.

2.2.6. Thực hiện truy vấn

Hệ thống thực hiện truy vấn từ một chuỗi kí tự theo các bước sau:

1. Tiền xử câu truy vấn tương tự như một văn bản trong tập corpus và lấy tập các term riêng biệt T.

2. Tham chiếu đến IPL để lấy được các tập văn bản D có chứa các term thuộc câu truy vấn. Nếu tồn tại term t_i không tồn tại trong IPL thì term t_i này sẽ bị loại khỏi T.

3. Xây dựng vector định danh cho mỗi văn bản trong tập D cũng như câu truy vấn.

3.1 Đối với mỗi văn bản d_i trong tập D, trọng số của term sẽ bằng 0 nếu term không tồn tại trong d_i và ngược lại, sẽ bằng giá trị tf-idf được lưu trong IPL.

3.1 Đối với câu truy vấn q, các term sẽ được tính giá trị tf và nhận với giá trị idf được lưu trong IPL. Nếu term không tồn tại trong IPL, trọng số sẽ bằng 0.

4. Sắp xếp các vector định danh của mỗi văn bản d_i trong tập D theo mức độ tương đồng giảm dần đối với vector định danh của câu truy vấn.

2.2.7. Mức độ tương đồng

Một bước trong phần thực hiện truy vấn đó chính là đánh giá mức độ tương đồng giữa vector định danh của mỗi văn bản và vector định danh của câu truy vấn. Để giải quyết vấn đề này, ta sử dụng độ đo cosine similarity.

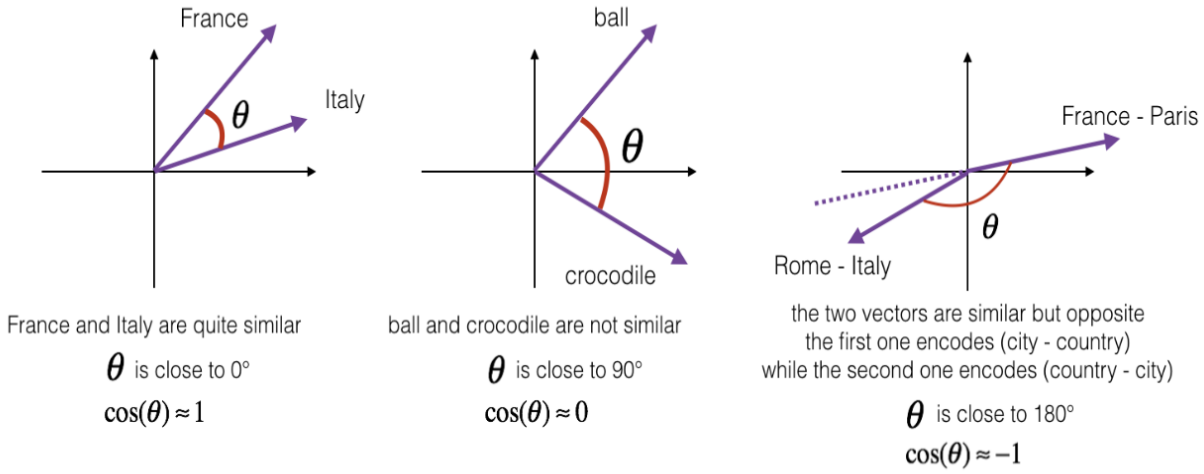


Figure 1: The cosine of the angle between two vectors is a measure of how similar they are

5. Minh hoạ cosine giữa 2 vector

Ý tưởng của độ đo này chính là so sánh góc giữa vector định danh của mỗi văn bản với vector định danh của câu truy vấn. Mức độ tương đồng giữa 2 vector càng cao thì góc càng nhỏ. Dẫn đến cosine của góc giữa hai vector sẽ lớn. Độ đo cosine similarity của hai vector được tính bằng công thức sau:

$$\cos(d_i, q) = \frac{d_i * q}{||d_i||_2 * ||q||_2}$$

Trong đó:

d_i là vector định danh của tài liệu thứ i trong corpus.

q là vector định danh của câu truy vấn.

2.2.8. Đánh giá một hệ thống truy vấn

Để đánh giá một hệ thống truy vấn, ta có một số độ đo cơ bản như sau:

- Presision(Độ chính xác) là tỉ lệ tài liệu có liên quan trên tổng số tài liệu được truy vấn và được tính bằng công thức:

$$precision = \frac{\#\ relevent\ document\ retrieved}{\#\ retrieved\ document}$$

- Recall(Độ phủ) là tỉ lệ tài liệu có liên quan được truy vấn trên tổng số tài liệu có liên quan trong corpus. Recall được tính bằng công thức:

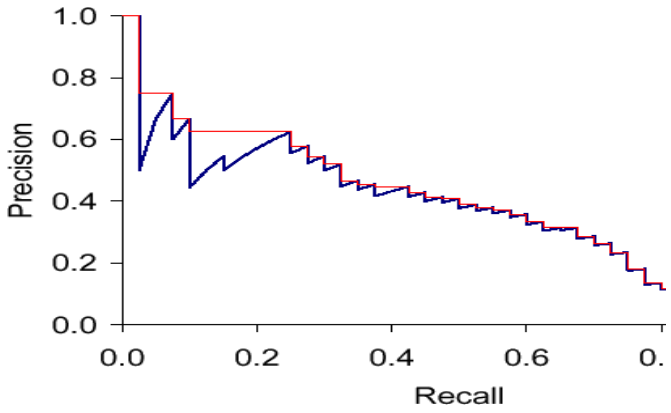
$$recall = \frac{\#\ relevent\ document\ retrieved}{\#\ total\ relevent\ document\ in\ corpus}$$

- F-measure là trung bình điều hoà của precision và recall, được tính bằng công thức:

$$F - measure = 2 * \frac{precision * recall}{precision + recall}$$

Tuy nhiên, các độ đo nêu trên chỉ phù hợp với tập văn bản không có thứ tự. Cần phải có sự điều chỉnh để có thể sử dụng để đánh giá mô hình truy vấn vì kết quả có thứ tự xếp hạng.

Ta có thể xem k văn bản đầu tiên trong tập văn bản được truy vấn là một tập văn bản không có thứ tự. Với mỗi tập văn bản như vậy, ta có thể tính được precision và recall dựa trên công thức được nêu trước đây và biểu diễn được dưới dạng đồ thị trong biểu đồ dưới.



6. Biểu đồ biểu diễn recall và precision

Đồ thị của precision và recall được tính như trên có dạng răng cưa đặc trưng. Có thể loại bỏ được dạng răng cưa này bằng cách sử dụng interpolated precision(độ chính xác nội suy). Độ chính xác nội suy có thể được tính sử dụng công thức sau:

$p_{interp}(r) = \max p(r')_{r' \geq r}$

Trong đó:

- $p_{interp}(r)$: độ chính xác nội suy tại recall rank r
- $\max p(r')$: độ chính xác lớn nhất xét giữa các recall rank r' với $r' \geq r$

Sử dụng công thức trên, ta tính độ chính xác nội suy cho 11 điểm recall 0.0, 0.1, 0.2, … 1.0. Độ chính xác nội suy được biểu diễn bằng đường bằng đỏ trong biểu đồ 1.

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

Mã nguồn: https://github.com/jonathanha52/VectorSpace-IR-Model/tree/replaced

Quy trình xây dựng hệ thống truy vấn của nhóm được thực hiện trên ngôn ngữ Python 3.8 do tính đơn giản về mặt cú pháp và tính quen thuộc từ những lần thực hiện đồ án trước đây. Kèm theo đó, nhóm sử dụng một số thư viện hỗ trợ của Python, bao gồm:

- json: Thư viện json được sử dụng để đọc, ghi file chỉ mục này.

- math: Trong quá trình lập chỉ mục, các hàm toán học được sử dụng thường xuyên trong quá trình đánh trong số các từ. Việc sử dụng các hàm có sẵn của thư viện math giúp giảm tải công việc.
- nltk 3.5: thư viện cung cấp các lớp và hàm ngôn ngữ tự nhiên, tham gia vào quá trình xử lí văn bản như tách từ, chuẩn hoá từ.
- os: Quá trình lập chỉ mục và truy vấn cần đọc đường dẫn đến vị trí lưu tập dữ liệu được lưu. Thư viện os là giúp việc đọc đường dẫn thực hiện được trên đa nền tảng.
- numpy: Mỗi văn bản và câu truy vấn có thể được biểu diễn bằng 1 vector đặc trưng. Sử dụng thư viện numpy để có thể thao tác trên vector này và đo mức độ tương đồng giữa câu truy vấn và văn bản.
- time: sử dụng trong quá trình đánh giá hiệu năng của mô hình bao gồm đo thời gian lập chỉ mục và thời gian thực hiện truy vấn.

Hệ thống được xây dựng theo hướng đối tượng bao gồm các class như sau:

1. **CorpusPreprocess**:

CorpusPreproces
+path: string +id: hashmap +processed: hashmap +listfile: array
+preprocess(path:string, lemma: boolean, stem: boolean, st: boolean) - idmapping() - process(lemma: boolean, stem: boolean, st: boolean)

Class này chịu trách nhiệm gán id và tiền xử lí từng văn bản trong corpus. Các văn bản trong corpus đầu tiên sẽ được gán id bằng hàm **idmapping** và lưu vào hashmap ***id***, sau đó mỗi văn bản trong corpus được tiền xử theo các bước ở phần 2.2.2 trong hàm **process**. Hai

hàm này sẽ được gọi thông qua giao diện là hàm **preprocess**. Sau khi tiền xử lí, nội dung đã được tiền xử lí sẽ được map với id của văn bản tương ứng trong hashmap ***processed***. Cuối cùng, hashmap *id* và *processed* sẽ được lưu vào hai file json có tên *id.json* và *processed.json* cùng vị trí với corpus để chuẩn bị cho bước lập chỉ mục.File id.json và process.json có cấu trúc tương ứng như sau:

id.json

{
<id> : <doc_i>,
<doc_i> : < id>
}

processed.json

{
<id> : [<$term_1$>,
<$term_2$>,…,<$term_n$>
}

2. CorpusIndexing

CorpusIndexing
+doc: hashmap +index: hashmap +doccount: int
+indexing(path:string)

Class CorpusIndexing thực hiện xây dựng IPL theo các bước ở phần 2.2.4. CorpusIndexing sẽ nhận tham số là *path* là đường dẫn đến vị trí corpus. Tại đây, class sẽ tìm 2 file id.json và process.json để thực hiện lập chỉ mục. IPL của corpus sẽ được lưu vào biến *index*, sau đó, thông qua thư viện json sẽ được lưu vào file *index.json* cùng vị trí với corpus. File *index.json* có cấu trúc như sau:

index.json

{
<term>:
‘idf’ : <term’s idf>,
‘posting’ : <term’s posting list>
}

3. QueryParser

QueryParser
+processor: preprocess
+parse(query:string, lemma: boolean, stem: boolean, st: boolean)

QueryParser nhận input là chuỗi kí tự tương ứng với query. Thông qua hàm parse, QueryParser sẽ thực hiện các bước tiền xử lí văn bản ở phần 2.2.2 và trả về một mảng các term đã được xử lí để thực hiện truy vấn.

4. QuerySearch

QuerySearcher
+index: hashmap +id: hashmap +path: string +retrived: array
+open(path:string) +search(query:array)

QuerySearch sẽ truy xuất IPL của corpus thông qua hàm **open()**. Do không thể thao tác trực tiếp trên file, IPL của corpus, tức là file index.json, sẽ được lưu vào biến index. Sau đó hàm **search()** sẽ nhận tập các term đã được xử lí từ class QueryParser và tham chiếu

đến biến *index* để thực hiện các bước truy vấn ở phần 2.2.6. Biến *retrived* được sử dụng để lưu và trả về kết quả truy vấn.

5. **preprocess**

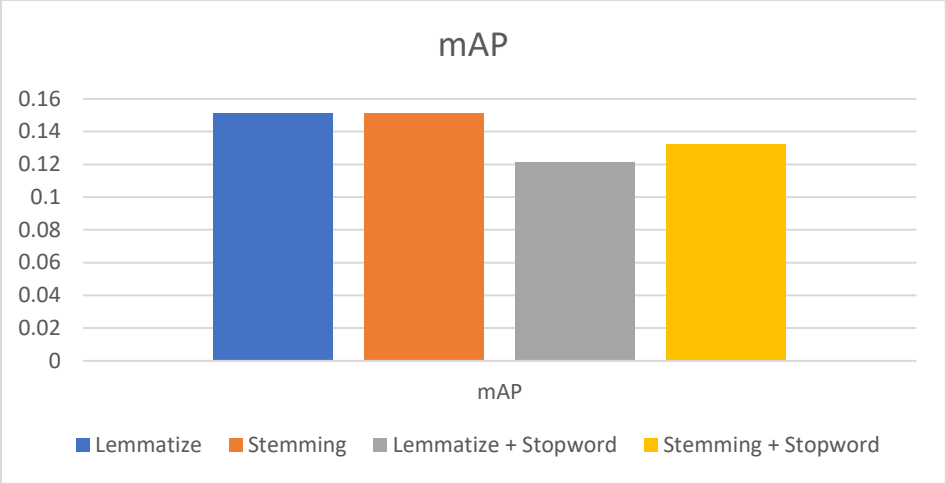
preprocess
+tokenizer: nltk.word_tokenize +stemmer: SnowballStemmer +lemmatizer: WordNetLemmatizer +stopwords: array
+process(query:string) + getPOSTag(treebank_tag)

preprocess là một class hỗ trợ, thực hiện các bước tiền xử lí văn bản cũng như truy vấn được nêu ở phần 2.2.2. Do tính tái sử dụng cao nên được tách thành một class riêng lẻ.
Tại đây, văn bản sẽ được tách token sử dụng hàm **word_tokenize** của nltk. Các token nằm trong danh sách stop word sẽ được loại bỏ. Danh sách stop word sử dụng được liệt kê tại phụ 1 của bài báo cáo. Sau đó, tuỳ vào như cầu mà sẽ được stem bằng **SnowballStemmer** hoặc được lemmatize bằng **WordNetLemmatizer**. Nếu sử dụng WordNetLemmatizer thì token sẽ được hàm **getPOSTag** gắn nhãn part-of-speed trước khi thực hiện lemma.

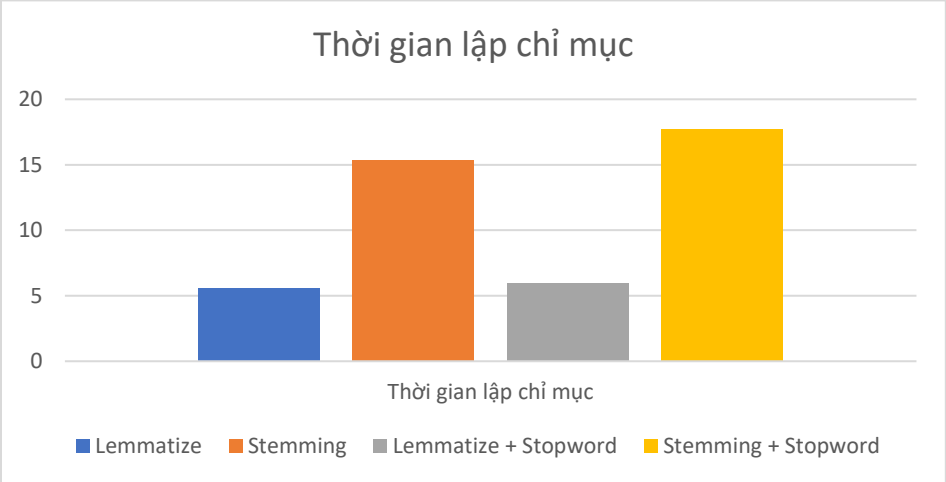
4. KẾT QUẢ THỰC NGHIỆM

Mô hình sẽ được đánh giá dựa trên 3 tiêu chí: thời gian lập chỉ mục, thời gian thực hiện truy vấn và điểm mAP. Mô hình sẽ được so sánh với chính nó, khác biệt chính là việc sử lemmatizer và stemmer, có loại bỏ stop word và không loại bỏ stop word. Kết quả thực nghiệm được hiện trên máy có cấu hình: Window 10 SL, Intel Core i5-8250U 1.6GHz, 8GB RAM. Kết quả đánh giá có ảnh làm bằng chứng ở phụ lục 3.

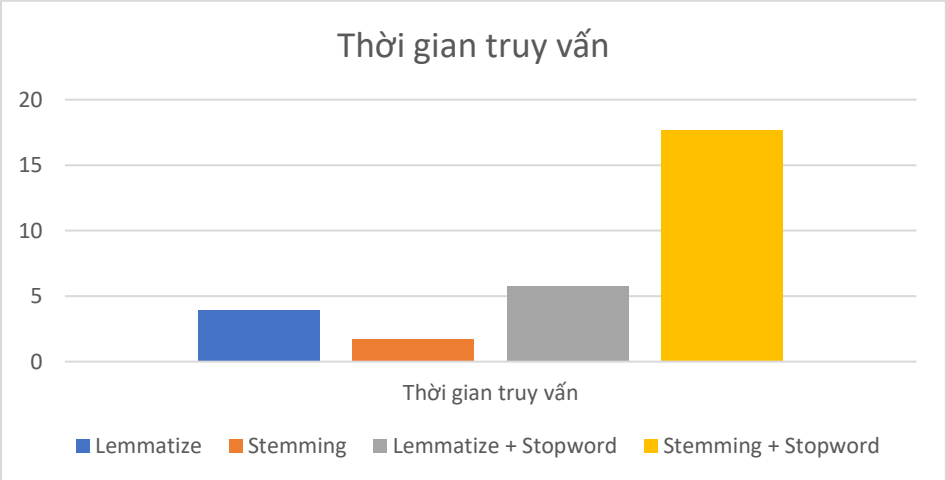
1. mAP



2. Thời gian lập chỉ mục



3. Thời gian truy vấn



5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Ảnh hưởng của lemmatizer và stemmer không đánh kể đối với điểm mAP nhưng ảnh hưởng đáng kể đối với quá trình truy vấn.
- Việc loại giữ lại stop word trong corpus làm ảnh hưởng xấu đến điểm mAP của hệ thống, đồng thời làm tăng thời gian lập chỉ mục cũng như truy vấn.
- Việc sử dụng stemmer làm tăng thời gian lập chỉ mục là kết quả nằm ngoài dự đoán vì về mặt lí thuyết, quá trình lemmatize sẽ tốn thời gian hơn stemming.

2. Hướng phát triển

- Cải thiện cấu trúc lớp, áp dụng tính kế thừa thay vì tạo một đối tượng trong lớp quản lí.
- Thực hiện mở rộng query với hi vọng cải thiện điểm mAP.

6. TÀI LIỆU THAM KHẢO

[1]: Đại học Stanford, *Evaluation of ranked retrieval result*. Truy cập lúc: 12-2020. [Online]: https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ran ked-retrieval-results-1.html

[2]: Đại học Stanford, *Term frequency and weighting*. Truy cập lúc: 12-2020. [Online]: https://nlp.stanford.edu/IR-book/html/htmledition/term-frequency-and-weighting-1.html

[3]: Đại học Stanford, *Stemming and lemmatization*, Truy cập lúc: 12-2020. [Online]: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[4]:Wikipedia, *Information Retrieval*. Truy cập lúc: 12-2020. [Online]: https://en.wikipedia.org/wiki/Information_retrieval

[5]: Christopher, Prabhakar Raghavan, Hinrich Schutze, *Introduction to Information retrieval*, Cambridge University Press, 2008

[6]: Wikipedia, *Vector space model*. Truy cập lúc: 25-1-2020. [Online]: https://en.wikipedia.org/wiki/Vector_space_model

[7]: Đại học Pennsylvania, *Penn Treebank POS tag*. Truy cập lúc: 26-1-2021. [Online]: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

7. PHỤ LỤC

1. Tập stop word sử dụng trong quá trình thực nghiệm

```
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:30:28) [MSC v.1926 32 bit
 (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from nltk.corpus import stopwords
>>> stoplist = stopwords.words('english')
>>> print(stoplist)
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're
", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', '
he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it'
, "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'wit
h', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'a
fter', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off
', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'wh
en', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'mo
st', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so
', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", '
should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'a
ren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma'
, 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "sha
n't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "
won't", 'wouldn', "wouldn't"]
>>> len(stoplist)
179
>>>
```

2. Các nhãn POS phổ biến

Các nhãn POS được sử dụng trong dự án Treebank của đại học Pennsylvalnia[7]

Number	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential *there*
5	FW	Foreign word

6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle

31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

3. Ảnh kết quả thực hiện đánh giá

Ghi chú: test1.py thực hiện stem và loại bỏ stop word, test2.py thực hiện lemmatize và loại bỏ stop word, vì lí do kĩ thuật mà nội dung test không được hiển thị

```
PS D:\PyProject\IR\practice1\VectorSpace-IR-Model> python test1.py
Indexing Elapsed Time: 15.335989713668823
-----*-----
mAP score: 0.151381372316218
Retrival elasped time: 1.7315545082092285
PS D:\PyProject\IR\practice1\VectorSpace-IR-Model> python test2.py
Indexing Elapsed Time: 5.569757461547852
-----*-----
mAP score: 0.1513484808593485
Retrival elasped time: 3.94875431060791
PS D:\PyProject\IR\practice1\VectorSpace-IR-Model> python test3.py
Stemming + Keeping stopword testing
Indexing Elapsed Time: 17.67235016822815
-----*-----
mAP score: 0.13236348229335565
Retrival elasped time: 3.998267889022827
PS D:\PyProject\IR\practice1\VectorSpace-IR-Model> python test4.py
Lemma + Keeping stopwords testing
Indexing Elapsed Time: 5.935523509979248
-----*-----
mAP score: 0.12161438929284352
Retrival elasped time: 5.753743410110474
PS D:\PyProject\IR\practice1\VectorSpace-IR-Model>
```