

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỌC MÁY TÍNH



Truy xuất thông tin – CS419.N21

BÁO CÁO ĐỒ ÁN MÔN HỌC

Đề tài

**TÌM HIỂU VÀ CÀI ĐẶT MÔ HÌNH VECTOR SPACE VÀ
LATENT SEMANTIC INDEXING (LSI)**

GVHD: Ths. Nguyễn Trọng Chính

Nhóm sinh viên thực hiện:

Nguyễn Trần Minh Anh 20520394

Lê Nguyễn Bảo Hân 20520174

Đoàn Phương Khanh 20521443

Hồ Chí Minh, 12 tháng 07 năm 2023

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN.....	1
1.1. Đặt vấn đề	1
1.2. Mục tiêu	2
1.3. Quá trình truy xuất	2
1.4. Giới thiệu chung.....	4
1.4.1. Bộ dữ liệu.....	4
1.4.1.1. Cranfield.....	4
1.4.1.2. NFCorpus	5
1.4.2. Mô hình truy xuất thông tin	5
CHƯƠNG 2. MÔ HÌNH VECTOR SPACE	7
2.1. Giới thiệu mô hình	7
2.1.1. Ý tưởng	7
2.1.2. Phương pháp biểu diễn	7
2.2. Tiền xử lý dữ liệu.....	8
2.2.1. Term.....	9
2.2.2. Tách từ	9
2.2.3. Loại bỏ dấu câu.....	10
2.2.4. Chuyển đổi chữ thường	12
2.2.5. Loại bỏ stopwords.....	12
2.2.6. Lấy gốc từ	13
2.3. Lập chỉ mục.....	16
2.4. Truy xuất chỉ mục	18
2.4.1. Concept vector	18
2.4.2. Trọng số (weight).....	19

2.4.2.1. Term Frequency -TF (Tần số xuất hiện của Term).....	20
2.4.2.2. Inverse Document Frequency – IDF (Nghịch đảo tần suất tài liệu)	21
2.4.2.3. TF-IDF.....	22
2.5. Tính độ tương đồng và xếp hạng	23
CHƯƠNG 3. MÔ HÌNH LATEN SEMANTIC INDEXING.....	25
3.1. Giới thiệu mô hình	25
3.1.1. Những hạn chế của mô hình Vector Space.....	25
3.1.2. Giới thiệu mô hình LSI.....	26
3.1.3. Ý tưởng	27
3.2. Tiền xử lý dữ liệu.....	28
3.3. Lập chỉ mục.....	29
3.3.1. Lập ma trận tần số term-document	29
3.3.2. Phân rã ma trận term-document bằng thuật toán SVD	31
3.3.2.1. Thuật toán phân rã ma trận SVD.....	31
3.3.2.1.1. Trị riêng và vector riêng	31
3.3.2.1.2. Ý tưởng thuật toán	32
3.3.2.1.3. Các bước thực hiện	32
3.3.2.2. Phân rã ma trận term-document bằng thuật toán SVD	33
3.3.2.2.1. Các ma trận thừa số trong phép phân rã	33
3.3.2.2.2. Minh họa các bước thực hiện phân rã ma trận term-document....	35
3.3.3. Biểu diễn tài liệu trong không gian ngữ nghĩa	37
3.3.3.1. Ma trận từ khóa và ma trận tài liệu	37
3.3.3.2. Minh họa biểu diễn tài liệu trong không gian ngữ nghĩa	38
3.4. Truy xuất chỉ mục	39
3.4.1. Tiền xử lý câu truy vấn	40

3.4.2. Chuyển câu truy vấn về dạng vector thống kê tần số xuất hiện của các term có trong câu truy vấn	40
3.4.3. Ánh xạ vector biểu diễn câu truy vấn vào trong không gian ngữ nghĩa tiềm ẩn.....	41
3.4.4. Ánh xạ trực tiếp vector truy vấn vào không gian ngữ nghĩa	43
3.5. Tính độ tương đồng và xếp hạng	43
3.5.1.1. Tính độ tương đồng giữa câu truy vấn và các tài liệu	44
3.5.1.2. Xếp hạng kết quả truy vấn.....	46
3.6. Cắt giảm ma trận	47
3.6.1. Đặt vấn đề	47
3.6.2. Các bước thực hiện cắt giảm	47
3.6.3. Minh họa cắt giảm ma trận	50
3.6.4. Ý nghĩa của cắt giảm ma trận	51
CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM.....	52
4.1. Xử lý dữ liệu	52
4.1.1. Dữ liệu Cranfield	52
4.1.2. Dữ liệu NFCorpus.....	53
4.2. Phương pháp đánh giá.....	55
4.3. Mô hình Vector Space	57
4.4. Mô hình LSI.....	59
4.5. So sánh hai mô hình	62
4.6. Mô hình Vector Space, LSI và thư viện Whoosh	63
CHƯƠNG 5. TỔNG KẾT.....	64
5.1. Kết luận	64
5.2. Hướng phát triển	64
TÀI LIỆU THAM KHẢO.....	66

DANH MỤC HÌNH ẢNH

Hình 1.1. Minh họa truy xuất thông tin	1
Hình 1.2. Quá trình truy xuất thông tin	3
Hình 1.3. Mô hình hóa quá trình truy vấn nói chung	4
Hình 2.1: Minh họa cho Vector Space Model.....	8
Hình 2.2. Ví dụ tách từ	10
Hình 2.3. Các dấu câu thường gặp	11
Hình 2.4. Ví dụ loại bỏ dấu câu	12
Hình 2.5. Ví dụ chuyển đổi chữ thường	12
Hình 2.6. Minh họa loại bỏ stopwords	13
Hình 2.7. Ví dụ loại bỏ stopwords.....	13
Hình 2.8. Minh họa phương pháp Stemming	15
Hình 2.9. Ví dụ lấy gốc từ bằng Stemming.....	15
Hình 2.10. So sánh Stemming và Lemmatization	16
Hình 2.11. Ví dụ lấy gốc từ bằng Lemmatization	16
Hình 2.12. Mô tả concept vector	18
Hình 2.13. Các tài liệu và truy vấn sau khi được biểu diễn thành vector.....	23
Hình 3.1. Minh họa các vector tài liệu trong không gian hệ trục tọa độ	25
Hình 3.2. Các bước thực hiện truy xuất thông tin với mô hình LSI.....	27
Hình 3.3. Cắt giảm ma trận S	48
Hình 3.4. Cắt giảm ma trận Σ	49
Hình 3.5. Cắt giảm ma trận UT	49
Hình 4.1. Tập tài liệu Cranfield gốc	52
Hình 4.2. Tập truy vấn Cranfield gốc.....	53
Hình 4.3. Tập tài liệu liên quan Cranfield gốc	53
Hình 4.4. Tập tài liệu NF Corpus gốc	54
Hình 4.5. Tập truy vấn NF Corpus gốc	54
Hình 4.6 Ví dụ câu truy vấn trong NF Corpus	54
Hình 4.7. Tập tài liệu liên quan NF Corpus gốc.....	55
Hình 4.8. Ví dụ tập tài liệu liên quan của truy vấn 2.....	55

Hình 4.9. Minh họa đồ thị biểu diễn AP trước khi nội suy	57
Hình 4.10. Minh họa đồ thị biểu diễn AP sau khi nội suy	57
Hình 4.11. Sơ đồ thực hiện của mô hình VSM	57
Hình 4.12. Sơ đồ thực hiện của mô hình LSI	60
Hình 4.13. Kết quả truy vấn theo số chiều được giữ lại trên Cranfield	61
Hình 4.14. Kết quả truy vấn theo số chiều được giữ lại trên NFCorpus	62

DANH MỤC BẢNG BIỂU

Bảng 3.1. Bảng thống kê tần số xuất hiện của mỗi term trong câu truy vấn.....	41
Bảng 3.2. Bảng xếp hạng kết quả truy vấn thông qua độ đo <i>cosine</i>	46
Bảng 4.1. Thống kê kết quả thực nghiệm mô hình Vector Space	58
Bảng 4.2. Kết quả thực nghiệm trên bộ dữ liệu NFCorpus	59
Bảng 4.3. Kết quả thực nghiệm các lần thử số chiều khác nhau	61
Biểu đồ 4.1. Số lượng term thu được của mỗi phương pháp.....	58
Biểu đồ 4.2. Kết quả thực nghiệm trên bộ dữ liệu Cranfield	59
Biểu đồ 4.3. Khảo sát giá trị mAP và thời gian truy vấn khi thay đổi giá trị k.....	60
Biểu đồ 4.4. So sánh 2 mô hình với 2 bộ dữ liệu về kết quả mAP và thời gian truy vấn	62
Biểu đồ 4.5. So sánh mô hình Vector Space, mô hình LSI và thư viện Whoosh.....	63

PHÂN CÔNG CÔNG VIỆC

<div>Thành viên</div> <div>Công việc</div>	<div>Đoàn Phương Khanh</div> <div>20521443</div>	<div>Nguyễn Trần Minh Anh</div> <div>20520394</div>	<div>Lê Nguyễn Bảo Hân</div> <div>20520174</div>
<i>Tìm hiểu mô hình Vector Space</i>		✓	✓
<i>Tìm hiểu mô hình LSI</i>	✓		✓
<i>Thực nghiệm Vector Space</i>		✓	✓
<i>Thực nghiệm LSI</i>	✓		✓
<i>Viết báo cáo Vector Space</i>		✓	
<i>Viết báo cáo LSI</i>	✓		✓
<i>Viết báo cáo thực nghiệm</i>	✓	✓	
<div>Mức độ hoàn thành (%)</div>	100%	100%	100%

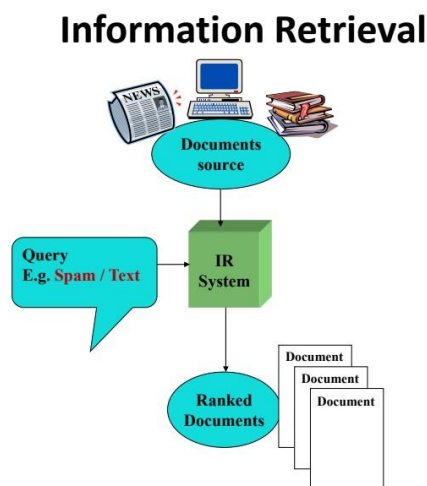
CHƯƠNG 1. TỔNG QUAN

1.1. Đặt vấn đề

Trong thời đại hiện nay, việc sử dụng Internet đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày, từ giao tiếp, thu thập thông tin cho đến hoạt động kinh doanh và nhiều lĩnh vực khác. Việc lưu trữ thông tin đã phát triển vượt bậc và chứa đựng một lượng lớn tài liệu có thể được truy cập công khai thông qua hàng ngàn máy chủ trên toàn cầu.

Tuy nhiên, với các kho tài liệu ngày càng lớn, việc quản lý chúng trở nên đắt đỏ và khó khăn hơn. Sự bùng nổ thông tin đã đặt ra những thách thức lớn đối với các nhà nghiên cứu, đòi hỏi phải tìm ra các phương pháp để tìm kiếm thông tin hiệu quả hơn trong thời gian ngắn hơn.

Truy xuất thông tin (Information Retrieval - IR) là một lĩnh vực nghiên cứu chung đang giải quyết vấn đề này. Thuật ngữ "thông tin" được sử dụng rất rộng rãi, vì vậy truy xuất thông tin là quá trình áp dụng cho nhiều loại thông tin và ứng dụng khác nhau, nhằm tìm ra các phương pháp lưu trữ và truy xuất thông tin nhanh chóng nhất cho các nguồn thông tin đó.



Hình 1.1. Minh họa truy xuất thông tin

Truy xuất thông tin là quá trình tìm kiếm thông tin trong tài liệu, có thể dựa trên văn bản, hình ảnh và âm thanh. Trong hệ thống truy xuất thông tin, ta có các thành phần

đầu vào và đầu ra như sau:

- *Đầu vào*: một bộ sưu tập các tài liệu và một câu truy vấn, đại diện cho nhu cầu thông tin của người dùng.
- *Đầu ra*: một tập hợp các tài liệu được xem là liên quan đến câu truy vấn và được xếp hạng theo mức độ tương quan.

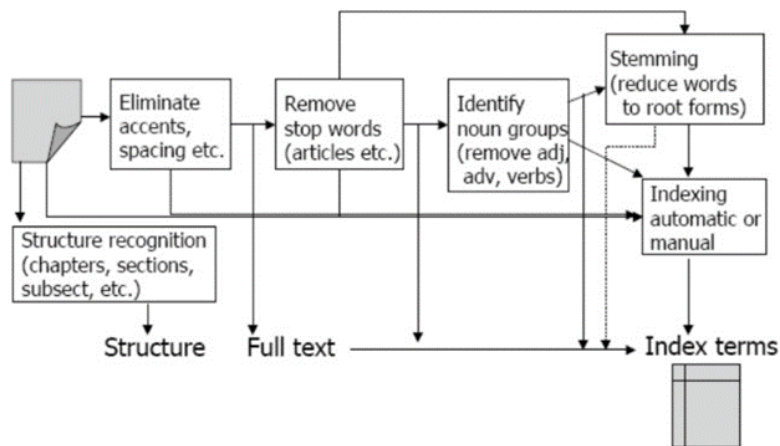
1.2. Mục tiêu

Thông qua đề tài này, chúng em sẽ thử nghiệm nhiều phương thức tiền xử lý khác nhau, kết hợp với 2 mô hình VSM và LSI để tìm và cho ra hệ thống IR hiệu quả nhất có thể. Bên cạnh đó, nhóm chúng em mong muốn sẽ hiểu cặn kẽ 2 mô hình này. Từ đó tìm ra nhiều cách để cải tiến cũng như là tiền đề để có thể dễ dàng tìm hiểu các mô hình xử lý khác được dùng trong hệ thống IR.

Trong báo cáo, sẽ chỉ ra các bước xử lý đối với một IRS, ở mỗi bước sẽ làm rõ mục đích sử dụng cũng như cách hoạt động của chúng. Đối với 2 mô hình chính là VSM và LSI, chúng em sẽ làm rõ cách hoạt động của chúng, từ đó cho ra môi trường phù hợp với 2 mô hình này. Bên cạnh đó, sẽ tiến hành đưa ra so sánh và kết luận sau khi thực hiện các thử nghiệm trên các bộ dữ liệu khác nhau. Ngoài ra, sẽ kết hợp so sánh thêm với thư viện Whoosh – một trong những thư viện tìm kiếm văn bản phổ biến nhất trong cộng đồng Python, để đưa ra nhận xét khách quan nhất với 2 mô hình VSM và LSI.

1.3. Quá trình truy xuất

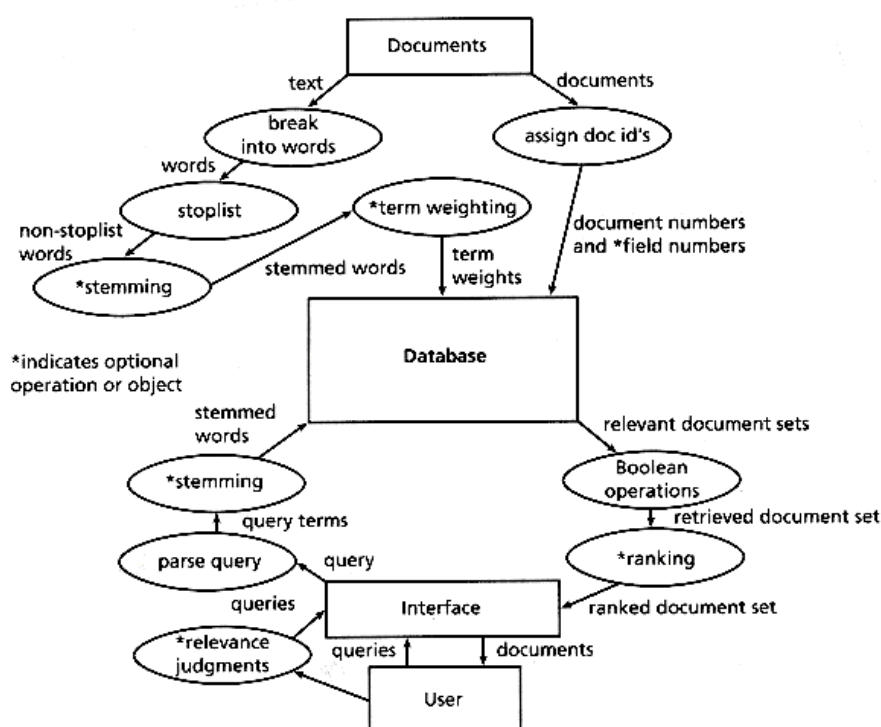
Truy xuất thông tin được thực hiện qua 3 giai đoạn:



Hình 1.2. Quá trình truy xuất thông tin

- *Giai đoạn 1: Tiền xử lý dữ liệu.* Tập tài liệu văn bản được tiền xử lý bằng một số phương pháp thao tác văn bản tự động như phân tích từ vựng (lexical analysis), loại bỏ từ dừng (stopword removing), lấy gốc từ (stemming) từ dạng văn bản đơn giản (plain text) của tài liệu. Kết quả nhận được là tập các từ (term) hay còn được hiểu là các khái niệm (concept), được coi là khung nhìn logic (logical view) của tài liệu.
- *Giai đoạn 2: Lập chỉ mục.* Quá trình tạo ra một danh sách các từ khóa (terms) được trích xuất thông qua giai đoạn tiền xử lý dữ liệu. Mục đích của giai đoạn này là giúp cho việc tìm kiếm thông tin trở nên nhanh chóng và chính xác hơn. Sau đó, các term sẽ được đánh trọng số. Việc làm này nhằm định lượng mức độ quan trọng của các thuộc tính và từ khóa trong văn bản. Trọng số được đánh dựa trên các phương pháp tính toán như tần suất xuất hiện của từ khóa trong văn bản (term frequency weighting - tf), tần suất của từ khóa trong toàn bộ nguồn dữ liệu (inverse document frequency – idf).
- *Giai đoạn 3: Truy vấn.* Người dùng sẽ đưa ra yêu cầu (request) của mình dưới dạng ngôn ngữ tự nhiên. Từ đó đòi hỏi hệ thống phải xử lý yêu cầu đó của người dùng để cho ra tập các term. Sau đó, các term sẽ được tìm kiếm trên tập tài liệu lớn. Nếu term xuất hiện trong tài liệu nào thì hệ thống sẽ trả về tập hợp của các tài liệu thu thập được theo tất cả các từ hoặc một số từ, tùy theo kiểu truy vấn. Các tài liệu thu thập được sẽ được đánh giá theo độ liên quan tương đối của truy vấn, tài liệu nào có độ liên quan cao thì có số ranking càng thấp.

Từ đó hệ thống sẽ trả về tài liệu phù hợp nhất với yêu cầu của người dùng.



Hình 1.3. Mô hình hóa quá trình truy vấn nói chung

1.4. Giới thiệu chung

Trong đồ án môn học này, nhóm lựa chọn thực nghiệm hai mô hình Vector Space (Vector Space Model – VSM) và mô hình LSI (Laten Semantic Indexing) trên hai bộ dữ liệu là Cranfield và NFCorpus.

1.4.1. Bộ dữ liệu

1.4.1.1. Cranfield

Cranfield là một tập dữ liệu tiếng Anh có chứa các bản ghi thông tin ngắn gọn liên quan đến lĩnh vực nghiên cứu khoa học. Tập dữ liệu này gồm 1400 tài liệu và 225 câu truy vấn kèm theo kết quả tương ứng đã được biết trước. Các câu truy vấn này thường được xây dựng dựa trên các tài liệu trong bộ dữ liệu và sử dụng để đánh giá hiệu suất của các hệ thống truy vấn thông tin và công cụ xử lý ngôn ngữ tự nhiên.

Cranfield là một nguồn tài nguyên quan trọng trong việc nghiên cứu và phát triển các công nghệ và phương pháp xử lý ngôn ngữ tự nhiên, truy vấn thông tin, phân loại và tìm kiếm thông tin khoa học.

1.4.1.2. NFCorpus

NFCorpus là một tập dữ liệu truy xuất thông tin y tế bằng tiếng Anh. Tập dữ liệu này chứa tổng cộng 3.244 câu truy vấn bằng ngôn ngữ tự nhiên (viết bằng ngôn ngữ không chuyên ngành) được thu thập từ trang web NutritionFacts.org, cùng với 169.756 đánh giá tài liệu liên quan được trích xuất tự động cho 9.964 tài liệu y tế (viết bằng một ngôn ngữ phức tạp và chuyên ngành).

Các đánh giá về mức độ liên quan được xây dựng từ các liên kết trực tiếp và gián tiếp trên trang web NutritionFacts.org. Cấp độ phù hợp nhất tương ứng với một liên kết trực tiếp từ một bài báo NutritionFacts (truy vấn) đến một bài báo y tế (tài liệu) từ phần nguồn được trích dẫn của một trang. Cấp độ tiếp theo là một truy vấn liên kết nội bộ đến một bài báo NutritionFacts khác, sau đó liên kết trực tiếp một tài liệu y tế. Cuối cùng, mức thấp nhất được dành cho các truy vấn và tài liệu được kết nối thông qua hệ thống chủ đề/thẻ trên NutritionFacts.org.

1.4.2. Mô hình truy xuất thông tin

Vector Space Model (VSM) là một mô hình đại số đơn giản và phổ biến được sử dụng để biểu diễn văn bản. Trong VSM, mỗi văn bản được biểu diễn thành một vector trong không gian đa chiều, mỗi chiều tương ứng với một thuộc tính hoặc từ trong văn bản. Độ tương đồng giữa các văn bản được tính toán dựa trên sự tương đồng của các vector này. Sau đó, độ tương đồng này được xếp hạng để hệ thống có thể trả về các kết quả liên quan đến câu truy vấn của người dùng.

Tuy nhiên, mô hình này tồn tại một số hạn chế, đặc biệt là khi áp dụng để xử lý các văn bản dài hoặc văn bản có vấn đề từ đa nghĩa hay đồng nghĩa. Đồng thời, VSM không có khả năng hiểu và xử lý sự tương quan ngữ nghĩa giữa các từ. Điều này đồng nghĩa với việc mất đi khả năng hiểu được ý nghĩa và ngữ cảnh của các từ và cụm từ trong văn bản.

Để cải thiện các vấn đề trên, mô hình Latent Semantic Indexing (LSI) được đề xuất sử dụng phương pháp SVD (Singular Value Decomposition) để giảm chiều dữ liệu và tìm ra những khái niệm ẩn trong văn bản. Kết quả của LSI là một không gian vector mới có số chiều thấp hơn, trong đó các vector biểu diễn văn bản dựa trên các khái niệm ẩn thay vì các thuộc tính cụ thể. Điều này giúp mô hình LSI xử lý tốt hơn với các văn bản dài và nhận diện được các văn bản có chủ đề tương tự nhưng có sử dụng các từ khác nhau.

CHƯƠNG 2. MÔ HÌNH VECTOR SPACE

2.1. Giới thiệu mô hình

2.1.1. Ý tưởng

Ý tưởng của mô hình không gian vector (vector space model) trong lĩnh vực truy xuất thông tin là biểu diễn các văn bản và câu truy vấn dưới dạng các vector trong không gian đa chiều. Mô hình này đánh giá mức độ tương đồng giữa văn bản và câu truy vấn dựa trên sự gần nhau của các vector biểu diễn trong không gian vector.

Trong mô hình này, mỗi từ trong văn bản và câu truy vấn được chuyển đổi thành một thành phần trong vector, sau đó các vector biểu diễn được xây dựng dựa trên tần suất xuất hiện hoặc trọng số của các từ trong văn bản.

Sau khi biểu diễn các văn bản và câu truy vấn dưới dạng vector, mô hình tính toán độ tương đồng giữa chúng bằng cách sử dụng các phép toán đại số tuyến tính như cosine similarity. Các văn bản có độ tương đồng cao sẽ có các vector biểu diễn gần nhau trong không gian vector.

2.1.2. Phương pháp biểu diễn

Mô hình Vector Space được biểu diễn như sau:

- i. Biến đổi các truy vấn (queries) và tài liệu (documentaries) thành dạng vector:

$$d_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$$

$$q_i = (w_{1,i}, w_{2,i}, \dots, w_{t,i})$$

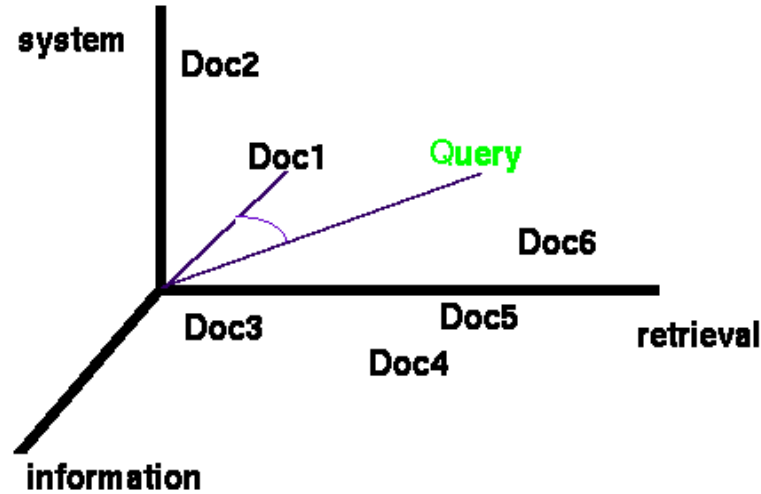
- ii. Tính toán độ tương đồng *Similarity* giữa tài liệu và truy vấn

$$Similarity(Rep(q), Rep(d))$$

Trong đó:

- *Similarity* là độ tương đồng
- *Rep(d)* là hàm biểu diễn tài liệu *d*

- $Rep(q)$ là hàm biểu diễn của truy vấn q
- iii. Sau đó sử dụng một độ đo khoảng cách trên vector q và d_j để xếp hạng các tài liệu.



Hình 2.1: Minh họa cho Vector Space Model

Các giả định:

- Các tài liệu và các truy vấn được biểu diễn với cùng định dạng: cùng kích thước và cách biểu diễn trọng số.
- Độ liên quan (relevance) giữa tài liệu d và truy vấn q :

$$Relevance(d, q) \leftrightarrow Similarity(d, q)$$

- Độ tương đồng $Similarity(d, q)$ được tính bằng Cosine Similarity.

2.2. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu (Data Preprocessing) là một thao tác quan trọng trước khi phân tích hay truy xuất dữ liệu. Nhiệm vụ của quá trình này là lấy dữ liệu thô và biến đổi chúng thành định dạng mà máy tính và máy học có thể hiểu và phân tích được. Với các bài toán khác nhau, quá trình này cũng có sự khác nhau trong thao tác xử lý. Trong vấn đề truy xuất thông tin, quá trình tiền xử lý có ảnh hưởng mạnh mẽ đến việc biểu diễn các vector tài liệu hay truy vấn, từ đó ảnh hưởng đến kết quả truy xuất.

2.2.1. Term

Thuật ngữ "term" thường được sử dụng để chỉ các từ hoặc thuật ngữ xuất hiện trong văn bản. Một term có thể là một từ đơn, một từ ghép hoặc một cụm từ. Việc xác định term đóng vai trò quan trọng trong việc biểu diễn và tính toán độ liên quan giữa các tài liệu và truy vấn.

Báo cáo này thực hiện lần lượt các bước tiền xử lý sau để thu được term:

- Tách từ
- Loại bỏ dấu câu
- Chuyển đổi chữ thường
- Loại bỏ stopwords
- Lấy gốc từ (Stemming và Lemmatization)

Điều này giúp rút gọn và tối ưu hóa tập tài liệu văn bản, đồng thời làm tăng hiệu quả truy xuất

2.2.2. Tách từ

Tách từ (Tokenization) là một trong những bước quan trọng nhất trong quá trình tiền xử lý văn bản. Đây là quá trình phân tách một cụm từ, câu, đoạn văn hoặc tài liệu văn bản thành các đơn vị nhỏ hơn. Mỗi đơn vị này là một token. Các token có thể là bất kỳ đơn vị nào như từ (word), phụ từ (sub-word) hoặc thậm chí là một ký tự (character).

Thông thường một đoạn văn bản sẽ bao gồm các chuỗi ký tự được phân cách nhau bởi khoảng trắng. Và bước tách từ sẽ khiến các chuỗi này trở thành riêng lẻ. Đây là nền tảng cho việc xây dựng từ điển và tính toán biểu diễn ở bước tiếp theo.

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.



'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', '.'

Hình 2.2. Ví dụ tách từ

Có thể thấy, ví dụ trên được xử lý tương đối đơn giản vì chỉ tách từ dựa trên khoảng trắng. Tuy nhiên, ta sẽ xử lý dấu phẩy trên ' như thế nào?

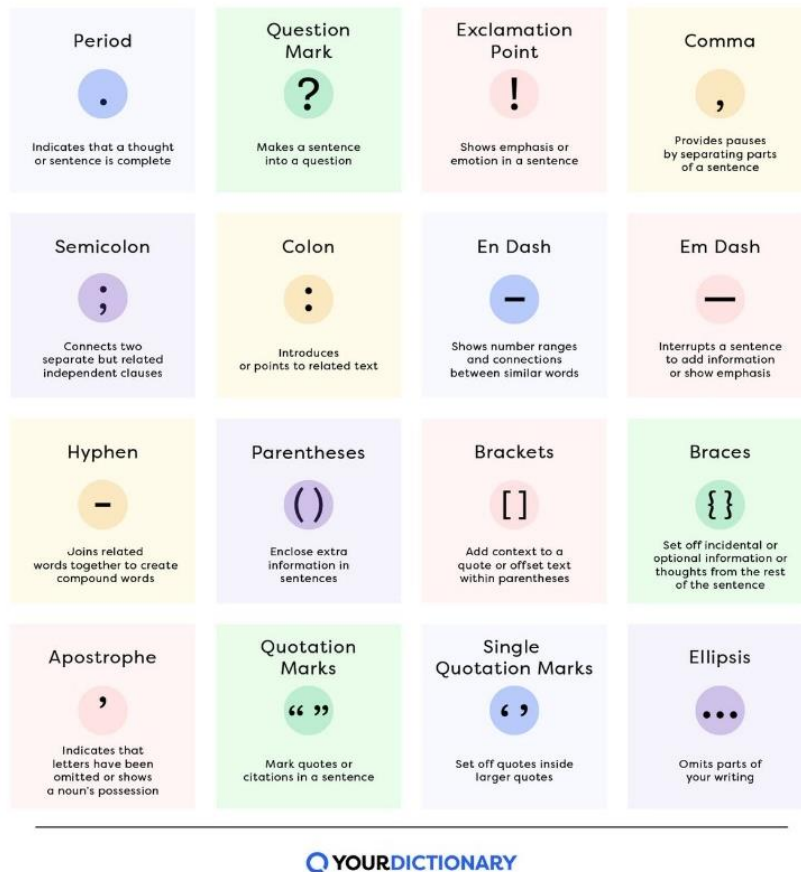
Lấy ví dụ trong câu: *Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.*

- Với *O'Neill*, ta có những phương án sau: (i) neill, (ii) oneill, (iii) o'neill, (iv) o' neill, (v) o neill
- Với *aren't*, ta có: (i) aren't, (ii) arent, (iii) are n't, (iv) aren t

Một cách đơn giản là tách dựa trên các kí tự không phải kí tự chữ cái, chữ số, nhưng o neill trông khá ổn, còn aren t thì lại khá tệ. Vì vậy, việc chọn cách token phù hợp với yêu cầu bài toán sẽ ảnh hưởng rất lớn.

2.2.3. Loại bỏ dấu câu

Dấu câu (punctuations) là tập hợp các ký tự sau: !"#\$%&'()*+,-./:;<=>?@[\\]^_`{|}~



Hình 2.3. Các dấu câu thường gặp

Loại bỏ dấu câu (punctuation removal) là quá trình loại bỏ các dấu câu trong một văn bản. Sau khi tokenization, các token (chuỗi ký tự) lúc này có thể sẽ chứa cả các dấu câu hay chính bản thân token đó cũng là một dấu câu. Bản thân các dấu câu này không đóng góp nhiều ý nghĩa về mặt ngữ nghĩa trong việc phân tích văn bản. Vì vậy, việc loại bỏ các dấu câu giúp đơn giản hóa văn bản và giảm số lượng đặc trưng không cần thiết.

'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', '!'!



'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'

Hình 2.4. Ví dụ loại bỏ dấu câu

2.2.4. Chuyển đổi chữ thường

Văn bản đầu vào có thể chứa cả các ký tự viết hoa và viết thường. Việc chuyển đổi chữ thường (Lowercase) giúp ta giúp đồng nhất các từ thành chuỗi ký tự chỉ gồm các ký tự viết thường, tránh việc các từ giống hệt nhau nhưng lại bị xem như khác nhau (Case Sensitivity). Ví dụ: “The” và “the” có thể bị xem là hai token hoàn toàn khác nhau.

'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'



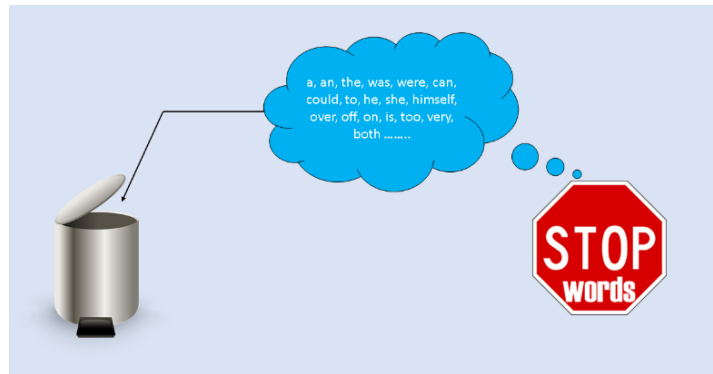
'the', 'boundary', 'layer', 'in', 'simple', 'shear',
'flow', 'past', 'a', 'flat', 'plate', 'the', 'boundary',
'layer', 'equations', 'are', 'presented', 'for', 'steady',
'incompressible', 'flow', 'with', 'no', 'pressure',
'gradient'

Hình 2.5. Ví dụ chuyển đổi chữ thường

2.2.5. Loại bỏ stopwords

Stopwords (từ dừng) là những từ không đóng góp nhiều ý nghĩa cho văn bản (bao gồm mạo từ, giới từ, liên từ, từ có tần xuất xuất hiện cao...). Ví dụ như các từ: “i”, “me” “the”,... Các từ này thường không giúp ích được nhiều trong phân tích văn

bản vì thường gây ra các vấn đề về sự mơ hồ. Tùy trường hợp mà ta có thể quyết định sẽ loại bỏ stopwords hay không.



Hình 2.6. Minh họa loại bỏ stopwords

Quá trình loại bỏ stopwords có thể được chia thành hai loại:

- Từ cần loại bỏ nằm trong danh sách stopwords có sẵn. Quá trình này sẽ được thực hiện trong phần nhận dạng từ, có nghĩa là từ này sẽ không thể đi qua bộ nhận dạng từ.
- Từ cần loại bỏ không nằm trong danh sách stopwords nhưng xuất hiện với tần suất vượt ngưỡng qui định trong tập dữ liệu. Quá trình này được thực hiện duyệt trên bảng băm để tìm các stopwords, thêm chúng vào danh sách danh sách stopwords và loại phần tử chứa từ đó khỏi bảng băm.

'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'



'boundary', 'layer', 'simple', 'shear', 'flow', 'past',
'flat', 'plate', 'boundary', 'layer', 'equations',
'presented', 'steady', 'incompressible', 'flow',
'pressure', 'gradient'

Hình 2.7. Ví dụ loại bỏ stopwords

2.2.6. Lấy gốc từ

Lấy gốc từ là quá trình biến đổi từ về dạng ngữ pháp gốc mà vẫn duy trì tính ngữ nghĩa. Thao tác này giúp giảm đi số lượng từ vựng mang ngữ pháp gốc giống nhau nhưng bị biến đổi sang dạng ngữ pháp khác.

Việc lấy gốc từ trước khi biểu diễn thành các vector đặc trưng có ưu điểm là làm giảm kích thước vector và quy nhiều dạng biến tố về cùng một từ (ví dụ từ development, developments và developing). Một phương pháp lấy gốc từ phổ biến là giải thuật của Porter (1980), giải thuật này áp dụng tập các quy tắc đối với hậu tố của một từ nhằm loại bỏ nó.

Có hai sự lựa chọn trong phương pháp lấy gốc từ:

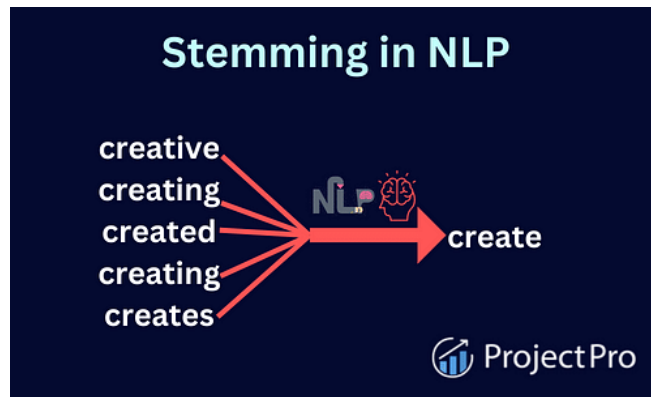
1) Stemming:

Phương pháp này cho rằng biến thể của từ được sinh ra từ việc thêm vào một vài ký tự vào cuối từ đó.

Ví dụ: những từ như talked, talking, talks 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 talk.

Các bộ xử lý stemming này thường được là Stemmer. Vì nguyên tắc hoạt động đơn giản, stemming sẽ có tốc độ thực hiện tương đối nhanh chóng. Tuy nhiên, nếu xét về sự đa dạng về các biến thể từ tiếng Anh, ta thấy nhược điểm của stemming là rất lớn khi không thể xử lý các biến thể bất quy tắc “grow - grew - grown”, “dig - dug - dug”,... Các từ này tuy thuộc cùng một biến thể từ, nhưng stemming lại không có cơ chế xử lý.

Một trường hợp đặc biệt khác là “watch” và “watches”. Sau khi thực hiện stemming, kết quả trả về lần lượt là “watch” và “watche”, hai từ này vẫn chưa được quy về cùng một biến thể.



Hình 2.8. Minh họa phương pháp Stemming

'boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat',
'plate', 'boundary', 'layer', 'equations', 'presented',
'steady', 'incompressible', 'flow', 'pressure', 'gradient'



Stemmer:

'boundari', 'layer', 'simpl', 'shear', 'flow', 'past', 'flat',
'plate', 'boundari', 'layer', 'equat', 'present', 'stead',
'incompress', 'flow', 'pressur', 'gradient'

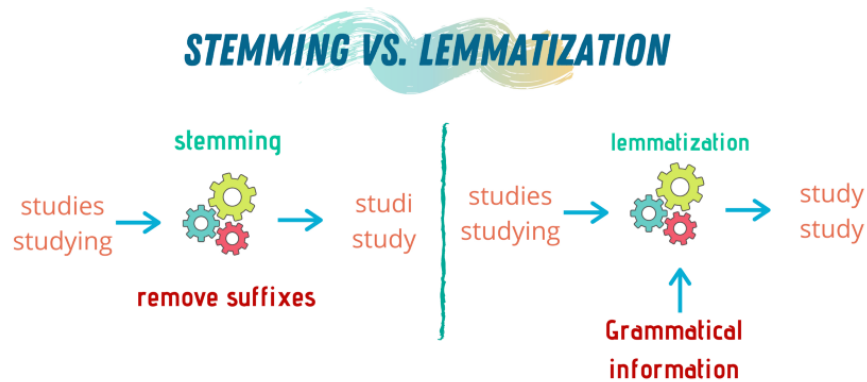
Hình 2.9. Ví dụ lấy gốc từ bằng Stemming

2) Lemmatization

Lemmatization là một phương pháp cải tiến hơn stemming trong việc biến đổi các từ về dạng ngữ pháp gốc. Phương pháp này xử lý được các trường hợp đã nêu trên mà stemming không thực hiện được.

Cụ thể, lemmatization sẽ xử lý bằng một bộ từ điển hoặc một bộ ontology để đảm bảo rằng các từ như “goes“, “went” và “go” chắc chắn có kết quả trả về như nhau. Các từ danh từ như mouse, mice cũng đều được đưa về cùng một dạng như nhau. Bộ xử lý lemmatization này được gọi 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. Trong các ứng dụng xử lý ngôn ngữ tự nhiên cần ưu tiên độ chính xác và thời gian không quan trọng, ta có thể sử dụng Lemmatization.



Hình 2.10. So sánh Stemming và Lemmatization

'boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat',
'plate', 'boundary', 'layer', 'equations', 'presented',
'steady', 'incompressible', 'flow', 'pressure', 'gradient'



Lemmatizer:

'boundary', 'layer', 'simple', 'shear', 'flow', 'past', 'flat',
'plate', 'boundary', 'layer', 'equation', 'present', 'steady',
'incompressible', 'flow', 'pressure', 'gradient'

Hình 2.11. Ví dụ lấy gốc từ bằng Lemmatization

2.3. Lập chỉ mục

Lập chỉ mục là một phương pháp quét một lần trên các file văn bản và tạo danh sách các thuật ngữ (từ, cụm từ) có trong file, đồng thời lưu trữ các thông tin liên quan đến thuật ngữ đó, chẳng hạn như vị trí xuất hiện, tần suất, độ quan trọng, và các thông tin khác. Quá trình này tạo ra một cấu trúc dữ liệu riêng gọi là chỉ mục.

Chỉ mục giúp tìm kiếm và truy xuất thông tin nhanh chóng hơn, vì thay vì phải quét lại toàn bộ file văn bản mỗi khi cần tìm kiếm một thuật ngữ, ta chỉ cần tra cứu trong chỉ mục để xác định vị trí và thông tin liên quan đến thuật ngữ đó. Việc tổ chức dữ liệu trong chỉ mục giúp tối ưu hóa thời gian và tài nguyên trong quá trình truy xuất thông tin.

Lập chỉ mục thường đi kèm với việc tạo danh sách các cặp token-*docID* cho tất cả các token trong bộ sưu tập (collection) *C*. Mục đích của việc này là lưu trữ thông tin

về các token để sử dụng trong các bước tính toán trọng số cho các thuật ngữ (term). Sau đó, các token được sắp xếp lại theo thứ tự bảng chữ cái và chữ số. Nếu có những token trùng nhau, thì thứ tự được xác định bằng *docID*.

Trong quá trình truy xuất thông tin, người ta thường sử dụng hai dạng chỉ mục chính là Chỉ mục chuyển tiếp (Forward Index) và Chỉ mục ngược (Inverted Index). Nhóm sử dụng phương pháp lập chỉ mục ngược.

Chỉ mục ngược là một cấu trúc dữ liệu phổ biến trong truy xuất thông tin. Thay vì lưu trữ thông tin theo tài liệu, chỉ mục ngược lưu trữ thông tin theo thuật ngữ. Mỗi thuật ngữ có một bản ghi trong chỉ mục ngược, trong đó chứa danh sách các tài liệu mà thuật ngữ đó xuất hiện. Khi thực hiện truy vấn, chỉ mục ngược cho phép truy cập nhanh chóng đến các tài liệu chứa thuật ngữ đó.

Quá trình xây dựng chỉ mục:

- Quét tài liệu: Quét qua từng tài liệu trong bộ sưu tập và trích xuất các thuật ngữ từ mỗi tài liệu. Thuật ngữ có thể là từ đơn, cụm từ, hay các đơn vị khác tùy thuộc vào yêu cầu của hệ thống.
- Xây dựng danh sách từ duy nhất: Từ các thuật ngữ trích xuất được từ các tài liệu, xây dựng một danh sách chứa các từ duy nhất.
- Xây dựng chỉ mục: Tạo một cấu trúc dữ liệu để lưu trữ thông tin về từng từ và tài liệu chứa từ đó. Mỗi từ trong danh sách từ duy nhất được ánh xạ đến danh sách tài liệu chứa từ đó.
- Lặp lại cho tất cả các tài liệu.

Ví dụ cho quá trình lập chỉ mục ngược như sau:

Cho tập tài liệu:

d1: Shortlist of gears that are damaged in a fire.

d2: Delivery of silver arrived in a silver truck.

d3: Shortlist of gears that arrived in a truck.

Sau bước tiền xử lý, ta sẽ có được tập tài liệu là:

d1: shortlist gear damaged fire

d2: delivery silver arrived silver truck

d3: shortlist gear arrived truck

Để lập chỉ mục cho tập tài liệu trên, đầu tiên ta lập danh sách các từ duy nhất gồm: *arrived*, *damaged*, *delivery*, *fire*, *gear*, *silver*, *shortlist*, *truck*. Sau đó, lập bảng danh sách các chỉ mục của chúng và ánh xạ:

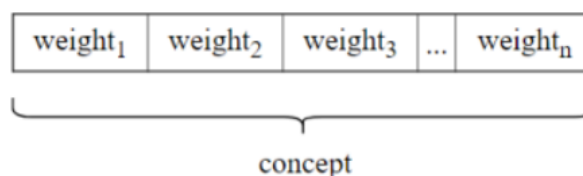
Terms	d1	d2	d3
<i>arrived</i>	0	1	1
<i>damaged</i>	1	0	0
<i>delivery</i>	0	1	0
<i>fire</i>	1	0	0
<i>gear</i>	1	0	1
<i>silver</i>	0	2	0
<i>shortlist</i>	1	0	1
<i>truck</i>	0	1	1

2.4. Truy xuất chỉ mục

Trong quá trình truy xuất chỉ mục, ta quan tâm tới hai yếu tố: Biểu diễn tài liệu bằng vector (Concept vectors) và trọng số (weight) cho các vectors đó.

2.4.1. Concept vector

Là hình thức mà ta dùng để biểu diễn tài liệu (document) và truy vấn (query), việc truy xuất chỉ mục cũng sẽ được thực thi dựa trên các concept vectors này.



Hình 2.12. Mô tả concept vector

Trong đó:

- Mỗi concept vector biểu diễn một chiều trong không gian
- k concept biểu diễn một không gian nhiều chiều
- Mỗi chiều của vector được định nghĩa trọng số (weight)

Sử dụng ví dụ phía trên, ta có thể biểu diễn tài liệu dưới dạng vector sau:

$$\vec{d}_1(0, 1, 0, 1, 1, 0, 1, 0)$$

$$\vec{d}_2(1, 0, 1, 0, 0, 2, 0, 1)$$

$$\vec{d}_3(1, 0, 0, 0, 1, 0, 1, 1)$$

Các tài liệu được đã được biểu diễn dưới dạng vector đếm.

2.4.2. Trọng số (weight)

Cách xác định và tính weights cho vector là hết sức quan trọng, ảnh hưởng đến độ chính xác của các thuật toán xếp hạng. Việc các từ có trọng số khác nhau là do không phải các từ đều có sự quan trọng giống nhau, sử dụng số lần xuất hiện của các từ làm vector không phải là một cách tối ưu. Ở phương diện các documents, một vài từ có thể mang nhiều thông tin hơn các từ còn lại.

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), trong full-text search nó được gọi là stopwords.

Đối với term frequency (*TF*), 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.

Có nhiều kỹ thuật tính trọng số: TF, IDF, TF-IDF,... Nhóm chọn kỹ thuật TF-IDF.

2.4.2.1. Term Frequency -TF (Tần số xuất hiện của Term)

Một document d có term t xuất hiện nhiều lần, mà term t lại xuất hiện trong query q , tức là document d này ít nhiều có liên quan với query q hơn là những document không có hoặc có term t xuất hiện với số lần xuất hiện ít hơn document d . Do đó, những document như vậy nên có độ liên quan cao hơn những document khác.

Để làm được việc này, chúng ta cần gán trọng số lớn cho những term xuất hiện nhiều lần trong một document bằng số lần xuất hiện (hay tần suất) của những term đó trong document đó. Gọi $f(t, d)$ là số lần xuất hiện của term t trong document d , $tf(t, d)$ sẽ được tính như sau:

$$tf(t, d) = f(t, d)$$

Nhưng đây lại là một ý tưởng rất tệ, giả sử nếu tài liệu $d1$ có 20 từ “chair” và tài liệu $d2$ có 2 từ “chair” thì không có nghĩa là $d1$ sẽ liên quan gấp 10 lần $d2$ với ví dụ trên. Khi biểu diễn mức độ liên quan bằng tần số xuất hiện của một từ thì các từ có tần số xuất hiện nhiều lần sẽ tác động rất lớn lên dữ liệu. Do đó, cần phải điều chỉnh lại sao cho độ liên quan $d1$ vẫn cao hơn $d2$ bằng các hàm tính toán không phải tuyến tính, ta có thể điều chỉnh lại công thức tính $tf(t, d)$ như sau:

$$tf(t, d) = \begin{cases} 1 + \log(f(t, d)), & f(t, d) < 0 \\ 0, & f(t, d) \geq 0 \end{cases}$$

- $tf(t, d) = 1 + \log(f(t, d))$ để công thức không còn tuyến tính và cộng thêm 1 để kết quả không xác định không xảy ra khi $f(t, d) = 0$
- Với công thức trên, miền giá trị của $tf(t, d)$ sẽ là $[0; +\infty)$

Quay lại với ví dụ trên, ta có thể lập bảng tf của các term xuất hiện ở các tài liệu cùng với truy vấn như sau:

<i>Terms</i>	<i>Q</i>	<i>d1</i>	<i>d2</i>	<i>d3</i>
<i>arrived</i>	0	0	1	1
<i>damaged</i>	0	1	0	0
<i>delivery</i>	0	0	1	0
<i>fire</i>	0	1	0	0

<i>gear</i>	1	1	0	1
<i>silver</i>	1	0	2	0
<i>shortlist</i>	0	1	0	1
<i>truck</i>	1	0	1	1

2.4.2.2. Inverse Document Frequency – IDF (Nghịch đảo tần suất tài liệu)

Đặc điểm của tf là chỉ quan tâm đến tần suất của một term trong một document chứ không quan tâm đó là term gì. Ví dụ ta có 1000 tài liệu cần truy xuất, term “*chair*” xuất hiện trong cả 1000 tài liệu đó thì khi truy xuất, term “*chair*” trong query “*buying chair*” sẽ không cung cấp được thông tin gì vào việc xếp hạng các tài liệu liên quan trả về, và việc tài liệu chứa nhiều từ “*chair*” cũng không có nghĩa độ liên quan của những tài liệu này cao hơn những tài liệu khác chứa ít từ “*chair*” hơn. Bởi vì term “*chair*” nằm trong tất cả tài liệu, nên chỉ có term “*buying*” góp phần vào việc xếp hạng tài liệu. Chỉ sử dụng giá trị tf sẽ làm trọng số mất đi hiệu quả truy xuất trong nhiều trường hợp.

Do đó idf ra đời với ý tưởng chính là giảm sự tác động của các term phổ biến: các term xuất hiện trong nhiều document thì sẽ ít quan trọng hơn những term xuất hiện trong ít document. Nói cách khác, những term hiếm nên có giá trị cao hơn những term phổ biến.

Công thức idf (inverse document frequency, tạm dịch: tần suất nghịch đảo của document) của term t được định nghĩa bởi:

$$idf(t, d) = \log \left(\frac{N}{df(t)} \right)$$

Trong đó:

- N là tổng số tài liệu truy xuất
- $df(t)$ là số lượng tài liệu chứa term t trong N tài liệu

Xét tiếp ví dụ đã nêu trên, ta lập được bảng idf của các term như sau:

Terms	N/df(i)	idf
--------------	----------------	------------

<i>arrived</i>	$3/2 = 1.5$	0.1761
<i>damaged</i>	$3/1 = 3$	0.4771
<i>delivery</i>	$3/1 = 3$	0.4771
<i>fire</i>	$3/1 = 3$	0.4771
<i>gear</i>	$3/2 = 1.5$	0.1761
<i>silver</i>	$3/1 = 3$	0.4771
<i>shortlist</i>	$3/2 = 1.5$	0.1761
<i>truck</i>	$3/2 = 1.5$	0.1761

2.4.2.3. TF-IDF

Để có được giá trị $tf - idf$ được tích hợp bởi hai giá trị tf và idf , ta nhân $tf(t, q)$ với $idf(t)$.

Áp dụng lên ví dụ ta có bảng giá trị $tf - idf$ như sau:

Terms	Q	d1	d2	d3
<i>arrived</i>	0	0	0.1761	0.1761
<i>damaged</i>	0	0.4771	0	0
<i>delivery</i>	0	0	0.4771	0
<i>fire</i>	0	0.4771	0	0
<i>gear</i>	0.1761	0.1761	0	0.1761
<i>silver</i>	0.4771	0	0.9542	0
<i>shortlist</i>	0	0.1761	0	0.1761
<i>truck</i>	0.1761	0	0.1761	0.1761

Sau khi đã tính được trọng số, ta biểu diễn các tài liệu và truy vấn dưới dạng vector trọng số như sau:

$$\vec{d_1}(0, 0.4771, 0, 0.4771, 0.1761, 0, 0.1761, 0)$$

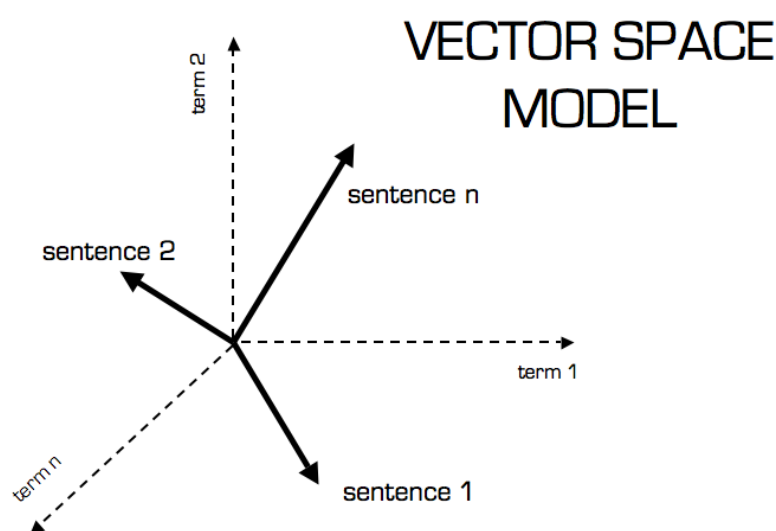
$$\vec{d_2}(0.1761, 0, 0.4771, 0, 0, 0.9542, 0, 0.1761)$$

$$\vec{d_3}(0.1761, 0, 0, 0, 0.1761, 0, 0.1761, 0.1761)$$

$$\vec{q}(0, 0, 0, 0, 0.1761, 0.4771, 0, 0.1761)$$

2.5. Tính độ tương đồng và xếp hạng

Khi các tài liệu và truy vấn đã được biểu diễn thành các vector hay các point (điểm, cả vector và điểm đều nằm trong không gian $|V|$ chiều, $|V|$ là kích thước của vocabulary V hay số lượng term trong V). Một vector được hình thành bởi sự liên kết của hai điểm trong không gian, chúng ta chỉ mới có một điểm, vậy thì điểm còn lại để tạo nên vector là điểm gì, điểm đó nằm ở đâu? Câu trả lời là gốc tọa độ $O(0, 0, \dots, |V|)$ như chúng ta có thể thấy rõ ở hình sau.



Hình 2.13. Các tài liệu và truy vấn sau khi được biểu diễn thành vector

Sau khi đã gán trọng số cho các vector document và vector query xong, bước tiếp theo là tính độ liên quan giữa các document với query. Vậy chúng ta phải xếp hạng các document như thế nào với query trong không gian $|V|$ chiều này?

Câu trả lời là, vì chúng ta đã biểu diễn các document và query trong không gian $|V|$ chiều này, nên chúng ta có thể xếp hạng các document dựa theo proximity (tạm dịch: sự lân cận) với query trong không gian này, và proximity = similarity (tạm dịch: sự giống nhau) giữa những vector.

Bây giờ ta sẽ tiến hành định lượng proximity giữa các vector sử dụng Cosine Similarity. Tuy nhiên trước khi tính toán bằng Cosine, ta phải chuẩn hoá các vector

tài liệu và vector truy vấn. Bằng cách chuẩn hóa các vector document từ trước, chúng ta đã giảm một lượng đáng kể thời gian và chi phí tính toán khi người dùng truy vấn.

Công thức chuẩn hóa:

Cho vector tài liệu: $|\vec{d}_i| = \sqrt{\sum_i w_{i,j}^2}$ và vector truy vấn: $|\vec{q}| = \sqrt{\sum_i w_{q,j}^2}$

Quay lại ví dụ trên, ta có kết quả chuẩn hoá của từng vector trọng số là:

$$|\vec{d}_1| = 0.7192$$

$$|\vec{d}_2| = 1.0955$$

$$|\vec{d}_3| = 0.3522$$

$$|\vec{q}| = 0.5382$$

Sau khi đã chuẩn hoá các vector, để kiểm tra độ tương đồng giữa các tài liệu với truy vấn, ta sử dụng Công thức Cosine:

$$\text{Cosine}(\vec{q}, \vec{d}_i) = \frac{\vec{q} * \vec{d}_i}{|\vec{q}| * |\vec{d}_i|}$$

Và với ví dụ trên, ta có thể tính cosin giữa vector truy vấn và từng vector tài liệu:

$$\text{Cosine}(\vec{q}, \vec{d}_1) = 0.081$$

$$\text{Cosine}(\vec{q}, \vec{d}_2) = 0.8246$$

$$\text{Cosine}(\vec{q}, \vec{d}_3) = 0.3271$$

Từ đó, xếp hạng độ tương đồng là: Vì tài liệu thứ 2 (d2) có độ tương đồng Cosine cao nhất nên sẽ đứng vị trí rank 1, giảm dần với tài liệu thứ 3 (d3) và 1 (d1).

Hạng 1	Hạng 2	Hạng 3
Doc 2 (0.8246)	Doc 3 (0.3271)	Doc 4 (0.0801)

CHƯƠNG 3. MÔ HÌNH LATEN SEMANTIC INDEXING

3.1. Giới thiệu mô hình

3.1.1. Những hạn chế của mô hình Vector Space

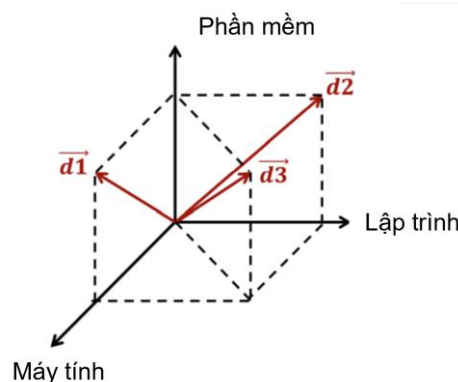
Mặc dù đạt được độ chính xác tương đối cao, mô hình Vector Space còn tồn tại một số hạn chế trong cả việc biểu diễn và so khớp giữa tài liệu và truy vấn.

Thứ nhất, VSM Không đảm bảo độc lập ngữ nghĩa giữa các tài liệu. Mỗi tài liệu có thể mang nhiều ngữ nghĩa khác nhau. Do đó, mỗi vector biểu diễn tài liệu sẽ thuộc đồng thời rất nhiều chiều. VSM không thể đảm bảo được sự độc lập về ngữ nghĩa giữa các tài liệu.

Ví dụ, với các tài liệu như sau:

- Doc1: phần_mềm máy_tính
- Doc2: lập_trình phần_mềm
- Doc3: lập_trình phần_mềm máy_tính

Từ các tài liệu trên, ta có tập từ khóa: $\{phần\ mềm, máy\ tính, lập\ trình\}$. Xét không gian vector tương ứng với tập từ khóa, ta sẽ có hình minh họa ba vector với hệ trục tọa độ tương ứng như bên dưới.



Hình 3.1. Minh họa các vector tài liệu trong không gian hệ trục tọa độ

Qua cách biểu diễn của VSM, ta thấy mỗi vector biểu diễn tài liệu đều thuộc đồng thời nhiều chiều, mỗi chiều có một nghĩa khác nhau. Do đó, mô hình không đảm bảo được sự độc lập về ngữ nghĩa của mỗi tài liệu.

Thứ hai, VSM chỉ dựa trên tần số xuất hiện của các từ khóa và số lượng tài liệu chứa từ khóa để tính trọng số của các chiều. Đây đều là các đại lượng dương. Do đó, *các vector biểu diễn tài liệu chỉ nằm trong phần dương của không gian vector*.

Thứ ba, với các tài liệu có chứa rất nhiều từ (tác phẩm văn học, các công trình nghiên cứu,...), các vector biểu diễn tài liệu sẽ có *số lượng chiều rất lớn*. Điều này dẫn đến tốn nhiều tài nguyên và thời gian xử lý.

Thứ tư, khi so khớp VSM chỉ dựa vào từ khóa, nếu tài liệu và truy vấn không có bất cứ từ khóa chung nào thì độ tương đồng bằng 0. Mô hình chưa khai thác được các đặc trưng về ngữ nghĩa của từ.

Với những hạn chế như phân tích trên, có thể nhận thấy nếu chỉ truy vấn thông tin với mục tiêu là tìm những tài liệu có chứa từ khóa trùng khớp với câu truy vấn, mô hình Vector Space là một lựa chọn hiệu quả. Tuy nhiên, nếu cần thực hiện truy vấn thông tin dựa trên ngữ nghĩa hay nội dung, mô hình này vẫn chưa thể đáp ứng tốt yêu cầu.

3.1.2. Giới thiệu mô hình LSI

Dựa trên ý tưởng khắc phục những hạn chế của mô hình Vector Space, mô hình LSI được đề xuất. Trước hết, nó cũng mang những đặc điểm của mô hình Vector Space như xem mỗi truy vấn và tài liệu là một vector, xếp hạng kết quả truy vấn dựa trên độ tương đồng cosine.

Mô hình LSI truy xuất thông tin dựa trên phân tích ngữ nghĩa tiềm ẩn. Mô hình sẽ phân tích ma trận term-document của các văn bản để tìm ra các ngữ nghĩa tiềm ẩn, mối quan hệ giữa các từ ngữ, các tài liệu trong không gian vector. Các ngữ nghĩa tiềm ẩn ở đây có thể hiểu là các chủ đề hay đặc tính của văn bản.

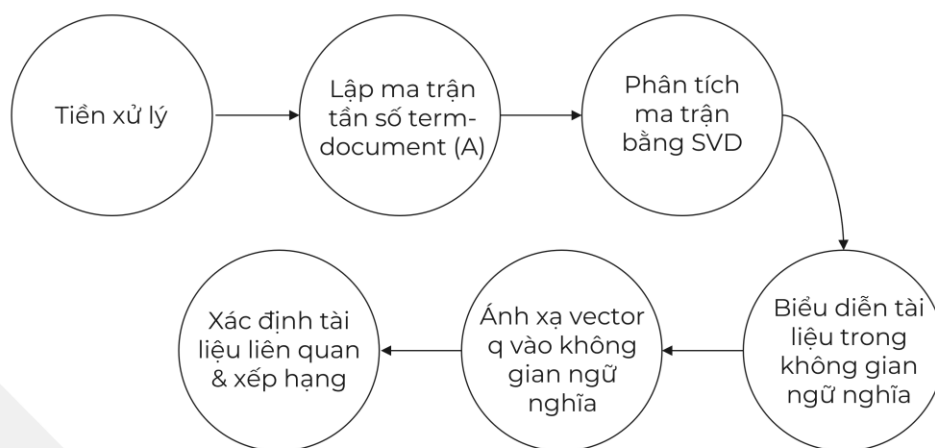
Điểm cốt lõi trong cơ chế giải quyết vấn đề của mô hình LSI là sử dụng thuật toán phân rã ma trận (SVD) giúp làm giảm kích thước của ma trận tần số term – document. Mô hình LSI xem mỗi tài liệu là một vector, sau khi thực hiện phân tích thành giá trị riêng, chỉ giữ lại k ngữ nghĩa quan trọng nhất. Điều này giúp mô hình cải thiện đáng kể kết quả tìm kiếm thông tin nhờ loại bỏ các từ không quan trọng và xác định các khái niệm liên quan đến các tài liệu.

3.1.3. Ý tưởng

Mô hình LSI được xây dựng dựa trên các cơ sở ý tưởng:

- i. Xây dựng một mô hình biểu diễn có không gian biểu diễn có các chiều độc lập nhau
- ii. Giảm số chiều của các vector biểu diễn tài liệu và câu truy vấn trong không gian ngữ nghĩa mà vẫn thu được kết quả xấp xỉ
- iii. Xây dựng không gian ngữ nghĩa tiềm ẩn và biểu diễn các tài liệu và câu truy vấn trong không gian này.

Quá trình thực hiện truy xuất thông tin với mô hình LSI gồm các bước thực hiện như sau:



Hình 3.2. Các bước thực hiện truy xuất thông tin với mô hình LSI

1) Tiền xử lý dữ liệu

Tiến hành thực hiện một số thao tác tiền xử lý dữ liệu ban đầu để thu được tập term.

2) *Lập ma trận tần số term-document*

Lập bảng thống kê tần số xuất hiện của mỗi term trong các tài liệu, dựa vào bảng này để lập ma trận tần số term-document.

3) *Phân tích ma trận term-document bằng thuật toán phân rã SVD*

Áp dụng thuật toán phân rã SVD để phân tích ma trận term-document thành tích của 3 ma trận thừa số. Từ đây, thu được không gian ngữ nghĩa tiềm ẩn.

4) *Biểu diễn tài liệu trong không gian ngữ nghĩa thu được*

Xác định ma trận từ khóa và ma trận tài liệu từ 3 ma trận được phân rã. Từ đó xác định được các vector biểu diễn các từ khóa và các vector biểu diễn các tài liệu trong không gian ngữ nghĩa tiềm ẩn.

5) *Ánh xạ truy vấn vào không gian ngữ nghĩa*

Tương tự như với tài liệu, ta tiền xử lý câu truy vấn và chuyển về dạng vector tần số xuất hiện của các term, ánh xạ vector vào trong không gian ngữ nghĩa tiềm ẩn.

6) *Xác định các tài liệu liên quan và xếp hạng*

Tính độ tương đồng giữa câu truy vấn và các tài liệu, sau đó xếp hạng kết quả truy vấn dựa trên giá trị độ đo tương đồng giảm dần. Tìm trong không gian ngữ nghĩa, các vector tài liệu nào gần với vector truy vấn dựa trên kết quả xếp hạng.

3.2. Tiền xử lý dữ liệu

Các thao tác tiền xử lý được thực hiện tương tự như mô hình Vector Space (được trình bày ở mục 2.2) với các bước:

- Tách từ
- Loại bỏ dấu câu

- Chuyển đổi chữ thường
- Loại bỏ stopwords
- Lấy gốc từ (Stemming và Lemmatization)

3.3. Lập chỉ mục

3.3.1. Lập ma trận tần số term-document

Ma trận term-document được lập bằng cách thống kê tần số xuất hiện của mỗi term trong các tài liệu. Ma trận này có kí hiệu là A , và có các đặc điểm như sau:

- Có kích thước $t \times d$ (t là số lượng term có trong tập term T và d là số lượng tài liệu có trong tập tài liệu D)
- Mỗi dòng của ma trận biểu diễn một term trong tập term:

$$T = \{t_1, t_2, t_3, \dots, t_t\}$$

- Mỗi cột của ma trận biểu diễn một tài liệu trong tập tài liệu:

$$D = \{d_1, d_2, d_3, \dots, d_t\}$$

- Mỗi phần tử a_{ij} ($i = \overline{1, t}, j = \overline{1, d}$) của ma trận là giá trị tần số xuất hiện mà term ở dòng đó xuất hiện trong tài liệu được biểu diễn tại cột đó. Tuy nhiên, giá trị phần tử ở đây cũng có thể tùy thuộc vào cách chọn trọng số term.
- Ma trận có thể biểu diễn dưới dạng như sau:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{t1} & \dots & a_{td} \end{pmatrix}$$

Ma trận này là cơ sở quan trọng của mô hình LSI, nếu thống kê không đúng có thể ảnh hưởng đến toàn bộ các bước thực hiện. Để lập ma trận term – document A , chúng ta thực hiện hai thao tác:

- Lập bảng thống kê tần số xuất hiện của mỗi term trong các tài liệu bằng cách đếm số lần xuất hiện của mỗi term trong từng tài liệu của tập tài liệu.

- Từ bảng thống kê tần số vừa lập, chúng ta sẽ lập được ma trận tần số term – document A .

Chẳng hạn xét ba tài liệu đã được đề cập ở ví dụ trên:

- Doc1: {phần_mềm, máy_tính}
- Doc2: {lập_trình, phần_mềm}
- Doc3: {lập_trình, phần_mềm, máy_tính}

Từ các tài liệu trên, ta có được tập term và tập tài liệu như sau:

$$T = \{\text{máy_tính}, \text{lập_trình}, \text{phần_mềm}\}$$

$$D = \{\text{Doc1}, \text{Doc2}, \text{Doc3}\}$$

Thống kê tần số xuất hiện của các term trong các tài liệu, ta lập được bảng thống kê như sau:

	Doc1	Doc2	Doc3
máy_tính	1	0	1
lập_trình	0	1	1
phần_mềm	1	1	1

Như vậy, ma trận term – document A là:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Trong ma trận này:

- Ba dòng của ma trận A là vector biểu diễn của ba term “máy_tính”, “lập_trình” và “phần_mềm” thuộc tập term.

$$A = \begin{pmatrix} t_1 = (1,0,1) \\ t_2 = (0,1,1) \\ t_3 = (1,1,1) \end{pmatrix}$$

- Ba cột của ma trận A là vector biểu diễn ba tài liệu Doc1, Doc2 và Doc3 thuộc tập tài liệu

$$A = \left(d_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad d_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad d_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right)$$

3.3.2. Phân rã ma trận term-document bằng thuật toán SVD

3.3.2.1. Thuật toán phân rã ma trận SVD

3.3.2.1.1. Trị riêng và vector riêng

Cho một ma trận vuông $A \in \mathbb{R}^{n \times n}$, nếu tồn tại một số vô hướng λ và vector $x \neq \vec{0}$ và $x \in \mathbb{R}^n$ thoả mãn hệ thức: $Ax = \lambda x$.

Ở đây:

- λ là một giá trị riêng (gọi tắt là trị riêng) của ma trận A
- Vector x được gọi là vector riêng (hay vector đặc trưng) của ma trận A tương ứng với trị riêng λ .

Các bước xác định trị riêng và vector riêng của ma trận vuông:

1) Tìm các trị riêng của ma trận A

Giải phương trình đặc trưng của ma trận A :

$$P_A(\lambda) = 0 \Leftrightarrow \det(A - \lambda I) = 0$$

Nghiệm của phương trình đặc trưng chính là các trị riêng λ_i của ma trận A .

2) Tìm các vector riêng của ma trận A

Với mỗi trị riêng của ma trận A , giải hệ phương trình thuần nhất sau:

$$(A - \lambda_i I)x = 0$$

Biết I là ma trận đơn vị cùng cấp với ma trận A . Nghiệm không tầm thường của hệ phương trình này là các vector riêng v_j của ma trận A ứng với trị riêng λ_i .

3.3.2.1.2. Ý tưởng thuật toán

Ma trận A kích thước $t \times d$. Giả sử σ_i là các trị riêng (với $\sigma_1^2 \geq \sigma_2^2 \geq \dots \sigma_d^2$) ứng với các vector riêng x_1, x_2, \dots, x_n .

Ta có:

- $U = \{x_1, x_2, \dots, x_d\}$ trong đó x_i là các vector cột
- $y_i = \frac{1}{\sigma_i} Ax_i$
- $S = \{y_1, y_2, \dots, y_d\}$ trong đó y_i là các vector dòng
- Σ là ma trận đường chéo có các giá trị trên đường chéo chính là $\sigma_1, \sigma_2, \dots, \sigma_d$

Khi đó, mọi ma trận A đều có thể được thành tích của ba ma trận thừa số S, Σ, U^T có số chiều nhỏ hơn ma trận A theo công thức sau:

$$A = S\Sigma U^T$$

Cách phân tích như trên được gọi là một phép phân rã ma trận SVD (Singular Value Decomposition). Trong đó:

- S là một ma trận trực giao có kích thước $t \times m$
- Σ là một ma trận đường chéo có kích thước $m \times m$
- U là một ma trận trực giao có kích thước $m \times d$

3.3.2.1.3. Các bước thực hiện

Thuật toán SVD gồm các bước thực hiện chính:

1) Xác định ma trận Σ

- Giải phương trình đặc trưng của ma trận A để tìm các trị riêng.
- Với các trị riêng vừa tìm được, lập ma trận Σ bằng cách xếp các trị riêng nằm trên đường chéo chính theo thứ tự giảm dần độ lớn từ trên xuống dưới.

2) Xác định ma trận U

Với mỗi trị riêng tìm được, xác định vector riêng ứng với trị riêng đó.

Lập ma trận U bằng cách xếp các vector riêng vừa tìm được thành các cột của thành một ma trận.

3) Xác định ma trận S

Với mỗi trị riêng và vector riêng tương ứng, xác định các ma trận cột là các cột tương ứng của ma trận S theo công thức:

$$s_i = \frac{1}{\lambda_i} A u_i$$

Với $i = \overline{1, m}$ và m là số trị riêng của ma trận A

3.3.2.2. Phân rã ma trận term-document bằng thuật toán SVD

3.3.2.2.1. Các ma trận thừa số trong phép phân rã

Ta tiến hành sử dụng thuật toán phân rã ma trận SVD để phân tích ma trận term – document A vừa tìm được ở bước trên thành tích của ba ma trận thừa số S , Σ , U^T có số chiều nhỏ hơn ma trận A theo công thức sau:

$$A = S \Sigma U^T$$

Trong đó:

- S là ma trận trực giao có kích thước $t \times m$, với t là số lượng term và m là số lượng ngữ nghĩa (chủ đề), m cũng chính là số chiều của không gian ngữ nghĩa tiềm ẩn
- Σ là ma trận đường chéo có kích thước $m \times m$, có các phần tử trên đường chéo là trị riêng của ma trận A được sắp xếp theo thứ tự từ trên xuống dưới giảm dần độ lớn (nghĩa là phần tử ở góc trái trên cùng của ma trận là giá trị riêng có độ lớn lớn nhất, và phần tử ở góc phải dưới cùng của ma trận là giá trị riêng có độ lớn nhỏ nhất)
- U là ma trận trực giao có kích thước $m \times d$, với d là số tượng tài liệu có trong tập tài liệu

Như vậy, khi nhân ba ma trận này lại, ta sẽ thu được ma trận A ban đầu. Vậy, mục tiêu ở đây là phải xác định được ba ma trận thừa số S , Σ , U^T trong công thức này. Trên cơ sở của việc phân tích, ta sẽ rút trích được mối tương quan ngữ nghĩa tiềm ẩn giữa các tài liệu trong tập tài liệu, hay nói cách khác là mối quan hệ về mặt ngữ nghĩa giữa các tài liệu trong tập tài liệu.

Có thể nhận xét về ý nghĩa của ba ma trận này như sau:

a) *Ma trận S*

Đây là ma trận biểu thị mối quan hệ giữa các term và các ngữ nghĩa tiềm ẩn. Dựa vào ma trận S , ta sẽ trả lời được câu hỏi: “*Term nào thuộc về chủ đề nào?*” Các ngữ nghĩa được đề cập ở đây chính là những ngữ nghĩa tiềm ẩn và tương ứng với các chiều của không gian ngữ nghĩa.

b) *Ma trận Σ*

Đây là ma trận ngữ nghĩa tiềm ẩn, hay nói cách khác là ma trận này chứa đựng các chủ đề được lựa chọn và quan tâm. Ma trận này có thể được xem là không gian ngữ nghĩa tiềm ẩn. Ta cần tìm cách biểu diễn các tài liệu và câu truy vấn trong không gian ngữ nghĩa này, hay nói cách khác là ánh xạ các vector biểu diễn các tài liệu và câu truy vấn vào trong không gian này, để từ đó có thể tính độ tương đồng giữa chúng.

Độ lớn của trị riêng biểu thị mức độ quan trọng của các chiều trong không gian. Do đó, các trị riêng có độ lớn càng lớn thì càng quan trọng, tức là càng được quan tâm nhiều hơn. Ngược lại, các trị riêng có độ lớn càng nhỏ thì càng kém quan trọng. Các trị riêng có độ lớn nhỏ có thể được xem như là nhiễu và có thể được loại bỏ bằng cách cắt giảm ma trận.

c) *Ma trận U^T*

Đây là ma trận biểu thị mối quan hệ giữa các tài liệu và các ngữ nghĩa tiềm ẩn. Dựa vào ma trận U , chúng ta sẽ trả lời được câu hỏi: “*Tài liệu nào thuộc về chủ đề nào?*” Ta không sử dụng trực tiếp ma trận U , mà sẽ sử dụng ma

trận U^T là ma trận chuyển vị của ma trận U , để đảm bảo yêu cầu về số chiều của các ma trận thừa số trong phép nhân ma trận.

3.3.2.2.2. Minh họa các bước thực hiện phân rã ma trận term-document

Dựa trên như các bước thực hiện của thuật toán SVD, xét ví dụ đã được trình bày ở trên, ta có ma trận A :

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

1) Xác định ma trận Σ

Để tìm trị riêng của ma trận vuông, dựa vào định nghĩa của trị riêng và vector riêng, chúng ta có hệ thức:

$$\begin{aligned} Ax = \lambda x &\Leftrightarrow Ax = \lambda Ix \Leftrightarrow (A - \lambda I)x = 0 \\ \Rightarrow A - \lambda I &= \begin{pmatrix} 1 - \lambda & 0 & 1 \\ 0 & 1 - \lambda & 1 \\ 1 & 1 & 1 - \lambda \end{pmatrix} \end{aligned}$$

Đa thức đặc trưng của ma trận A :

$$\begin{aligned} P_A(\lambda) &= \det(A - \lambda I) = \det \begin{pmatrix} 1 - \lambda & 0 & 1 \\ 0 & 1 - \lambda & 1 \\ 1 & 1 & 1 - \lambda \end{pmatrix} \\ &= (1 - \lambda)^3 - (1 - \lambda) - (1 - \lambda) \\ &= (1 - \lambda)^3 - 2(1 - \lambda) \end{aligned}$$

Như vậy, phương trình đặc trưng của ma trận A đã cho là:

$$\begin{aligned} (1 - \lambda)^3 - 2(1 - \lambda) &= 0 \\ \Leftrightarrow (1 - \lambda)(\lambda^2 - 2\lambda - 1) &= 0 \\ \Leftrightarrow \begin{cases} \lambda_1 = 1 + \sqrt{2} \\ \lambda_2 = 1 \\ \lambda_3 = 1 - \sqrt{2} \end{cases} \end{aligned}$$

Ta được ma trận Σ sẽ có dạng như sau:

$$\Sigma = \begin{pmatrix} 1 + \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \sqrt{2} \end{pmatrix}$$

2) Xác định ma trận U

Chúng ta giải phương trình sau để tìm vector riêng ứng với từng trị riêng:

$$(A - \lambda I)x = 0$$

$$\Leftrightarrow \begin{pmatrix} 1 - \lambda & 0 & 1 \\ 0 & 1 - \lambda & 1 \\ 1 & 1 & 1 - \lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Giải hệ phương trình trên tương ứng với từng trị riêng, chúng ta thu được các vector riêng:

$$u_1 = \begin{pmatrix} x_1 \\ x_1 \\ \sqrt{2}x_1 \end{pmatrix} \quad u_2 = \begin{pmatrix} x_1 \\ -x_1 \\ 0 \end{pmatrix} \quad u_3 = \begin{pmatrix} x_1 \\ x_1 \\ -\sqrt{2}x_1 \end{pmatrix}$$

Lần lượt chọn giá trị x_1 sao cho ba vector u_1, u_2, u_3 đều có độ dài bằng 1. Ta được ma trận U :

$$U = \begin{pmatrix} 0.5 & 0.71 & -0.5 \\ 0.5 & -0.71 & -0.5 \\ 0.71 & 0 & 0.71 \end{pmatrix}$$

$$\Rightarrow U^T = \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \\ -0.5 & -0.5 & 0.71 \end{pmatrix}$$

3) Xác định ma trận S

Với mỗi trị riêng của ma trận A và vector riêng tương ứng, chúng ta sẽ xác định các ma trận cột là các cột tương ứng của ma trận S theo công thức:

$$s_i = \frac{1}{\lambda_i} A u_i$$

- Cột s_1 : $s_1 = \frac{1}{\lambda_1} A u_1 = \frac{1}{1+\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.71 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.71 \end{pmatrix}$

- Cột s_2 : $s_2 = \frac{1}{\lambda_2} Au_2 = \frac{1}{1} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0.71 \\ -0.71 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.71 \\ -0.71 \\ 0 \end{pmatrix}$
- Cột s_3 : $s_3 = \frac{1}{\lambda_3} Au_3 = \frac{1}{1-\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} -0.5 \\ -0.5 \\ 0.71 \end{pmatrix} = \begin{pmatrix} -0.5 \\ -0.5 \\ 0.71 \end{pmatrix}$

Ma trận S tìm được như sau:

$$S = \begin{pmatrix} 0.5 & 0.71 & -0.5 \\ 0.5 & -0.71 & -0.5 \\ 0.71 & 0 & 0.71 \end{pmatrix}$$

Vậy, ma trận term – document A ban đầu được phân rã thành tích của ba ma trận thừa số S , Σ , U^T có số chiều nhỏ hơn ma trận A như sau:

$$\begin{aligned} A &= S\Sigma U^T \\ &= \begin{pmatrix} 0.5 & 0.71 & -0.5 \\ 0.5 & -0.71 & -0.5 \\ 0.71 & 0 & 0.71 \end{pmatrix} \begin{pmatrix} 1 + \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \sqrt{2} \end{pmatrix} \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \\ -0.5 & -0.5 & 0.71 \end{pmatrix} \end{aligned}$$

3.3.3. Biểu diễn tài liệu trong không gian ngữ nghĩa

3.3.3.1. Ma trận từ khóa và ma trận tài liệu

Từ ba ma trận thừa số S , Σ , U^T đã tìm được, ta cần tìm cách xây dựng các vector biểu diễn (ma trận biểu diễn) các từ khóa và các tài liệu trong không gian ngữ nghĩa tiềm ẩn.

Qua phân tích, có thể nhận thấy rằng:

- *Ma trận tích $S\Sigma$:*

Đây là ma trận biểu diễn các từ khóa trong không gian ngữ nghĩa tiềm ẩn. Đặt ma trận này là K :

$$K = S\Sigma$$

Ma trận K là ma trận chứa các vector biểu diễn các term trong không gian ngữ nghĩa tiềm ẩn. Trong đó, mỗi dòng của ma trận K chính là một vector

biểu diễn của một term trong không gian này.

- *Ma trận tích ΣU^T :*

Đây là ma trận biểu diễn các tài liệu trong không gian ngữ nghĩa tiềm ẩn. Đặt ma trận này là D :

$$D = \Sigma U^T$$

Ma trận D là ma trận chứa các vector biểu diễn các tài liệu trong không gian ngữ nghĩa tiềm ẩn. Trong đó, mỗi cột của ma trận D chính là một vector biểu diễn của một tài liệu trong không gian này.

Như vậy, ma trận từ khóa K và ma trận tài liệu D sẽ lần lượt chứa các ảnh ánh xạ của các vector biểu diễn các term và các tài liệu vào trong không gian ngữ nghĩa tiềm ẩn. Việc xác định ma trận term K và ma trận tài liệu D ở bước này tương đương với việc biểu diễn các term và các tài liệu trong không gian ngữ nghĩa tiềm ẩn.

3.3.3.2. Minh họa biểu diễn tài liệu trong không gian ngữ nghĩa

Xét ví dụ minh họa đã trình bày ở phần trước đó.

1) *Xác định ma trận K*

Kích thước của ma trận từ khóa:

$$K_{t \times m} = S_{t \times m} \Sigma_{m \times m}$$

Tìm ma trận K bằng cách áp dụng công thức trên:

$$K = S \Sigma = \begin{pmatrix} 0.5 & 0.71 & -0.5 \\ 0.5 & -0.71 & -0.5 \\ 0.71 & 0 & 0.71 \end{pmatrix} \begin{pmatrix} 1 + \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \sqrt{2} \end{pmatrix}$$

$$\Rightarrow K = \begin{pmatrix} 1.21 & 0.71 & 0.21 \\ 1.21 & -0.71 & 0.21 \\ 1.71 & 0 & -0.29 \end{pmatrix}$$

2) *Xác định ma trận D*

Kích thước của ma trận tài liệu:

$$D_{m \times d} = \Sigma_{m \times m} U_{m \times d}^T$$

Tìm ma trận D bằng cách áp dụng công thức trên:

$$D = \Sigma U^T = \begin{pmatrix} 1 + \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \sqrt{2} \end{pmatrix} \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \\ -0.5 & -0.5 & 0.71 \end{pmatrix}$$
$$\Rightarrow D = \begin{pmatrix} 1.21 & 1.21 & 1.71 \\ 0.71 & -0.71 & 0 \\ 0.21 & 0.21 & -0.29 \end{pmatrix}$$

3.4. Truy xuất chỉ mục

Đối với mô hình LSI, mục tiêu của chúng ta là tìm cách biểu diễn các tài liệu và câu truy vấn trong cùng một không gian ngữ nghĩa, hay nói cách khác là ánh xạ các vector biểu diễn các tài liệu và câu truy vấn vào trong cùng một không gian ngữ nghĩa, để từ đó có thể xác định được các tài liệu có liên quan với câu truy vấn thông qua việc tính độ tương đồng giữa các vector với nhau.

Do đó, mục tiêu của chúng ta trong bước này là biểu diễn câu truy vấn q trong không gian ngữ nghĩa mà chúng ta đã xây dựng ở các bước trước, hay nói cách khác là ánh xạ vector biểu diễn câu truy vấn q vào trong cùng một không gian ngữ nghĩa.

Như vậy, trong bước này, chúng ta sẽ thực hiện thao tác như sau:

- Tiền xử lý câu truy vấn q .
- Chuyển câu truy vấn q về dạng vector thống kê tần số xuất hiện của các term có trong câu truy vấn q .
- Ánh xạ vector biểu diễn câu truy vấn q vào trong không gian ngữ nghĩa tiềm ẩn đã xây dựng ở các bước trước.

Trong đó, hai thao tác đầu tiên sẽ được thực hiện tương tự như chúng ta đã thực hiện với các tài liệu trong các bước trước.

Kết quả thực hiện của bước này sẽ là vector biểu diễn câu truy vấn q trong không gian ngữ nghĩa tiềm ẩn, hay nói cách khác là ánh xạ của vector biểu diễn câu truy vấn vào trong không gian này.

3.4.1. Tiền xử lý câu truy vấn

Tương tự như việc tiền xử lý các tài liệu đã được trình bày ở các bước trước, tùy vào đặc điểm của câu truy vấn, chúng ta sẽ lựa chọn và thực hiện một số thao tác tiền xử lý dữ liệu để thực hiện tiền xử lý câu truy vấn q , chẳng hạn như:

- Chuyển sang chữ thường.
- Loại bỏ các ký tự đặc biệt.
- Tách từ.
- Loại bỏ stopwords.
- Lemmatization.
- ...

Các thao tác này chúng ta sẽ thực hiện tương tự như đã thực hiện đối với các tài liệu. Chẳng hạn, với câu truy vấn q :

q : [máy_tính và lập_trình]/*

Sau khi thực hiện các bước tiền xử lý, câu truy vấn q sẽ trở thành:

q : {"máy_tính", "lập_trình"}

3.4.2. Chuyển câu truy vấn về dạng vector thống kê tần số xuất hiện của các term có trong câu truy vấn

Tương tự như đã thực hiện với các tài liệu, trong bước này, chúng ta sẽ lập bảng thống kê tần số xuất hiện của mỗi term trong câu truy vấn q bằng cách đếm số lần xuất hiện của mỗi term trong câu truy vấn. Tiếp theo, từ bảng thống kê tần số vừa lập, chúng ta sẽ lập được vector biểu diễn câu truy vấn q .

Với câu truy vấn:

q : {"máy_tính", "phần_mềm"}

Chúng ta lập được bảng thống kê như sau:

Bảng 3.1. Bảng thống kê tần số xuất hiện của mỗi term trong câu truy vấn

Term \ Tài liệu	q
máy_tính	1
lập_trình	1
phần_mềm	0

Trên cơ sở bảng thống kê vừa lập, chúng ta sẽ xác định được vector biểu diễn câu truy vấn q . Vector này sẽ thống kê tần số xuất hiện của các term có trong câu truy vấn q .

$$q = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Tiếp theo, chúng ta sẽ ánh xạ vector biểu diễn câu truy vấn q vào trong không gian ngữ nghĩa tiềm ẩn đã xây dựng ở các bước trước.

3.4.3. Ánh xạ vector biểu diễn câu truy vấn vào trong không gian ngữ nghĩa tiềm ẩn

Như đã trình bày, chúng ta có nhận, trong mô hình LSI, ngữ nghĩa của câu văn, đoạn văn là tổng ngữ nghĩa của các từ có trong câu văn, đoạn văn đó. Đối với một câu truy vấn, chúng ta có ngữ nghĩa của câu truy vấn đó chính là tổng ngữ nghĩa của các term có trong câu truy vấn đó. Chúng ta có thể xây dựng công thức như sau:

$$query_meaning = \sum_{word \in query} m_{word}$$

Trong đó, m_{word} là ngữ nghĩa của từ trong câu truy vấn.

Ở các bước trước, chúng ta đã xây dựng được không gian ngữ nghĩa tiềm ẩn và đã biểu diễn được các tài liệu đó trong không gian này, tức là đã xây dựng được các vector biểu diễn các tài liệu trong không gian này. Như vậy, mục tiêu của chúng ta

trong bước này là ánh xạ vector biểu diễn câu truy vấn q vào trong không gian ngữ nghĩa tiềm ẩn chứa các vector biểu diễn các tài liệu đã xây dựng được ở các bước trước.

Xét tập tài liệu $D = \{d1, d2, d3\}$, với:

doc1: {"máy_tính", "phần_mềm"}

doc2: {"phần_mềm", "lập_trình"}

doc3: {"máy_tính", "phần_mềm", "lập_trình"}

Tập term của tập tài liệu D là $T = \{\text{"máy_tính"}, \text{"phần_mềm"}, \text{"lập_trình"}\}$

Tập D có ma trận K như sau:

$$K = \begin{pmatrix} 1.21 & 0.71 & 0.21 \\ 1.21 & -0.71 & 0.21 \\ 1.71 & 0 & -0.29 \end{pmatrix}$$

Với các vector biểu diễn của các term trong không gian ngữ nghĩa tiềm ẩn:

$$k_1 = (1.21 \quad 0.71 \quad 0.21)$$

$$k_2 = (1.21 \quad -0.71 \quad 0.21)$$

$$k_3 = (1.71 \quad 0 \quad -0.29)$$

Trong đó k_1, k_2, k_3 lần lượt là các vector biểu diễn của các term "máy_tính", "phần_mềm", "lập_trình". Chúng ta sẽ khởi tạo tọa độ của vector q_K là:

$$q_K = (0 \quad 0 \quad 0)$$

Chúng ta sẽ tính toán giá trị của q_K theo công thức sau:

$$q_K = \sum_{q_i \in T, q} k_i$$

Công thức trên có thể được phát biểu như sau:

"Vector q_K đại diện cho truy vấn q là tổng của các vector k_i trong không gian ngữ nghĩa tiềm ẩn, với k_i là vector đại diện cho term q_i xuất hiện trong truy vấn q ."

Xét câu truy vấn q : {"máy_tính", "lập_trình"}.

Trong câu truy vấn q có chứa 1 từ “máy_tính”, 1 từ “lập_trình” và không có chứa từ “phần_mềm”.

Do đó, tọa độ của vector q_K sẽ được cộng thêm tọa độ của hai vector biểu diễn hai term “máy_tính” và “lập_trình” trong không gian ngữ nghĩa tiềm ẩn, đó là hai vector k_1 và k_2 .

Khi đó, tọa độ của vector q_K được tính như sau:

$$q_K = k_1 + k_2 = (1.21, -0.71, 0.21) + (1.21, 0.71, 0.21) = (2.42, 0, 0.42)$$

3.4.4. Ánh xạ trực tiếp vector truy vấn vào không gian ngữ nghĩa

Ngoài cách ánh xạ vector truy vấn vào không gian ngữ nghĩa tiềm ẩn được giới thiệu ở phần trên. Chúng ta có thể ánh xạ trực tiếp vector truy vấn q vào không gian ngữ nghĩa mà không cần thông qua ma trận K .

Quá trình này có thể được thực hiện thông qua công thức sau:

$$q_k = \Sigma_k^{-1} S_k^T q$$

Trong đó:

- Σ^k, S^k lần lượt là ma trận cắt giảm của Σ, S (sẽ được giới thiệu ở phần Cắt giảm ma trận).
- q là vector đại diện cho truy vấn, được xây dựng tương tự như trong mô hình Vector Space.
- q_k là vector đại diện cho truy vấn trong không gian ngữ nghĩa tiềm ẩn.

3.5. Tính độ tương đồng và xếp hạng

Ở bước này, chúng ta sẽ thực hiện tương tự như đối với mô hình Vector Space: Tính độ tương đồng giữa câu truy vấn và các tài liệu trong tập tài liệu bằng cách tính độ đo *cosine* giữa vector truy vấn và các vector tài liệu, sau đó xếp hạng kết quả truy vấn dựa trên giá trị độ đo *cosine* giảm dần.

Mục tiêu của chúng ta trong bước này là tìm trong không gian ngữ nghĩa tiềm ẩn, các vector biểu diễn các tài liệu nào có liên quan (“gần”) với vector biểu diễn câu truy vấn q dựa trên kết quả xếp hạng độ đo *cosine*. Các tài liệu tương ứng với các vector đó chính là các tài liệu liên quan cần tìm.

Trong bước này, chúng ta sẽ thực hiện hai thao tác, đó là:

- Tính độ tương đồng giữa câu truy vấn và các tài liệu trong tập tài liệu bằng cách tính độ đo *cosine* giữa các vector biểu diễn các tài liệu và vector biểu diễn câu truy vấn trong không gian ngữ nghĩa tiềm ẩn.
- Xếp hạng kết quả truy vấn dựa trên giá trị độ đo *cosine* giảm dần. Từ kết quả xếp hạng này, chúng ta sẽ tìm được các tài liệu có liên quan với câu truy vấn tùy vào yêu cầu danh sách TOP- k tài liệu liên quan nhất.

3.5.1.1. Tính độ tương đồng giữa câu truy vấn và các tài liệu

Đối với mô hình LSI, độ tương đồng giữa câu truy vấn và các tài liệu chính là độ đo sự tương đồng (sự liên quan) về mặt ngữ nghĩa giữa câu truy vấn và các tài liệu.

Độ tương đồng ngữ nghĩa là khái niệm thể hiện tỷ lệ dựa trên sự giống nhau về nội dung ý nghĩa của tập các tài liệu hoặc các thuật ngữ trong một danh sách các thuật ngữ. Độ tương đồng ngữ nghĩa phản ánh mối quan hệ ngữ nghĩa giữa các câu, các tài liệu văn bản.

Do đó, để tính độ tương đồng giữa câu truy vấn và các tài liệu, chúng ta sẽ tính độ tương đồng giữa các vector biểu diễn các tài liệu và vector biểu diễn câu truy vấn trong không gian ngữ nghĩa tiềm ẩn mà chúng ta đã tìm được trong các bước trên. Đó chính là các vector tài liệu $doc1$, $doc2$, $doc3$ và vector q_k biểu diễn câu truy vấn trong không gian ngữ nghĩa tiềm ẩn.

Chúng ta sẽ tính độ tương đồng giữa câu truy vấn và các tài liệu trong tập tài liệu bằng cách tính độ đo cosine giữa các tài liệu và vector biểu diễn câu truy vấn trong không gian ngữ nghĩa tiềm ẩn theo công thức:

$$\cosine(q_k, doc) = \frac{q_k doc}{|q_k||doc|}$$

Trong đó:

- q_k là vector biểu diễn câu truy vấn trong không gian ngữ nghĩa tiềm ẩn
- doc là vector biểu diễn tài liệu trong không gian ngữ nghĩa tiềm ẩn
- $|q_k|$ và $|doc|$ lần lượt là độ dài của các vector q_k và doc

Chúng ta sẽ thực hiện tính toán giá trị độ đo cosine giữa vector q_k và các vector $doc1, doc2, doc3$ như sau:

$$\begin{aligned}\cosine(q_k, doc1) &= \frac{q_k doc1}{|q_k||doc1|} \\ &= \frac{2.42 \times 1.21 + 0 \times 0.71 + 0.42 \times 0.21}{\sqrt{2.42^2 + 0^2 + 0.42^2} \sqrt{1.21^2 + 0.71^2 + 0.21^2}} \\ &\simeq 0.8657\end{aligned}$$

$$\begin{aligned}\cosine(q_k, doc2) &= \frac{q_k doc2}{|q_k||doc2|} \\ &= \frac{2.42 \times 1.21 + 0 \times (-0.71) + 0.42 \times 0.21}{\sqrt{2.42^2 + 0^2 + 0.42^2} \sqrt{1.21^2 + (-0.71)^2 + 0.21^2}} \\ &\simeq 0.8657\end{aligned}$$

$$\begin{aligned}\cosine(q_k, doc3) &= \frac{q_k doc3}{|q_k||doc3|} \\ &= \frac{2.42 \times 1.71 + 0 \times 0 + 0.42 \times (-0.29)}{\sqrt{2.42^2 + 0^2 + 0.42^2} \sqrt{1.71^2 + 0^2 + (-0.29)^2}} \\ &\simeq 0.9428\end{aligned}$$

Như vậy, chúng ta đã tính được độ tương đồng giữa câu truy vấn và các tài liệu thông qua độ đo cosine như sau:

Độ tương đồng giữa câu truy vấn và các tài liệu thông qua độ đo cosine:

$$\cosine(q_k, doc1) \simeq 0.8657$$

$$\cosine(q_k, doc2) \simeq 0.8657$$

$$\text{cosine}(q_k, \text{doc3}) \approx 0.9428$$

Bên cạnh độ đo cosine, chúng ta có thể tính toán độ tương đồng giữa các vector biểu diễn câu truy vấn và các tài liệu thông qua một số độ đo khác như độ đo khoảng cách Euclide, khoảng cách Manhattan,...

3.5.1.2. Xếp hạng kết quả truy vấn

Theo tính chất của hàm số cosine, chúng ta có nhận xét: Nếu $\text{cosine}\varphi$ càng lớn thì φ càng nhỏ.

Điều này có nghĩa nếu giá trị độ đo *cosine* càng lớn thì góc giữa hai vector sẽ càng nhỏ, tức là hai vector đó càng “gần” nhau và càng liên quan với nhau. Ngược lại, nếu là giá trị độ đo *cosine* càng nhỏ thì góc giữa hai vector sẽ càng lớn, tức là hai vector đó càng “xa” nhau và càng ít liên quan với nhau.

Chúng ta sẽ tiến hành xếp hạng kết quả truy vấn dựa trên các giá trị độ đo *cosine* vừa tính được theo thứ tự giảm dần. Khi đó, các giá trị độ đo *cosine* càng lớn sẽ có thứ hạng càng cao.

Do đó, chúng ta sẽ có bảng xếp hạng kết quả truy vấn thông qua độ đo *cosine* như sau:

Bảng 3.2. Bảng xếp hạng kết quả truy vấn thông qua độ đo cosine

Thứ hạng	Tài liệu	Giá trị độ đo <i>cosine</i>
1	doc3	0.9428
2	doc1	0.8657
3	doc2	0.8657

Thông qua bảng xếp hạng trên, chúng ta có thứ tự kết quả truy vấn theo thứ tự giảm dần độ liên quan là:

$$\text{doc3} > \text{doc1} > \text{doc2}$$

Tùy theo yêu cầu của bài toán, chúng ta sẽ lựa chọn danh sách TOP-k tài liệu liên quan nhất, tức là k tài liệu đứng đầu và có thứ hạng cao nhất trên bảng xếp hạng.

Chẳng hạn, trong ví dụ này, chúng ta có TOP-1 tài liệu liên quan là *doc3*. Như vậy, chúng ta đã hoàn tất ví dụ minh họa đã đưa ra.

3.6. Cắt giảm ma trận

3.6.1. Đặt vấn đề

Việc giảm chiều dữ liệu tất yếu dẫn đến sự mất mát thông tin. Do đó, ta chỉ xem xét loại bỏ những thông tin ít quan trọng, hoặc được xem là nhiễu. Việc cắt giảm ma trận trong mô hình LSI hoàn toàn đáp ứng được yêu cầu này vì chỉ loại bỏ những ngữ nghĩa kém quan trọng, ít được quan tâm, ứng với các trị riêng có độ lớn nhỏ, tức là những thông tin không cần thiết.

Gọi k là số lượng ngữ nghĩa (hay chủ đề) cần giữ lại sau khi thực hiện cắt giảm ma trận ($k \ll m$), khi đó, k cũng chính là số chiều được giữ lại của không gian ngữ nghĩa tiềm ẩn. Ta giữ lại k chủ đề quan trọng nhất bằng cách chọn và giữ lại k trị riêng lớn nhất, cũng đồng nghĩa với việc không gian ngữ nghĩa tiềm ẩn chỉ giữ lại k chiều quan trọng nhất.

Số chiều của không gian ngữ nghĩa cũng chính là số trị riêng của ma trận term – document A , và đồng thời cũng là số chiều của ma trận Σ . Khi đó, ma trận Σ chỉ còn chứa k trị riêng có độ lớn lớn nhất, do đó ma trận Σ bây giờ sẽ có kích thước là $k \times k$.

Trong trường hợp $k = m$ nghĩa là không thực hiện cắt giảm ma trận. Ngoài ra, giá trị k được chọn nên đủ lớn để phù hợp với mọi đặc tính của dữ liệu, nhưng cũng phải đủ nhỏ để lọc ra các chi tiết không phù hợp hay các chi tiết không quan trọng.

3.6.2. Các bước thực hiện cắt giảm

Khi cắt giảm, ma trận sẽ giữ lại k trị riêng có độ lớn lớn nhất trong ma trận Σ , đồng thời sẽ giữ lại k vector riêng ứng với k trị riêng này. Theo thuật toán SVD, ma trận term – document A ban đầu được phân tích thành tích của ba ma trận thừa số như sau:

$$A_{t \times d} = S_{t \times m} \Sigma_{m \times m} U_{m \times d}^T$$

Với m là số lượng chủ đề, cũng chính là số chiều của không gian ngữ nghĩa tiềm ẩn.

Sau khi cắt giảm chỉ giữ lại k chiều, ma trận Σ bây giờ sẽ có kích thước là $k \times k$. Do đó, chúng ta cần phải xác định lại các kích thước của các ma trận S và U^T sao cho vẫn đảm bảo được tính chất của phép nhân ma trận với ma trận. Trong đó:

- Ma trận S phải có số cột bằng với số dòng của ma trận Σ
- Ma trận U^T có số dòng bằng với số cột của ma trận Σ

$$S_{t \times ?} \Sigma_{k \times k} U_{? \times d}^T$$

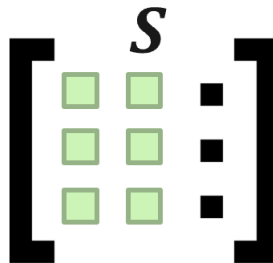
Do đó, chúng ta sẽ có kích thước của hai ma trận S mới và U^T mới là:

$$S_{t \times k} \Sigma_{k \times k} U_{k \times d}^T$$

Đối với mỗi ma trận thừa số trong công thức của thuật toán SVD, chúng ta sẽ thực hiện như sau:

1. Ma trận S

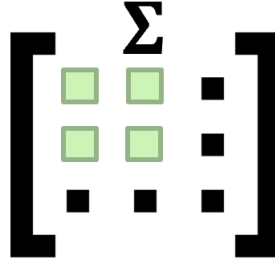
Giữ lại k cột đầu tiên của ma trận S , các cột còn lại sẽ được loại bỏ.



Hình 3.3. Cắt giảm ma trận S

2. Ma trận Σ

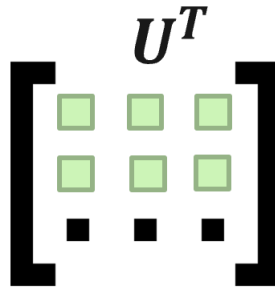
Giữ lại k dòng và k cột đầu tiên của ma trận Σ tính từ góc trái phía trên, tức là một ma trận con của ma trận Σ có kích thước $k \times k$, phần còn lại của ma trận Σ sẽ được loại bỏ.



Hình 3.4. Cắt giảm ma trận Σ

3. Ma trận U^T

Giữ lại k dòng đầu tiên của ma trận U^T , các dòng còn lại sẽ được loại bỏ.



Hình 3.5. Cắt giảm ma trận U^T

Khi đó K_k (ma trận từ khóa sau khi cắt giảm) và D_k (ma trận tài liệu sau khi cắt giảm) sẽ được tính theo công thức sau:

$$K_k = S_k \Sigma_k$$

$$D_k = \Sigma_k U_k^T$$

Vậy ma trận A_k mới thu được khi thực hiện phép nhân ba ma trận trên theo công thức của thuật toán SVD, ta được phép tính:

$$A_k = S_k \Sigma_k U_k^T$$

Theo tính chất của phép nhân ma trận với ma trận, kích thước của ma trận A_k là:

- Số dòng bằng với số dòng của ma trận S_k , tức bằng t
- Số cột bằng với số cột của ma trận U_k^T , tức bằng d

Như vậy, xét về kích thước, ma trận A_k sau khi cắt giảm sẽ có cùng kích thước với ma trận A ban đầu, tức cùng bằng $t \times d$.

Ma trận A_k lúc này là xấp xỉ của ma trận A ban đầu.

$$A_k \approx A$$

Điều này có nghĩa là mô hình LSI có khả năng tái cấu trúc lại được ma trận A ban đầu sau khi đã thực hiện cắt giảm ma trận.

3.6.3. Minh họa cắt giảm ma trận

Từ ví dụ được trình bày ở phần trước, ma trận A ban đầu được phân rã thành ba ma trận. Giả sử chỉ giữ lại 2 chiều ($k = 2$) tương ứng với 2 trị riêng lớn nhất:

$$\lambda_1 = 1 + \sqrt{2} \text{ và } \lambda_2 = 1$$

Khi đó, với mỗi ma trận S , Σ , U^T , ta thực hiện cắt giảm như sau:

$$S = \begin{pmatrix} 0.5 & 0.71 & -0.5 \\ 0.5 & -0.71 & -0.5 \\ 0.71 & 0 & 0.71 \end{pmatrix} \Rightarrow S_2 = \begin{pmatrix} 0.5 & 0.71 \\ 0.5 & -0.71 \\ 0.71 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 + \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \sqrt{2} \end{pmatrix} \Rightarrow \Sigma_2 = \begin{pmatrix} 1 + \sqrt{2} & 0 \\ 0 & 1 \end{pmatrix}$$

$$U^T = \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \\ -0.5 & -0.5 & 0.71 \end{pmatrix} \Rightarrow U_2^T = \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \end{pmatrix}$$

Thực hiện tính toán lại hai ma trận K_2 và D_2 :

$$K_2 = S_2 \Sigma_2 = \begin{pmatrix} 0.5 & 0.71 \\ 0.5 & -0.71 \\ 0.71 & 0 \end{pmatrix} \begin{pmatrix} 1 + \sqrt{2} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1.21 & 0.71 \\ 1.21 & -0.71 \\ 1.71 & 0 \end{pmatrix}$$

$$D_2 = \Sigma_2 U_2^T = \begin{pmatrix} 1 + \sqrt{2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0.5 & 0.71 \\ 0.71 & -0.71 & 0 \end{pmatrix} = \begin{pmatrix} 1.21 & 1.21 & 0.71 \\ 0.71 & -0.71 & 0 \end{pmatrix}$$

Như vậy, các vector biểu diễn các term trong không gian ngữ nghĩa tiềm ẩn trong trường hợp $k = 2$:

$$k_1 = (1.21, 0.71) \quad k_2 = (1.21, -0.71) \quad k_3 = (1.71, 0)$$

Các vector biểu diễn các tài liệu trong không gian ngữ nghĩa tiềm ẩn trong trường hợp $k = 2$:

$$Doc1 = \begin{pmatrix} 1.21 \\ 0.71 \end{pmatrix} \quad Doc2 = \begin{pmatrix} 1.21 \\ -0.71 \end{pmatrix} \quad Doc3 = \begin{pmatrix} 0.71 \\ 0 \end{pmatrix}$$

Tính lại tọa độ của vector q_k biểu diễn câu truy vấn q trong không gian ngữ nghĩa tiềm ẩn trong trường hợp này (giả sử truy vấn q chứa hai term t_1, t_2 tương ứng với vector k_1, k_2 trong không gian ngữ nghĩa tiềm ẩn):

$$q_k = k_2 + k_1 = (1.21, 0.71) + (1.21, -0.71) = (2.42, 0)$$

Kế đến, ta cần tính toán độ tương đồng giữa vector q_k và các vector tài liệu trong không gian ngữ nghĩa tiềm ẩn thông qua độ đo cosine, tương tự như khi chưa cắt giảm ma trận.

3.6.4. Ý nghĩa của cắt giảm ma trận

Mô hình LSI sau khi cắt giảm vẫn giữ được kích thước $t \times d$. Điều này thể hiện mô hình có thể tái cấu trúc lại ma trận ban đầu, và do đó gần như có thể *đảm bảo được tính toàn vẹn của dữ liệu*.

Khi thực hiện cắt giảm ma trận, mô hình LSI có khả năng tạo nên sự khác biệt rõ rệt giữa các vị trí thứ hạng, giúp tăng thứ hạng của các tài liệu thật sự liên quan (về ngữ nghĩa) và làm giảm thứ hạng của các tài liệu kém liên quan (về ngữ nghĩa), từ đó *giúp tăng độ chính xác của kết quả truy vấn*.

Việc cắt giảm ma trận còn giúp loại bỏ nhiễu (các chiều có mức độ quan trọng thấp) ứng với các trị riêng có độ lớn nhỏ, từ đó giúp tăng cường mối liên kết về mặt ngữ nghĩa giữa các từ có trong tập tài liệu.

CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM

4.1. Xử lý dữ liệu

4.1.1. Dữ liệu Cranfield

Bộ dữ liệu Cranfield sẽ được trích xuất thành 3 file: document (tập tài liệu), queries (tập truy vấn) và relevance (tập tài liệu liên quan) để phục vụ cho việc truy xuất.

Với tập tài liệu, dữ liệu có cấu trúc gồm các đầu dòng: .I, .T, .A, .B, .W thể hiện như hình dưới:

```
.I 1
.T
experimental investigation of the aerodynamics of a
wing in a slipstream .
.A
brenckman,m.
.B
j. ae. scs. 25, 1958, 324.
.W
experimental investigation of the aerodynamics of a
wing in a slipstream .
    an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios . the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
    the comparative span loading curves, together with
supporting evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or
boundary-layer-control effect . the integrated remaining lift
increment, after subtracting this destalling lift, was found to agree
well with a potential flow theory .
    an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
```

Hình 4.1. Tập tài liệu Cranfield gốc

Mỗi file document ứng với 1 tài liệu và sẽ chứa nội dung trong đoạn .W đến trước .I của tài liệu tiếp theo, ID của tài liệu là chữ số sau .I, đến trước .T trong cùng một tài liệu và file được lưu dưới dạng <ID tài liệu>.txt.

Với tập truy vấn, mỗi truy vấn sẽ có cấu trúc gồm các đầu dòng: .I, .W. thể hiện như hình dưới:

```
.I 001
.W
what similarity laws must be obeyed when constructing aeroelastic models
of heated high speed aircraft .
.I 002
.W
what are the structural and aeroelastic problems associated with flight
of high speed aircraft .
```

Hình 4.2. Tập truy vấn Cranfield gốc

Tương tự như cách xử lý của tập tài liệu, nội dung truy vấn sẽ là nội dung sau .W đến .I của tài liệu tiếp theo, ID sẽ là số sau .I và trước .W của tài liệu.

Với tập tài liệu liên quan, mỗi truy vấn sẽ có các tài liệu liên quan thể hiện như hình dưới:

```
1 184 2
1 29 2
1 31 2
1 12 3
1 51 3
1 102 3
1 13 4
1 14 4
1 15 4
1 57 2
```

Hình 4.3. Tập tài liệu liên quan Cranfield gốc

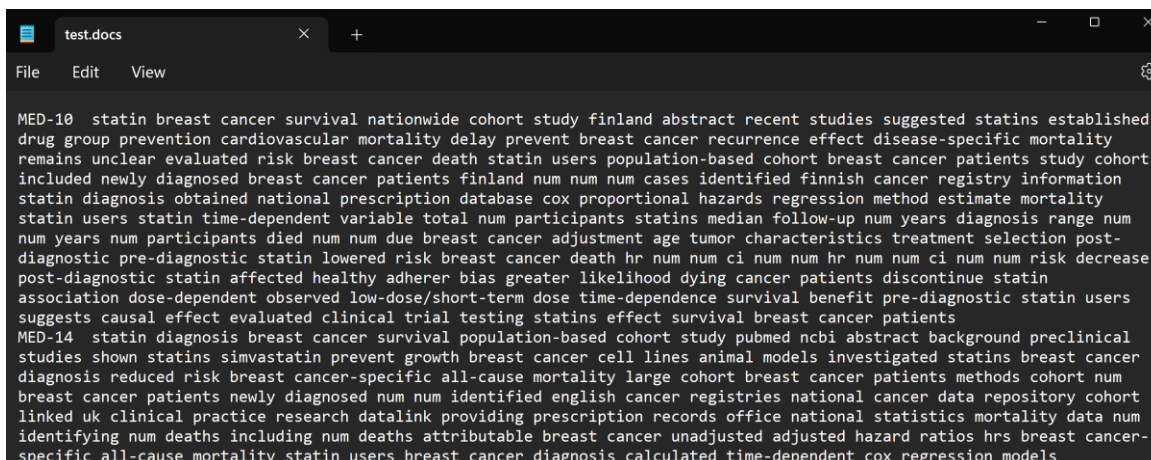
Trong đó, cột 1 sẽ tương ứng với số ID của truy vấn trong tập truy vấn, cột 2 sẽ tương ứng với tài liệu liên quan. Ta sẽ xử lý là đối với truy vấn có ID theo cột 1, thì sẽ có các ID tài liệu theo cột 2. File cũng sẽ được lưu dưới dạng <ID truy vấn>.txt. Ví dụ như trên thì ta sẽ lưu vào file 1.txt với nội dung là “184 29 31 12 51 102 13 14 15 57”.

Tổng số file thu được sau khi thực hiện trích xuất thông tin là: 1400 tài liệu document, 255 tài liệu truy vấn và 255 tài liệu liên quan. Và đây cũng chính là bộ dữ liệu mà nhóm chúng em sử dụng – bộ dữ liệu sau khi được tiền xử lý.

4.1.2. Dữ liệu NFCorpus

Tương tự như bộ dữ liệu Cranfield, bộ dữ liệu NF Corpus cũng sẽ được trích xuất thành 3 file: document, queries, relevance.

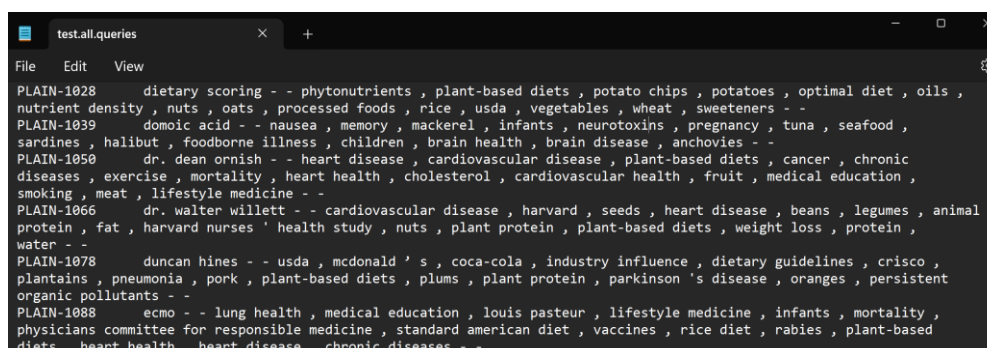
Đến với tập document, dữ liệu gốc là *test.docs*, có cấu trúc có cấu trúc gồm ID tài liệu (MED-X) và nội dung tài liệu như hình dưới:



Hình 4.4. Tập tài liệu NF Corpus gốc

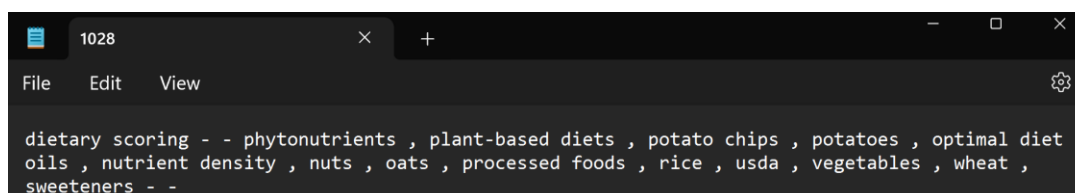
File document của mỗi tài liệu sẽ được xử lý bằng cách lấy nội dung sau ID tài liệu và lưu vào file <ID tài liệu>.txt.

Tập câu truy vấn gốc nằm trong file *test.all.queries*, cũng sẽ có cấu trúc gồm hai phần là ID truy vấn và nội dung truy vấn như hình dưới:



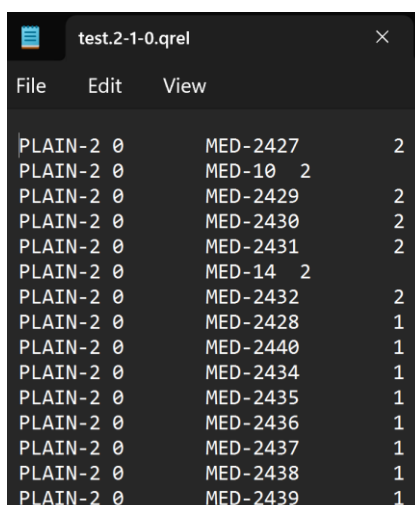
Hình 4.5. Tập truy vấn NF Corpus gốc

Ở đây, nhóm thực hiện lấy toàn bộ nội dung truy vấn, lưu dưới dạng <ID truy vấn>.txt. Chẳng hạn với file 1028.txt sẽ chứa nội dung:



Hình 4.6 Ví dụ câu truy vấn trong NF Corpus

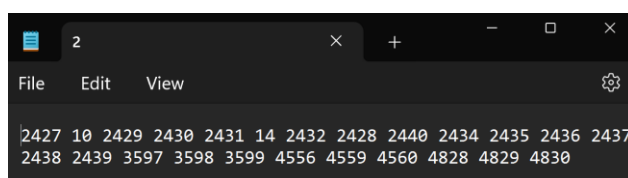
Tập tài liệu liên quan gốc chưa được tiền xử lý sẽ nằm trong file *test.2-1-0.qrel*. Sẽ có dạng như sau:



File	Edit	View
PLAIN-2 0	MED-2427	2
PLAIN-2 0	MED-10	2
PLAIN-2 0	MED-2429	2
PLAIN-2 0	MED-2430	2
PLAIN-2 0	MED-2431	2
PLAIN-2 0	MED-14	2
PLAIN-2 0	MED-2432	2
PLAIN-2 0	MED-2428	1
PLAIN-2 0	MED-2440	1
PLAIN-2 0	MED-2434	1
PLAIN-2 0	MED-2435	1
PLAIN-2 0	MED-2436	1
PLAIN-2 0	MED-2437	1
PLAIN-2 0	MED-2438	1
PLAIN-2 0	MED-2439	1

Hình 4.7. Tập tài liệu liên quan NF Corpus gốc

Ở đây, ta sử dụng dữ liệu ở cột 1 là ID truy vấn và cột 3 là tài liệu liên quan. Thông tin này sẽ được lưu dưới dạng <ID truy vấn>.txt. Ví dụ với truy vấn có ID là 2:



File	Edit	View
2427	10	2429 2430 2431 14 2432 2428 2440 2434 2435 2436 2437 2438 2439 3597 3598 3599 4556 4559 4560 4828 4829 4830

Hình 4.8. Ví dụ tập tài liệu liên quan của truy vấn 2

Sau khi thông qua bước xử lý này, số lượng tài liệu thu được là: 3162 tài liệu, 325 queries và 323 relevance. Bộ dữ liệu đã được tiền xử lý này sẽ được nhóm sử dụng trong các thử nghiệm của nhóm.

4.2. Phương pháp đánh giá

Nhóm sử dụng hai phương pháp đánh giá được sử dụng phổ biến hiện nay để đánh giá cho mô hình truy xuất thông tin là:

- Thời gian truy xuất
- mAP: Giá trị trung bình của Precision.

Trong đó mAP được tính như sau:

- *Bước 1: Tính Precision*

$$Precision = \frac{\text{Tổng số tài liệu được truy xuất có liên quan}}{\text{Tổng số tài liệu được truy xuất}}$$

- *Bước 2: Tính Recall*

$$Recall = \frac{\text{Tổng số tài liệu được truy xuất có liên quan}}{\text{Tổng số tài liệu có liên quan trong file relevance}}$$

- *Bước 3: Tính AP*

$$AP = \sum_{k=0}^{n-1} [Recalls(k) - Recalls(k + 1)] \times Precision(k)$$

Trong đó:

- $Recalls(n) = 0$
- $Precision(n) = 1$
- n : số ngưỡng

AP (giá trị trung bình của precision) là một cách để tóm tắt đường cong precision-recall thành một giá trị duy nhất đại diện cho giá trị trung bình của tất cả các giá trị precision. Bằng cách sử dụng một vòng lặp duyệt qua tất cả các lần precision/recalls, sự khác biệt giữa giá trị recalls hiện tại và giá trị recalls tiếp theo được tính toán và sau đó nhân với giá trị precision hiện tại. Nói cách khác, AP là tổng trọng số các giá trị precisions ở mỗi ngưỡng, trong đó trọng số là mức tăng của giá trị recalls.

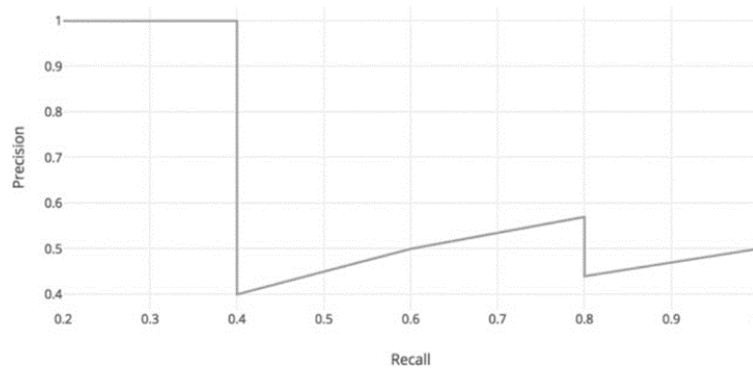
- *Bước 4: Tính nội suy 11 điểm*

Chia giá trị recalls từ 0 đến 1.0 thành 11 điểm là 0, 0.1, 0.2, ..., 0.9, 1.0. Tiếp theo, tính giá trị trung bình của giá trị max precision cho 11 giá trị recalls này.

$$AP = \frac{1}{11} \times (AP_R(0) + AP_R(0.1) + \dots + AP_R(1.0))$$

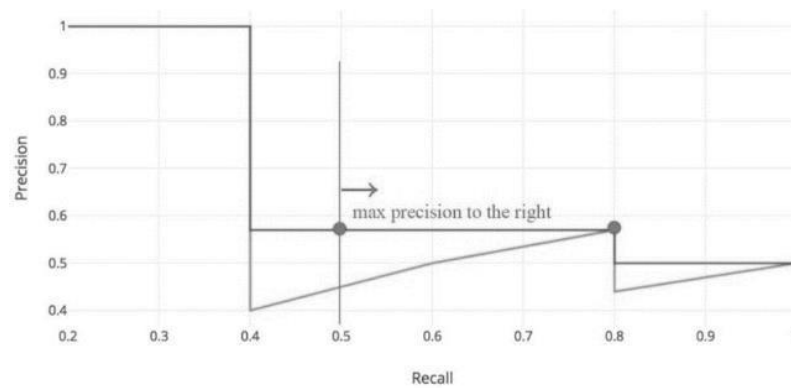
Mục đích trong việc tìm giá trị nội suy 11 điểm là khi mà ta biểu diễn các giá trị AP lên một biểu đồ, thì nó sẽ tạo thành đường cong tên đường cong

precision - recall, nhưng đường này nếu không nội suy thì nó sẽ gấp khúc theo zigzag do các thay đổi nhỏ lúc xếp hạng như hình minh họa dưới:



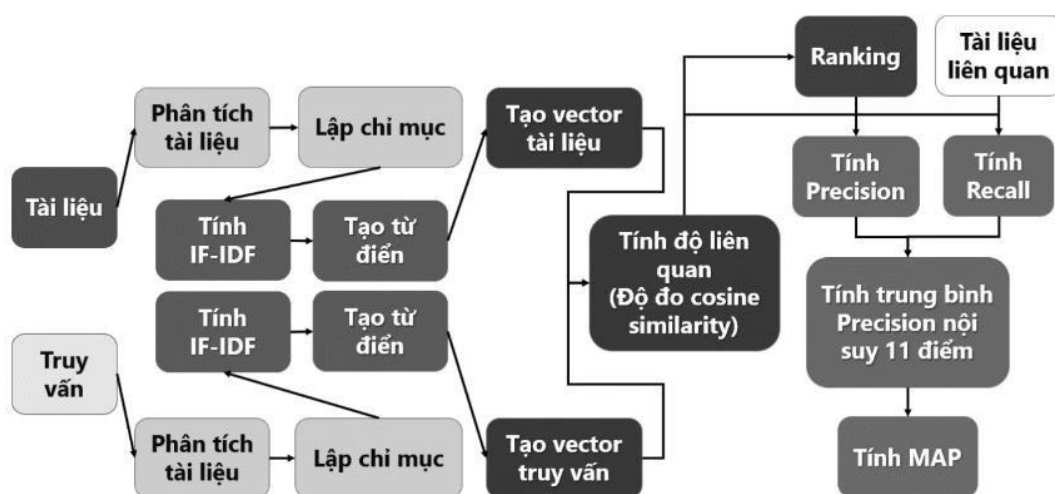
Hình 4.9. Minh họa đồ thị biểu diễn AP trước khi nội suy

Vì vậy, cần phải nội suy 11 điểm để làm giảm bớt đi sự gấp khúc này:



Hình 4.10. Minh họa đồ thị biểu diễn AP sau khi nội suy

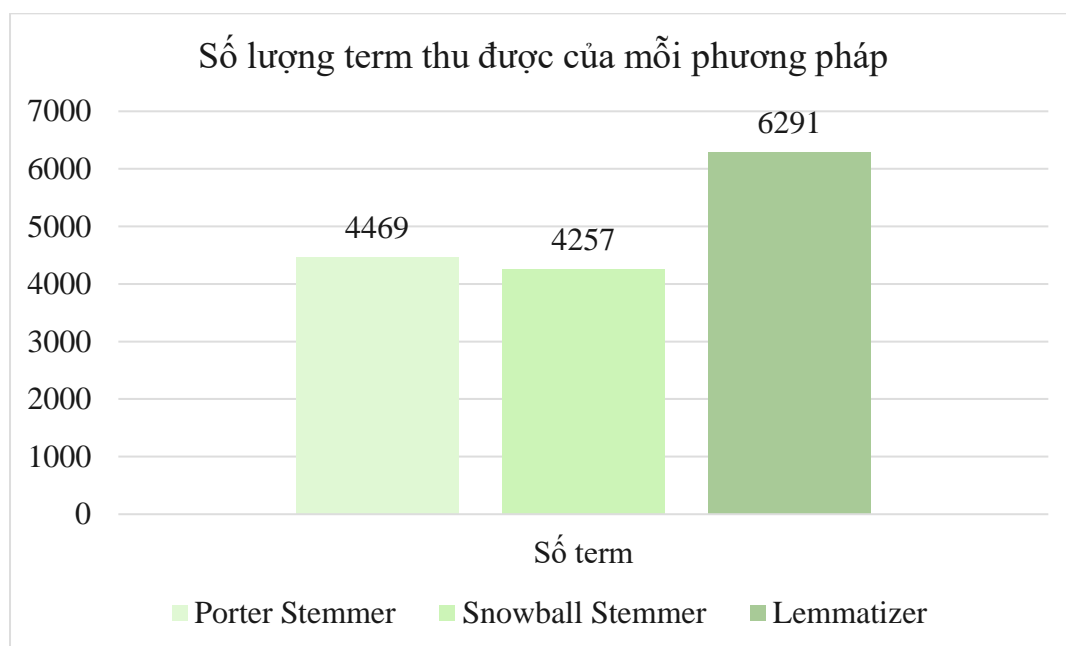
4.3. Mô hình Vector Space



Hình 4.11. Sơ đồ thực hiện của mô hình VSM

Dựa vào sơ đồ này, có thể nhận định rằng kết quả của truy vấn sẽ phụ thuộc khá nhiều vào giai đoạn phân tích tài liệu. Khi đó, việc chọn term và số lượng term sẽ ảnh hưởng lớn đến việc lập chỉ mục. Nhóm đã tiền xử lý dữ liệu theo phương pháp lấy gốc từ và số lượng term cho ra lần lượt là:

- PorterStemmer: 4469 terms
- Lemmatizer: 6291 terms
- SnowballStemmer: 4257 terms



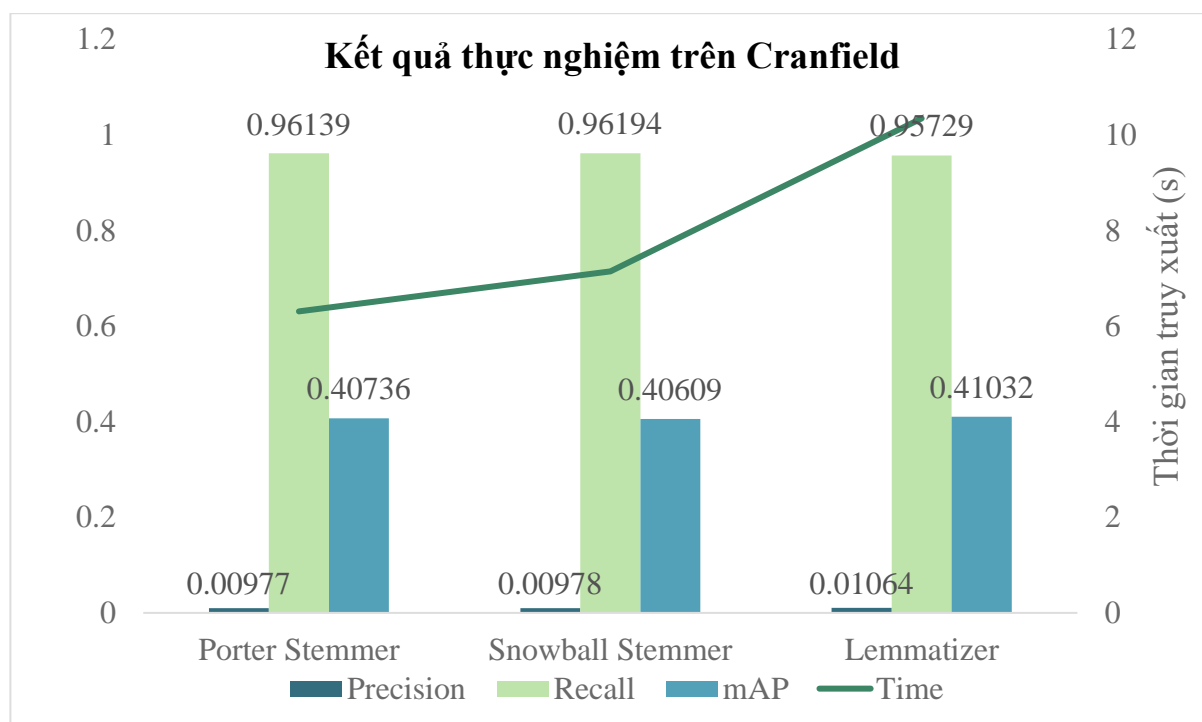
Biểu đồ 4.1. Số lượng term thu được của mỗi phương pháp

Tuy nhiên có thể thấy, mặc dù số lượng term ứng với từng phương pháp lấy gốc từ chênh lệch khá nhiều nhưng kết quả MAP của mô hình khi sử dụng 3 phương pháp này lại không có sự chênh lệch nào đáng kể.

Bảng 4.1. Thống kê kết quả thực nghiệm mô hình Vector Space

	Porter Stemmer	Snowball Stemmer	Lemmatizer
<i>Precision</i>	0.009775	0.009777	0.010636
<i>Recall</i>	0.961389	0.961944	0.957294
<i>mAP</i>	0.407357	0.406089	0.410316

<i>Time</i>	6.308572	7.144499	10.34679
-------------	----------	----------	----------



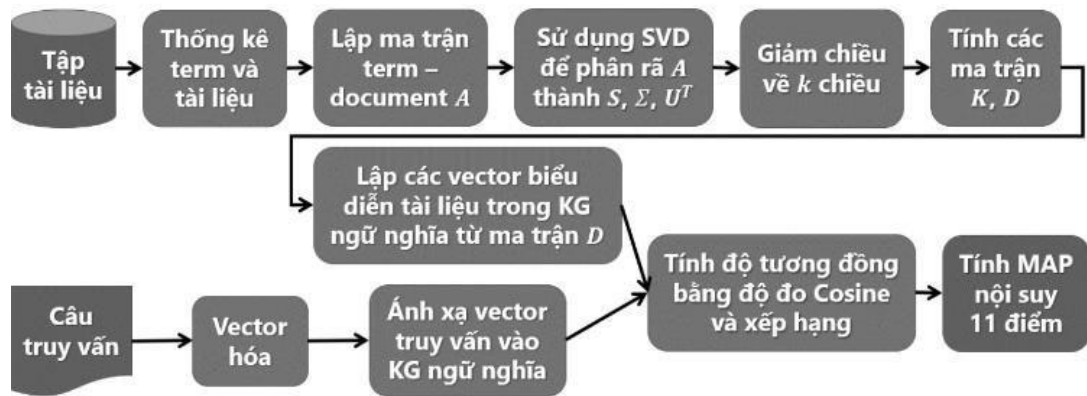
Biểu đồ 4.2. Kết quả thực nghiệm trên bộ dữ liệu Cranfield

Kết quả khi tiền xử lý dữ liệu với Letimatizer sẽ được sử dụng là kết quả chính của mô hình Vector Space để so sánh với mô hình LSI vì thực nghiệm này có kết quả tốt nhất.

Bảng 4.2. Kết quả thực nghiệm trên bộ dữ liệu NFCorpus

	Porter Stemmer	Lemmatizer
<i>Precision</i>	0.00584	0.0058
<i>Recall</i>	0.27844	0.27847
<i>mAP</i>	0.005458	0.0056
<i>Time</i>	129.61	145.39

4.4. Mô hình LSI

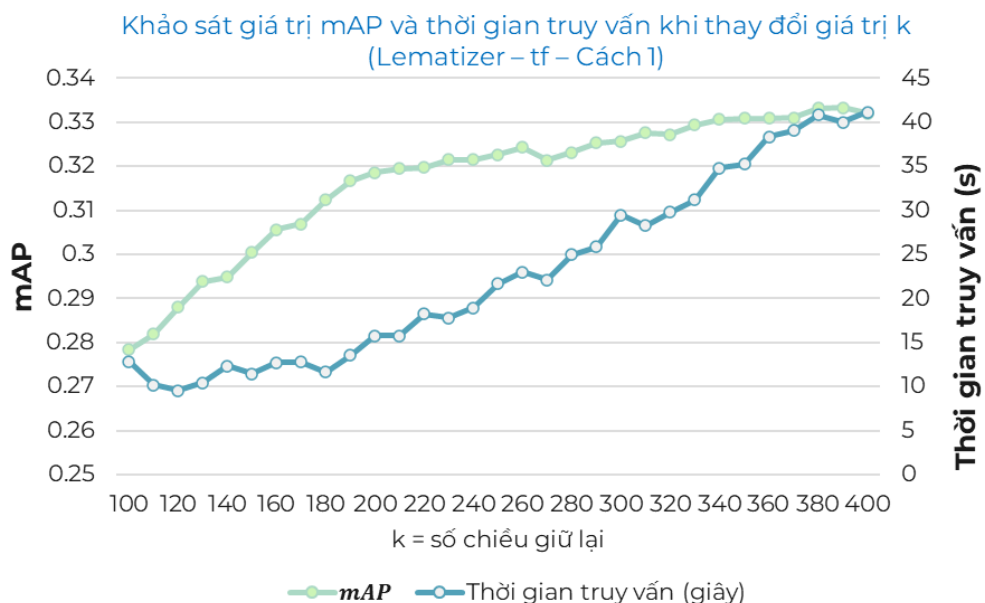


Hình 4.12. Sơ đồ thực hiện của mô hình LSI

Như đã trình bày về mô hình LSI, chúng ta thấy hệ số k (số chiều giữ lại sau khi thực hiện cắt giảm ma trận) là một trong những vấn đề cần được quan tâm của mô hình này.

Để chọn được hệ số k tối ưu, chúng ta cần phải thử nghiệm rất nhiều lần trên mỗi bộ ngữ liệu. Nhóm đã tiến hành chạy thử nghiệm tìm giá trị tối ưu của hệ số k trên cả hai tập dữ liệu Cranfield và NFCorpus để tìm ra giá trị tối ưu đối với từng bộ dữ liệu theo tiêu chí độ chính xác và thời gian truy vấn.

Ngoài ra, tương tự mô hình VSM, có thể nhận thấy phương pháp lấy gốc từ và công thức đánh trọng số cũng khá quan trọng. Với tập dữ liệu Cranfield, nhóm tiến hành khảo sát k thông qua các thử nghiệm để có thể xác định mô hình cho kết quả tốt nhất.



Biểu đồ 4.3. Khảo sát giá trị mAP và thời gian truy vấn khi thay đổi giá trị k

(Lemmatizer – tf – Cách 1)

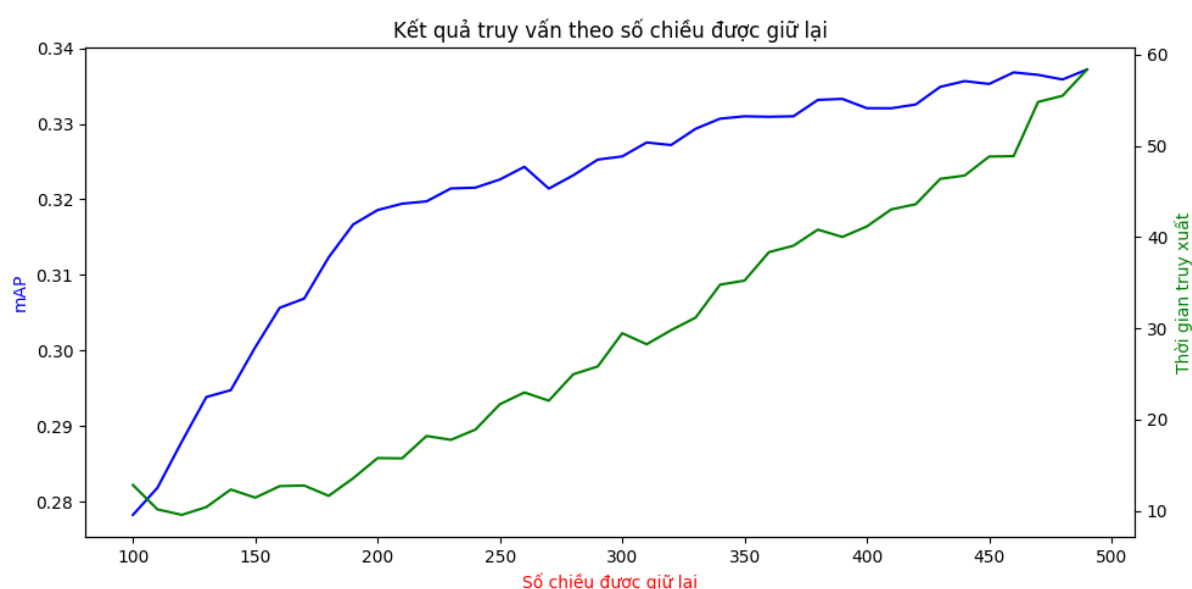
Kết quả khảo sát cho thấy, $k = 390$ là kết quả tối ưu nhất. Khi đó, $mAP = 0.33329$, thời gian truy vấn = 40.0145s.

Thực hiện tương tự với các phương thức khác, ta có:

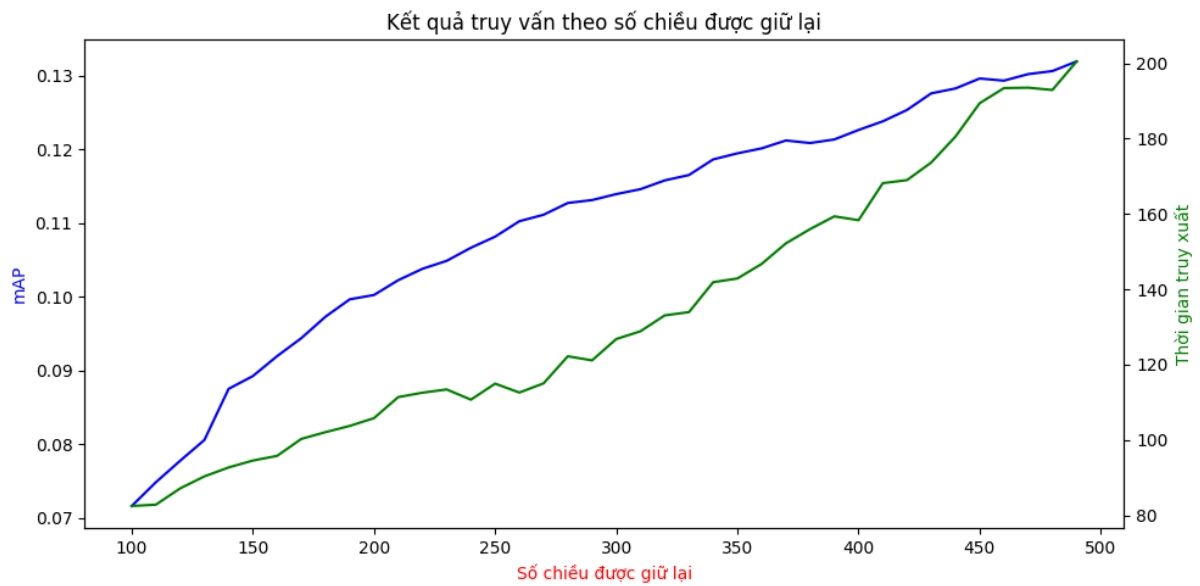
Bảng 4.3. Kết quả thực nghiệm các lần thử số chiều khác nhau

Lần thử	PP lấy gốc từ	Số chiều tối ưu	Kết quả (mAP & Time)
1	Lemmatizer	390	mAP = 0.33329 Time = 40.0145s
2	Lemmatizer	230	mAP = 0.32139 Time = 10.77429s
3	Lemmatizer	1400	mAP = 0.34744 Time = 15.1s
4	Lemmatizer	360	mAP = 0.34611 Time = 18.86389s

Kết quả của lần thử thứ 3 là kết quả tốt nhất. khi $mAP = 0.34744$ và thời gian ngắn nhất với 7.5101s.



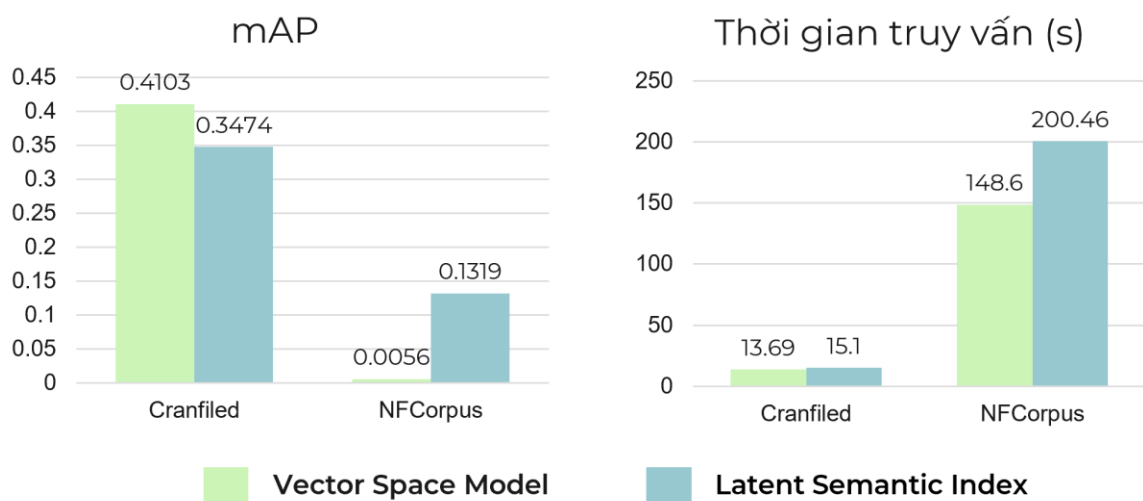
Hình 4.13. Kết quả truy vấn theo số chiều được giữ lại trên Cranfield



Hình 4.14. Kết quả truy vấn theo số chiều được giữ lại trên NFCorpus

4.5. So sánh hai mô hình

Để so sánh 1 cách khách quan, nhóm còn sử dụng thêm bộ dữ liệu NFCorpus. Kết quả cho ra như sau:



Biểu đồ 4.4. So sánh 2 mô hình với 2 bộ dữ liệu về kết quả mAP và thời gian truy vấn

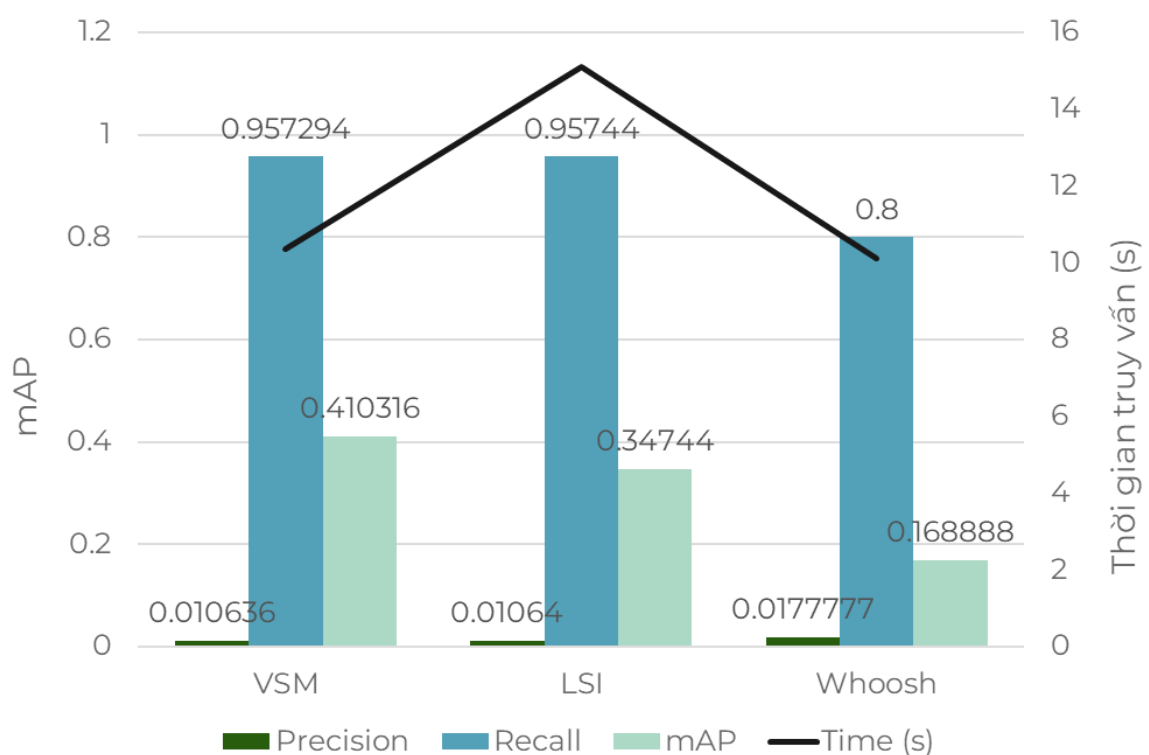
Đối với việc sử dụng hai bộ ngữ liệu khác nhau cài đặt cho mô hình Vector Space và LSI, nhóm rút ra một số nhận xét sau:

- Bộ ngữ liệu Cranfield cho kết quả mAP cao hơn so với bộ ngữ liệu NFCorpus trong cả hai mô hình.

- Khi sử dụng bộ ngữ liệu Cranfield cho mô hình Vector Space, ta thu được kết quả mAP cao nhất với giá trị là 0.41032.
- Khi sử dụng bộ ngữ liệu NFCorpus cho mô hình Vector Space, ta thu được kết quả mAP thấp nhất với giá trị là 0.0056.
- Thời gian truy vấn ở cả hai mô hình khi sử dụng bộ ngữ liệu Cranfield luôn thấp hơn so với khi sử dụng bộ ngữ liệu NFCorpus.

Mặc dù nói rằng, LSI là mô hình cải tiến của VSM. Nhưng kết quả lại không chênh lệch quá nhiều. Có thể giải thích như sau: Trong mô hình LSI, nhóm đã sử dụng Boolean để lập chỉ mục và truy vấn. Theo lý thuyết, mô hình Boolean có tính hiệu quả thấp hơn Vector Space. Vì vậy, kết quả mAP của LSI sẽ thấp hơn mô hình Vector Space.

4.6. Mô hình Vector Space, LSI và thư viện Whoosh



Biểu đồ 4.5. So sánh mô hình Vector Space, mô hình LSI và thư viện Whoosh

Có thể thấy rằng kết quả 2 mô hình VSM và LSI cao hơn khi dùng thư viện Whoosh. Tuy nhiên, Whoosh lại có lợi thế về thời gian. Điều này chứng tỏ là mô hình vẫn chưa thực sự tốt. Cần được cải tiến nhiều hơn.

CHƯƠNG 5. TỔNG KẾT

5.1. Kết luận

Sau khi thực hiện xong đề tài này, nhóm đã rút ra một số kết luận về hai mô hình Vector Space và Latent Semantic Analysis như sau:

- Mô hình Latent Semantic Analysis vẫn là một phiên bản cải tiến của mô hình Vector Space theo lý thuyết. Nhưng vì việc thực nghiệm mô hình LSI sử dụng Boolean để lập chỉ mục sẽ tạo ra kết quả chưa được như mong đợi, thay vì thực nghiệm mô hình LSI sử dụng Vector Space để lập chỉ mục.
- Độ đo mAP của cả 2 mô hình cũng phụ thuộc vào tính chất của bộ ngữ liệu.
- Tuy độ đo mAP vẫn chưa thể hiện được tính chính xác của mô hình LSI với Vector Space, nhưng thời gian thực hiện truy vấn thể hiện khá tốt vấn đề này.

5.2. Hướng phát triển

Một số định hướng phát triển nhằm cải tiến mô hình Vector Space và Latent Semantic Analysis như sau:

- Thử kết hợp mô hình Latent Semantic Analysis và mô hình Vector Space để kiểm tra rõ về mức độ hiệu quả của mô hình Latent Semantic Analysis so với mô hình Vector Space.
- Đồng thời thử nghiệm thêm các kết quả đánh giá dựa trên một số độ đo đánh giá như: Recall, thời gian lập chỉ mục, dung lượng bộ nhớ cần lưu trữ, F1-Score,... Để có một đánh giá khách quan hơn trong việc so sánh mức độ hiệu quả giữa 2 mô hình với nhau.
- Việc đánh giá hiệu quả hơn hay không phụ thuộc vào yêu cầu người xây dựng mô hình truy vấn thông tin, chẳng hạn như người xây dựng mô hình truy vấn có thể đặt ra yêu cầu là mô hình này cần có độ đo mAP cao và dung lượng bộ nhớ lưu trữ thấp. Ví dụ như người xây dựng mô hình truy vấn có yêu cầu là thời gian kể từ lúc huấn luyện cho đến lúc truy vấn phải thấp, thì khi đó chúng

ta cần phải xét thêm thời gian lập chỉ mục. Việc càng có nhiều hình thức đánh giá sẽ giúp cho mô hình của chúng ta sẽ được đánh giá khách quan và chính xác hơn, do đó sẽ có độ tin cậy cao hơn. Thông qua đó, chúng ta có thể đưa ra các kết luận, nhận xét một cách khách quan và trực quan hơn.

TÀI LIỆU THAM KHẢO

- [1] W. B. F. a. R. Baeza-Yates, Information Retrieval: Data Structures & Algorithms.
- [2] S. & R. V. Wong, "Vector Space Model of Information Retrieval - A Reevaluation," *Communications of The ACM - CACM*, 1984.
- [3] "Latent semantic indexing," [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/latent-semantic-indexing-1.html>.
- [4] "NFCorpus: A Full-Text Learning to Rank Dataset for Medical Information Retrieval," [Online]. Available: <https://www.cl.uni-heidelberg.de/statnlpgroup/nfcorpus/>.
- [5] "Cranfield collection," [Online]. Available: http://ir.dcs.gla.ac.uk/resources/test_collections/cran/.