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

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

BÁO CÁO ĐỒ ÁN
INFORMATION RETRIEVAL USING VECTOR SPACE MODEL & LATENT SEMANTIC INDEX

MÔN TRUY XUẤT THÔNG TIN

GVHD: Nguyễn Trọng Chỉnh
Nhóm 9:

1. Nguyễn Văn Thành Đạt 20520436
2. Lê Ngọc Mỹ Trang 20520817
3. Lê Minh 20521599

Thành phố Hồ Chí Minh, tháng 7 năm 2023

MỤC LỤC

BẢNG PHÂN CÔNG VÀ DANH SÁCH THÀNH VIÊN

Công việc	**Danh sách thành viên**		
	Nguyễn Văn Thành Đạt (20520436)	**Lê Ngọc Mỹ Trang (20520817)**	**Lê Minh (20521599)**
Xử lý data (đọc file, định dạng kiểu dữ liệu đầu vào, kiểu dữ liệu lưu trữ, tiền xử lý)	Có tham gia	Có tham gia	Có tham gia
Cài đặt các mô hình	Có tham gia	Không tham gia trực tiếp	Không tham gia trực tiếp
Đánh giá kết quả	Không tham gia trực tiếp	Có tham gia	Có tham gia
Làm slide	Có tham gia	Có tham gia	Có tham gia
Thuyết trình	Có tham gia	Có tham gia	Có tham gia
Viết báo cáo	Có tham gia	Có tham gia	Có tham gia
Đánh giá tiến độ từng người	100% Hoàn thành các công việc được phân công	100% Hoàn thành các công việc được phân công	100% Hoàn thành các công việc được phân công
Tổng đóng góp trên cả nhóm	33,33…%	33,33…%	33,33…%
	100%		

I. Giới thiệu bài toán

1. Giới thiệu

Trong thế giới hiện đại với nguồn thông tin ngày càng phong phú và đa dạng, việc tìm kiếm thông tin cần thiết đôi khi trở nên khó khăn. Bài toán truy xuất thông tin được đặt ra để giải quyết vấn đề này, nhằm giúp con người tìm ra thông tin mình cần một cách nhanh chóng và hiệu quả. Đây là một trong những bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên và học máy, liên quan tới việc tìm kiếm và khám phá thông tin hữu ích từ một kho dữ liệu lớn như cơ sở dữ liệu trong máy hoặc trang web.

Truy xuất thông tin có ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm công nghệ thông tin, khoa học thông tin, tìm kiếm web, truyền thông và nhiều lĩnh vực khác. Các công cụ truy xuất thông tin phổ biến như công cụ tìm kiếm web và hệ thống quản lý tài liệu đều dựa trên các nguyên tắc và phương pháp của bài toán truy xuất thông tin.

Với sự phát triển của trí tuệ nhân tạo và học máy, các phương pháp truy xuất thông tin đã được cải tiến đáng kể, cho phép hệ thống truy xuất thông tin trở nên thông minh hơn và đáp ứng tốt hơn với yêu cầu của người dùng. Việc nghiên cứu và phát triển các giải thuật truy xuất thông tin cung cấp một cơ sở quan trọng để tận dụng và khai thác tri thức từ nguồn thông tin phong phú của thế giới hiện đại.

2. Phát biểu bài toán

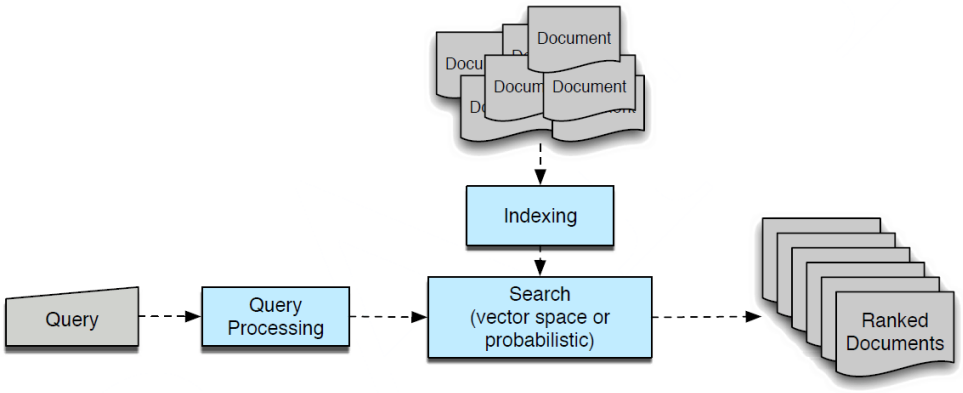
- Input: câu truy vấn từ người dùng, được biểu diễn dưới dạng văn bản tiếng Anh
- Output: một danh sách gồm tên các tài liệu có liên quan tới câu truy vấn trên được lấy từ cơ sở dữ liệu có sẵn (Database), được sắp xếp theo mức độ liên quan giảm dần
- Cơ sở dữ liệu (Database): một tập hợp gồm danh sách các tài liệu dưới dạng văn bản tiếng Anh
- Ví dụ:

Input	Output
1: “what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft.”	184: “scale models for thermo aeroelastic research. an investigation is made of the parameters to be satisfied for thermo aeroelastic similarity…” 29: “a simple model study of transient temperature and thermal stress distribution due to aerodynamic heating …” 31: “thermal buckling of supersonic wing panels. The temperature and thermal stress distributions are analyzed in multicellular supersonic wing structures. …” …

Bảng 1 – Ví dụ input và output của bài toán

3. Quy trình thực hiện

Truy xuất thông tin bao gồm hai phần chính: truy xuất và đánh giá.



Hình 1 - Mô hình hệ thống truy xuất thông tin

Quy trình thực hiện gồm các bước:

- Bước 1: Tiền xử lý dữ liệu Database

- Bước 2: Tiến hành lập chỉ mục
- Bước 3: Xử lý câu truy vấn
- Bước 4: Thực hiện truy vấn
- Bước 5: Đánh giá kết quả truy vấn

II. Bộ dữ liệu

1. Thông tin bộ dữ liệu

Bộ dữ liệu dùng trong đồ án này là bộ Cranfield và bộ NFCorpus.

a. Cranfield

Dây là một trong những bộ dữ liệu cổ điển và được sử dụng rộng rãi trong lĩnh vực truy xuất thông tin, được tạo ra vào những năm 1960 tại Đại học Cranfield ở Anh và được sử dụng để nghiên cứu và đánh giá các phương pháp truy xuất thông tin. Bộ dữ liệu Cranfield ban đầu bao gồm các tài liệu khoa học từ nhiều lĩnh vực, chẳng hạn như hàng không, hóa học, vật lý, y học và khoa học máy tính.

Bộ dữ liệu sử dụng ngôn ngữu là tiếng Anh. Cấu trúc bộ dữ liệu:

- Thư mục Cranfield chứa tổng cộng 1.400 tài liệu, trong đó có bài báo, báo cáo kỹ thuật và tóm tắt các bài báo. Các tài liệu được đánh số từ 1 đến 1400.
- Thư mục TEST gồm 2 thành phần:
    - Tệp query.txt chứa 225 câu truy vấn được đánh số từ 1 đến 225
    - Thư mục RES gồm 225 tệp .txt đánh số từ 1 đến 225, chứa các kết quả liên quan đến 225 câu truy vấn từ 1400 tài liệu Cranfield

b. NFCorpus

NFCorpus là một bộ dữ liệu được sử dụng cho mục đích đánh giá và phát triển các hệ thống tìm kiếm thông tin y tế, tập trung vào việc xếp hạng các tài liệu y tế để đáp ứng nhu cầu tìm kiếm thông tin y tế của người dùng.

Bộ dữ liệu này bao gồm 3244 câu truy vấn được viết bằng tiếng Anh không chuyên từ trang web "NutritionFacts" với 169756 đánh giá độ liên quan được

trích xuất tự động cho 9964 tài liệu y tế được viết theo chuyên môn, chủ yêu từ trang web "PubMed".

Cấu trúc bộ dữ liệu:

- Dữ liệu truy vấn từ NutritionFacts.org (là các file .queries, gồm 5 loại khác nhau) – được viết bằng ngôn ngữ tự nhiên không chuyên
- Các tài liệu y tế (là các file .docs) – được viết bằng ngôn ngữ chuyên môn, học thuật
- Các file chứa kết quả đánh giá độ liên quan (.qrel)

Do số lượng khổng lồ của bộ dữ liệu nên nhóm chỉ chọn một bộ phận nhỏ để tiến hành thực hiện đồ án, gồm các file:

- train.docs: danh sách tài liệu
- train.all.queries: danh sách câu truy vấn
- train.3-2-1.qrel: danh sách kết quả chứa độ liên quan với mỗi câu truy vấn

2. Các bước tiền xử lý dữ liệu

a. Chọn term

- Loại bỏ các ký tự đặc biệt:
    - Trước tiên, khi xem xét tập tài liệu nhóm, nhóm phát hiện do một số nguyên nhân như dấu chấm câu, đầu dòng, … khiến các ký tự có 2 dạng hoặc là chữ hoa, hoặc là chữ thường nên đã tiến hành biến đổi tất cả các ký tự trong từng tập tài liệu về dạng chữ thường để đảm bảo tính nhất quán, các từ giống nhau không bị hiểu là hai từ khác nhau.
    - Tiếp theo là loại bỏ các ký tự đặc biệt ngoài chữ và số như các dấu câu, hay là ký tự @#%$()... do chúng xuất hiện ở mọi tài liệu và không đóng góp lớn vào ý nghĩa của câu, bên cạnh đó việc đứng sát với các từ sẽ khiến máy hiểu nhầm đó là một từ mới như từ (spheres) và spheres trong ví dụ bên dưới
- Chọn term:

- Term hay từ khóa là đơn vị cơ bản nhất trong truy xuất thông tin, thường có dạng từ đơn hay cụm từ cụ thể.
- Thực hiện tách các từ cách nhau bởi khoảng trắng thành các term. Nhóm chọn term là các từ đơn cách nhau bởi khoảng trắng vì:
  - Đơn giản và tiện lợi, phù hợp với cách diễn đạt ngôn ngữ văn bản bình thường
  - Tổng quát được nhiều khía cạnh liên quan hơn, đảm bảo rằng kết quả truy xuất thông tin bao gồm các tài liệu và nguồn thông tin có liên quan đến nhiều khía cạnh của yêu cầu

- **Ví dụ: doc 7.txt**

```
experiments were performed in the 12 in. supersonic wind tunnel of the jet propulsion laboratory of the california institute of technology to investigate the effect of three dimensional roughness elements (spheres) on boundary layer transition on a tained at local mach numbers of 1.90, 2.71, and 3.67 by varying trip size, position, spacing, and reynolds number per inch. the results indicate that (1) transition from laminar to turbulent flow induced by three dimensional roughness elements begins when the double row of spiral vortices trailing each element contaminates and breaks down the surrounding field of vorticity,
```

Sẽ biến thành danh sách các term với 95 terms:

```
['experiments', 'were', 'performed', 'in', 'the', '12', 'in', 'supersonic', 'wind', 'tunnel', 'of', 'the', 'jet', 'propulsion', 'laboratory', 'of', 'the', 'california', 'institute', 'of', 'technology', 'to', 'investigate', 'the', 'effect', 'of', 'three', 'dimensional', 'roughness', 'elements', 'spheres', 'on', 'boundary', 'layer', 'transition', 'on', 'a', 'tained', 'at', 'local', 'mach', 'numbers', 'of', '190', '271', 'and', '367', 'by', 'varying', 'trip', 'size', 'position', 'spacing', 'and', 'reynolds', 'number', 'per', 'inch', 'the', 'results', 'indicate', 'that', '1', 'transition', 'from', 'laminar', 'to', 'turbulent', 'flow', 'induced', 'by',
```

```
'three', 'dimensional', 'roughness', 'elements', 'begins',
'when', 'the', 'double', 'row', 'of', 'spiral', 'vortices',
'trailing', 'each', 'element', 'contaminates', 'and', 'breaks',
'down', 'the', 'surrounding', 'field', 'of', 'vorticity']
```

b. Loại bỏ stopword

Loại bỏ stopwords khỏi câu, giữ lại các từ vựng mang ý nghĩa quan trọng

Nhóm sẽ sử dụng danh sách stopwords tiếng Anh của thư viện nltk, bao gồm các từ như sau:

- Các đại từ xưng hô với các biến thể: i, me, myself, we, our, you'll, it, it's, you'd, …
- Các trợ động từ: is, are, was, had, has, having, doing, …
- Giới từ: up, down, in, off, over, …
- Mạo từ: a, an, the, …

Danh sách này bao gồm 179 stopwords, được sử dụng để loại bỏ chúng ra khỏi các tài liệu cũng như truy vấn vì bản thân chúng không mang ý nghĩa ngữ nghĩa quá nhiều và có thể gây nhiễu cho mô hình tìm kiếm.

Thực hiện loại bỏ stopwords sẽ giảm thiểu nhiễu do số lần xuất hiện của chúng quá nhiều mà lại không mang nhiều ý nghĩa ngữ nghĩa. Đồng thời, khi loại bỏ stopwords, ta có thể giảm thiểu số lượng term, từ đó khiến cho tốc độ xử lí của mô hình nhanh hơn nhiều.

Trước	Sau
"... elements (spheres) **on** boundary layer transition **on** a tained at local mach numbers **of** 1.90, 2.71, **and** 3.67 **by** varying trip size, position, spacing, and reynolds number per inch. **the** results indicate **that** (1) transition …"	"... elements (spheres) boundary layer transition tained local mach numbers 1.90, 2.71, 3.67 varying trip size, position, spacing, reynolds number per inch. results indicate (1) transition …"

Bảng 2 – Ví dụ kết quả loại bỏ stopword

c. PorterStemer và Lemmatizer

Tiếp theo là áp dụng Porter Stemmer/Lemmatizer cho từng từ đã được tách ra từ dữ liệu. Porter Stemmer và Lemmatizer là 2 kĩ thuật được sử dụng để chuyển đổi các từ vựng thành từ nguyên bản (từ gốc của nó). Thế nhưng, cách xử lí của chúng khá khác nhau:

Porter Stemmer	Lemmatizer
Áp dụng một tập các luật để loại bỏ các hậu tố thường gặp, thu gọn từ đó về dạng stem của nó. Ví dụ từ: "running" → "run"	Biến đổi từ đang xét thành phiên bản gốc của nó, dựa vào các đặc điểm ngữ pháp của nó. Ví dụ từ: "running" → "run"
Tập trung vào các quy tắc ngôn ngữ để cắt tỉa từ ngữ. Ví dụ từ: "better" → "better"	Ánh xạ từ đang xét về phiên bản trong từ điển của nó. Ví dụ từ: "better" → "good"
Không phải lúc nào cũng sẽ trả về một từ trong từ điển. Ví dụ từ: "universities" → "univers"	Biến đổi danh từ số nhiều thành phiên bản danh từ số ít của nó. Ví dụ từ: "universities" → "university"

Bảng 3 – So sánh PorterStemmer và Lemmatizer

Nhìn chung, Porter Stemmer tập trung vào việc áp dụng các quy tắc ngôn ngữ để biến đổi các từ vựng về phiên bản gốc của chúng, trong khi Lemmatizer sẽ tập trung vào các đặc điểm ngữ pháp của từ đang xét để biến đổi chúng thành phiên bản gốc hoặc phiên bản có trong từ điển. Porter Stem có thể sẽ biến đổi từ vựng thành một phiên bản không thực sự là từ ngữ trong khi Lemma sẽ thường biến đổi ra thành các từ có trong từ điển tiếng Anh.

Ví dụ: Áp dụng Porter Stemmer và loại bỏ Stopwords ở trong doc 7.txt

```
experiments were performed in the 12 in. supersonic wind tunnel
of the jet propulsion laboratory of the california institute of
technology to investigate the effect of three dimensional
roughness elements (spheres) on boundary layer transition on a
tained at local mach numbers of 1.90, 2.71, and 3.67 by varying
```

```
trip size, position, spacing, and reynolds number per inch. the
results indicate that (1) transition from laminar to turbulent
flow induced by three dimensional roughness elements begins when
the double row of spiral vortices trailing each element
contaminates and breaks down the surrounding field of vorticity,
```

Khi ta đã lọc ra được danh sách các từ xuất hiện ở trong document 7, ta có thể thấy được rằng số lượng từ vựng khá nhiều và có chứa các từ không thực sự quá cần thiết/chưa được rút gọn hoàn toàn. Vì vậy, nhóm em sẽ áp dụng Porter Stemmer và loại bỏ đi Stopwords khỏi danh sách các term vừa tìm được:

- Với PorterStemmer ta có được danh sách các term gồm 62 term:

```
['experi','perform','12','superson','wind','tunnel','jet','pro
puls','laboratori','california','institut','technolog','invest
ig','effect','three','dimension','rough','element','sphere','b
oundari','layer','transit','tain','local','mach','number','190
','271','367','vari','trip','size','posit','space','reynold','
number','per','inch','result','indic','1','transit','laminar',
'turbul','flow','induc','three','dimension','rough','element',
'begin','doubl','row','spiral','vortic','trail','element','con
tamin','break','surround','field','vortic']
```

- Và phiên bản Lemmatizer với 62 term:

```
['experiment','performed','12','supersonic','wind','tunnel','j
et','propulsion','laboratory','california','institute','techno
logy','investigate','effect','three','dimensional','roughness'
,'element','sphere','boundary','layer','transition','tained','
local','mach','number','190','271','367','varying','trip','siz
e','position','spacing','reynolds','number','per','inch','resu
lt','indicate','1','transition','laminar','turbulent','flow','
induced','three','dimensional','roughness','element','begin','
double','row','spiral','vortex','trailing','element','contamin
ates','break','surrounding','field','vorticity']
```

d. Kết quả

Sau khi áp dụng các phương pháp tiền xử lí khác nhau, nhóm thu được kết quả như sau:

Phương pháp tiền xử lí	Số lượng term
Không tiền xử lí	7908
Porter Stemmer	4665
Porter Stemmer + Remove Stopwords	4556
Lemmatizer	6537
Lemmatizer + Remove Stopwords	6424

Bảng 4 – Kết quả quá trình tiền xử lý tập Cranfield

Phương pháp tiền xử lí	Số lượng term
Không tiền xử lí	28286
Porter Stemmer	21001
Porter Stemmer + Remove Stopwords	20978
Lemmatizer	25946
Lemmatizer + Remove Stopwords	25925

Bảng 5 – Kết quả quá trình tiền xử lý tập NFCorpus

Việc áp dụng Porter Stemmer và Lemmatizer giúp chúng ta chuẩn hóa lại các tài liệu của mình, từ đò giảm thiểu số lượng các từ vựng sẽ xuất hiện ở trong tập các term của chúng ta và giúp cho việc tìm kiếm, xem xét trở nên nhanh hơn, chính xác hơn thay vì không áp dụng chúng.

III. Lập chỉ mục

1. Vector Space Model (VSM)

a. Giới thiệu mô hình

Mô hình Vector Space là một mô hình toán học được sử dụng rộng rãi trong bài toán Truy xuất thông tin bằng cách biến đổi các tài liệu và câu truy vấn thành dạng vector với không gian chiều lớn. Trong mô hình này, mỗi term của tài liệu hoặc query sẽ được đánh trọng số, thường dựa vào tf (term frequency), idf

(inverse document frequency) và TF-IDF để đo lường tần suất và tầm quan trọng của các term.

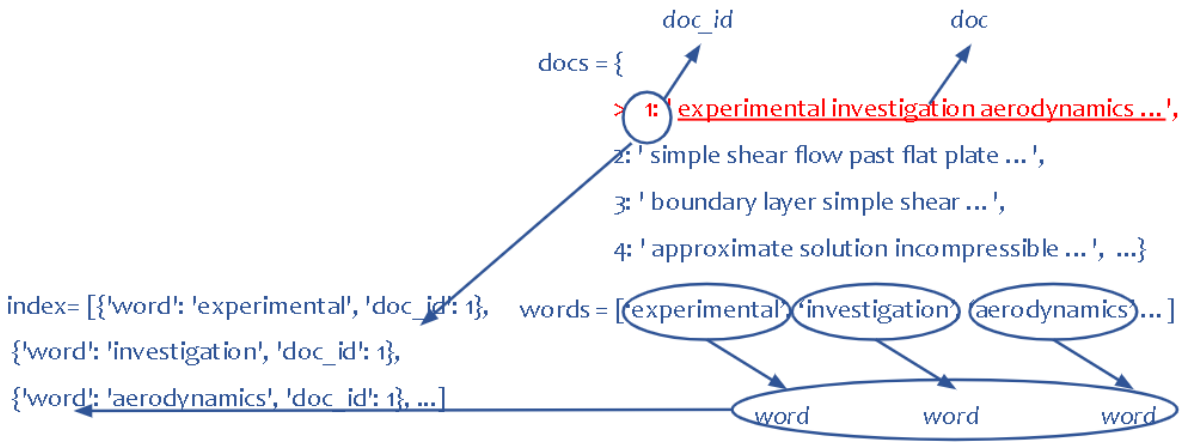
Quá trình tìm kiếm và xác định mức độ liên quan của tài liệu tới câu truy vấn được thực hiện bằng cách tính toán độ tương đồng giữa các vectơ này. Các phương pháp tính độ tương đồng thường sử dụng trong VSM bao gồm độ đo cosine và độ đo Euclidean.

b. Các bước tiến hành

- Indexing: Tạo tập từ điển tên là index từ tập tài liệu docs, mỗi từ của mỗi tài liệu sẽ được tách ra và gán doc_id của tài liệu đó.
    - Input: dictionary docs có dạng {doc_id: doc}, trong đó:
        - doc_id là đánh số của tập tài liệu
        - doc là nội dung của tập tài liệu
    - Output: mảng index chứa các dictionary có dạng {word:"…",doc_id:"…"} trong đó mỗi word là 1 từ trong tập tài liệu có doc_id tương ứng kèm theo, word có thể lặp lại do nhiều tài liệu có thể chứa cùng 1 từ và từ đó cũng có thể xuất hiện nhiều lần trong 1 tài liệu
    - Mã giả:

```
for doc_id, doc in docs.items():
        words = list of words in doc[doc_id]
        for each word in words:
                add{word,doc_id}to index
```

    - Ví dụ: cho tập tài liệu "docs", đối với mỗi tài liệu trong tập, thực hiện tách từ và gán doc_id của tài liệu tương ứng vào với từ đó

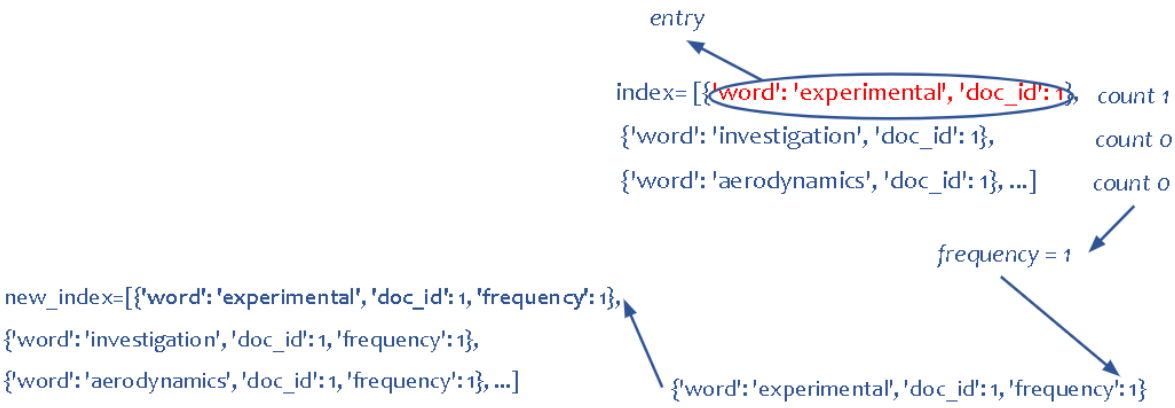


Hình 2 – Ví dụ indexing VSM

- Adding frequencies: thu gọn tập index và thêm cột frequency vào tập index để thể hiện số lần xuất hiện của từ đó trong tập tài liệu tương ứng nếu có. Từ đây thì một từ chỉ được ánh xạ một lần qua tập index trong một tài liệu tương ứng nếu có và kèm theo tần suất xuất hiện của nó
  - Input: mảng index ở trên
  - Output: mảng index mới chứa các dictionary có dạng {word:"…",doc_id:"…",frequency:"…"} trong đó mỗi word là 1 từ trong tập tài liệu có doc_id tương ứng kèm theo và có số lần xuất hiện trong tài liệu đó là frequency
  - Mã giả:

```
new_index = []
for each entry in index:
        frequency = count appear time of entry['word'] in entry['doc_id']
        add frequency to entry['frequency']
        add entry to new_index
```

  - Ví dụ: với mỗi entry trong tập index, đếm số lần xuất hiện của nó trong 1 tài liệu và thêm tần số xuất hiện vào sau dictionary chứa nó

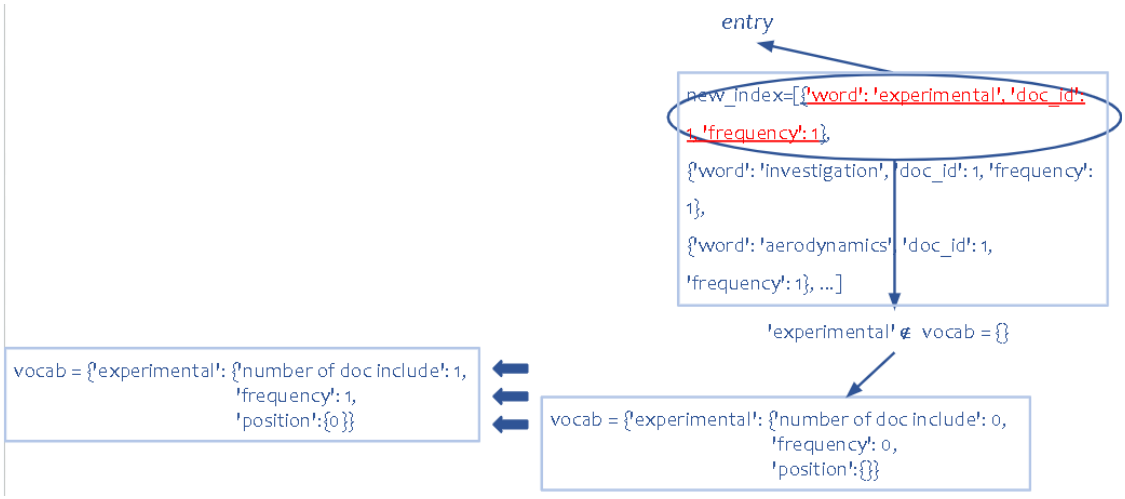


Hình 3 – Ví dụ Adding frequencies VSM

- Vocab: xây dựng tập từ vựng từ tập index
  - Input: mảng index ở trên
  - Output: dictionary vocab có dạng {word:{number of doc include':…, 'frequency': …, 'position':{…}}} trong đó word là term trong bộ dữ liệu, number of doc include là số lượng tài liệu chứa term đó, frequency là số lần xuất hiện của term đó trong toàn bộ tập tài liệu, position là danh sách các vị trí của từ đó trong tập index (chỉ số của mảng)
  - Mã giả:

```
vocab = {}
for i, entry in enumerate(new_sorted_indexs):
        if entry['word'] ∉ vocab:
                add {'number of doc include': 0, 'frequency': 0, 'position':
                set()} to vocab[word]
        Count num of docs with entry['word'] increase 1
        Increase num of frequency in docs have entry['word']
        Add position of entry['word'] to ['position']
```

  - Ví dụ: với mỗi entry trong tập index mới nếu chưa xuất hiện trong tập vocab, thêm nó vào tập vocab, nếu có thì đếm số tài liệu chứa nó, số lần xuất hiện và lấy danh sách vị trí trong tập index của nó



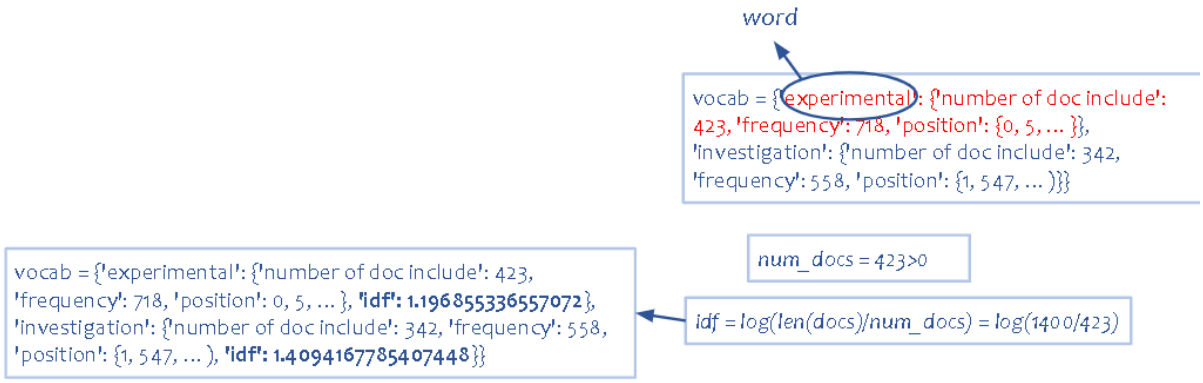
Hình 4 – Ví dụ xây dựng tập Vocab VSM

- Trước khi xác định được công thức tf và idf sẽ được dùng trong bài này, nhóm đã tiến hành thử nghiệm với các công thức tính tf, idf và $norm$ có trong slide, sau đó chọn ra tổ hợp có điểm số cao nhất để thực hiện trong bài báo cáo. Cụ thể:
  - $tf = \{0,1\}$ nhị phân
  - $idf = \log\left(\frac{NDoc}{Doc_k}\right)$: nghịch đảo
  - $w = tf * idf$
  - $norm = \sqrt{\sum_{vector} w_j^2}$: chuẩn cosine
- Tính idf: mức độ quan trọng của một term trong toàn bộ cơ sở dữ liệu
  - Công thức:
    - $idf = \log\left(\frac{total\ docs}{num\ docs}\right)$
    - Trong đó total_docs là tổng số tài liệu và num_docs là số lượng tài liệu chứa từ vựng
  - Input: tập vocab ở trên
  - Output: tập vocab mới có dạng {word:{ 'number of doc include':…, 'frequency': …, 'position':{…}}, 'idf':…} trong đó idf là giá trị idf của từ đó vừa tính được

- Mã giả

```
for each word in vocab:
    num_docs = vocab[word]['number of doc include']
    if num_docs > 0:
        idf = log(total_docs/num_docs)
        set vocab[word]['idf'] to idf
```

- Ví dụ: với mỗi từ trong tập vocab, nếu có tài liệu chứa nó thì tính idf bằng cách lấy tổng số tài liệu chia cho số tài liệu chứa nó và gán vào idf trong tập vocab

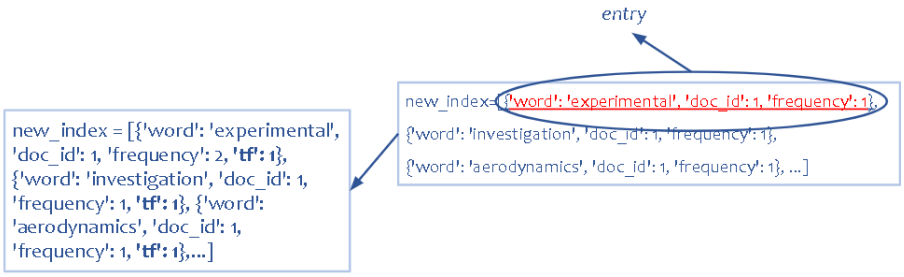


Hình 5 – Ví dụ Tính idf VSM

- Tính TF: sử dụng TF nhị phân, trong đó mỗi từ trong danh sách được coi là xuất hiện một lần duy nhất trong tài liệu, nên giá trị TF của từ đó là 1. Bằng cách gán giá trị 1 cho trường TF của mỗi mục, ta chỉ đơn giản đánh dấu rằng từ đó xuất hiện trong tài liệu
  - Input: mảng index ở trên
  - Output: mảng index mới chứa các dictionary có dạng {word:"…",doc_id:"…",frequency:"…", tf:"…"} trong đó thêm vào giá trị tf mới tính được
  - Mã giả:

```
for each entry in new_sorted_indices:
    set entry['tf'] to 1
```

- Ví dụ: với mỗi entry trong tập index mới, gán giá trị tf = 1 và thêm vào tập index

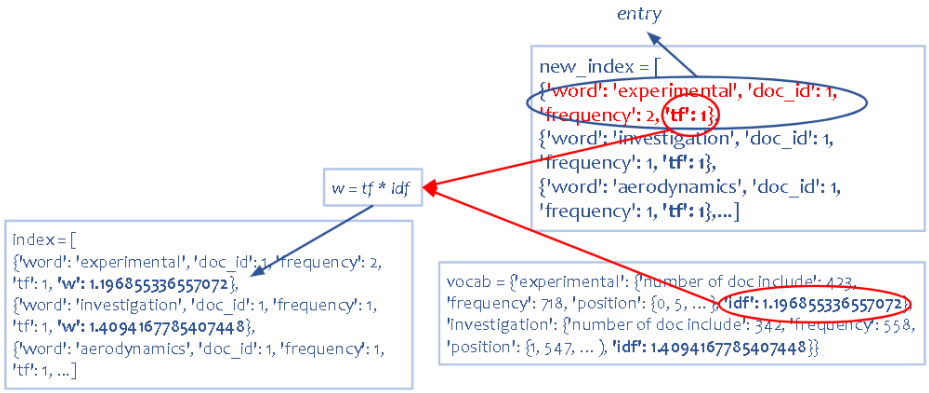


Hình 6 – Ví dụ tính tf VSM

- Tính trọng số Weights bằng tích của tf và idf ở trên
  - Công thức: $w = tf * idf$
  - Input: tập index và vocab ở trên
  - Output: mảng index mới chứa các dictionary có dạng {word:"…",doc_id:"…",frequency:"…", tf:"…",w:"…"} trong đó w là giá trị weight vừa tìm được
  - Mã giả

```
for entry in index(tf):
        w = vocab[entry['word']]['idf'] * entry['tf']
        entry['w'] ← w
```

  - Ví dụ: với mỗi entry trong tập index, tính trọng số bằng cách lấy tf của nó nhân với idf của từ tương ứng trong tập vocab rồi gán lại vào tập index

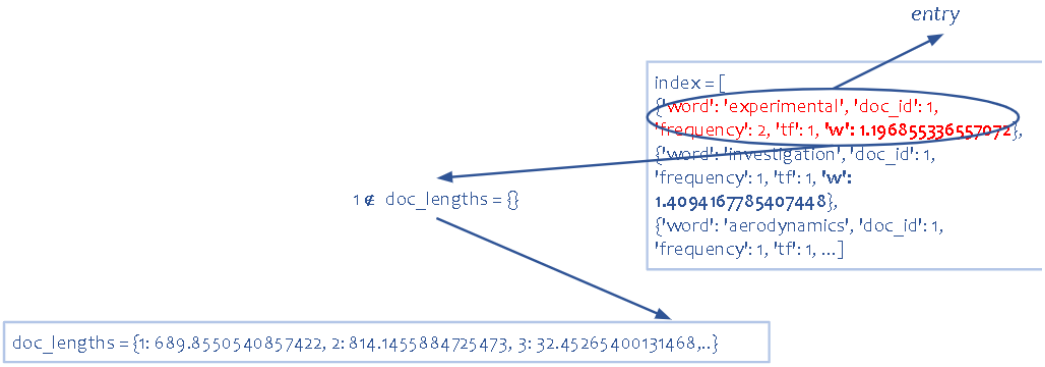


Hình 7 – Ví dụ Tính trọng số Weights VSM

- Chuẩn hóa các trọng số (weights) dựa trên độ dài của tài liệu (doc) trong mô hình Cosine.
    - Công thức: $cosine(A,B) = \frac{A*B}{\|A\|*\|B\|}$
    - Input: tập index ở trên
    - Output: mảng index mới chứa các dictionary có dạng {word:"…",doc_id:"…",frequency:"…", tf:"…",w:"…"} trong đó cập nhật lại giá trị weight đã được chuẩn hóa
    - Mã giả:

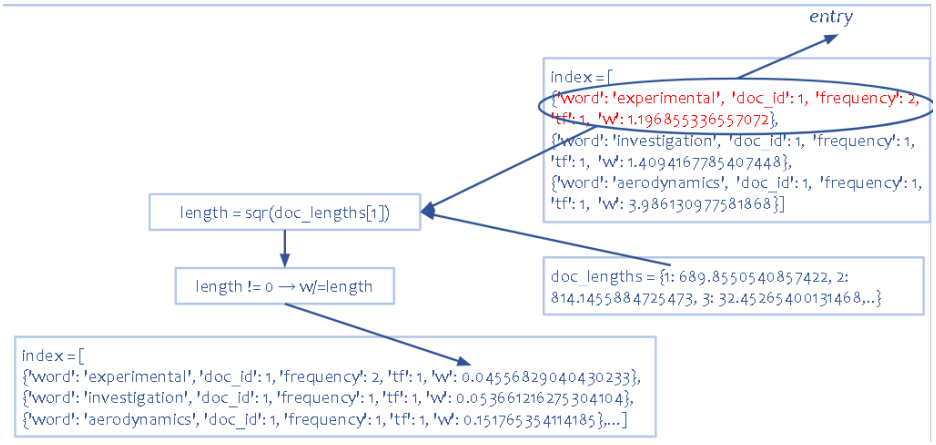
```
doc_lengths = {}
for each entry in new_tf_sorted_indices:
        if entry['doc_id'] ∈ doc_lengths:
                doc_lengths[entry['doc_id']]+=entry['w']^2
        else:
                doc_lengths[entry['doc_id']]=entry['w']^2
for each entry in new_tf_sorted_indices:
        length = sqrt(doc_lengths[entry['doc_id']])
        entry['w']=length!=0 ? (entry['w']/length) : 0
```

    - Ví dụ:
        - Thêm danh sách doc_id vào dictionary doc_lengths, ứng với mỗi doc_id là tổng giá trị trọng số w của các từ mà nó chứa trong tập index



Hình 8 – Ví dụ Chuẩn hóa các trọng số (a) VSM

- Với mỗi entry trong tập index, lấy w chia cho căn bật 2 của doc_length tài liệu chứa nó và gán lại cho w của tập index



Hình 9 – Ví dụ Chuẩn hóa các trọng số (b) VSM

2. Latent Semantic Index (LSI)

a. Giới thiệu mô hình

Mô hình LSI: biểu diễn các tài liệu và câu truy vấn bằng cách tìm ra những mối quan hệ ngữ nghĩa tìm ẩn giữa các thuật ngữ (term).

Các tài liệu sẽ được biểu diễn dưới dạng ma trận term-doc, với hàng là các thuật ngữ (term) và cột là các tài liệu. Ma trận này sẽ biểu diễn số lần xuất hiện của term trong doc hoặc liệu term đó có xuất hiện trong doc hay không

SVD sẽ được áp dụng lên ma trận Term-Doc dựa trên độ dài của câu truy vấn, từ đó tách ra 3 ma trận khác nhau:

- Ma trận S: thể hiện tầm quan trọng hoặc ý nghĩa của các concept (khái niệm) trong tài liệu
- Ma trận U: là ma trận suy biến trái với mỗi cột đại diện cho một term tong tập tài liệu; ma trận này ghi nhận lại các mối quan hệ giữa các term và concept
- Ma trận ZT: là ma trận suy biến phải với mỗi hàng đại diện cho một tài liệu (doc) trong tập tài liệu; ma trận này ghi nhận lại các mối quan hệ giữa các term và concept

Độ tương đồng giữa câu truy vấn và các tài liệu được xác định bằng các kĩ thuật như độ tương đồng Cosine, … Các tài liệu sẽ được xếp hạng dựa trên độ tương

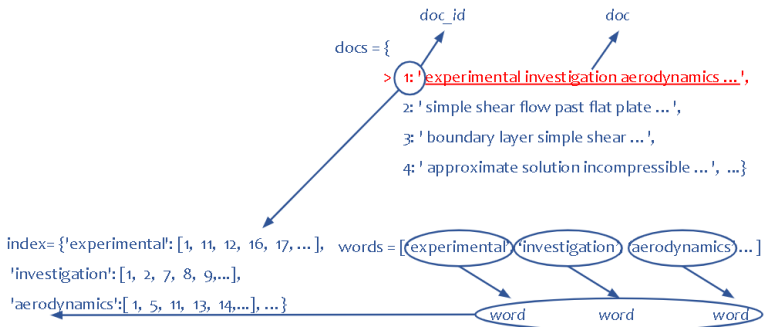
đồng của chúng với câu truy vấn, với các tài liệu xếp hạng cao được cho là có liên quan hơn.

b. Các bước tiến hành

- Indexing: Tạo tập từ điển tên là index từ tập tài liệu docs, các term sẽ được đi kèm với danh sách doc_id có chứa term đó.
    - Input: dictionary docs có dạng {doc_id: doc}, trong đó:
        - doc_id là đánh số của tập tài liệu
        - doc là nội dung của tập tài liệu
    - Output: dictionary index có dạng {word:[doc_id_1,doc_id_2,…]} trong đó mỗi word là 1 từ trong tập tài liệu kèm theo danh sách doc_id tương ứng chứa nó. Mỗi word là duy nhất
    - Mã giả

```
index = {}
for doc_id, doc in docs.items():
        words = list of words in doc[doc_id]
        foreach word in words:
                if word in index:
                        add doc_id to word
                else:
                        add word to index
                        add doc_id to word
```

    - Ví dụ: cho tập tài liệu “docs”, đối với mỗi tài liệu trong tập, thực hiện tách từ và thêm doc_id của tài liệu tương ứng vào với danh sách

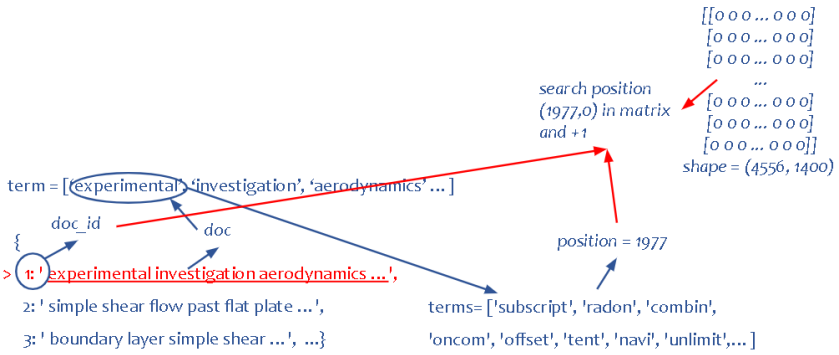


Hình 10 – Ví dụ indexing LSI

- Tạo ma trận term-doc với giá trị mỗi ô là tần suất xuất hiện của 1 term trong 1 doc
  - Input:
    - List terms chứa danh sách term
    - Dictionary docs có dạng {doc_id: doc}, trong đó: doc_id là đánh số của tập tài liệu và doc là nội dung của tập tài liệu
  - Output: ma trận term-doc frequency_matrix với kích thước len(terms)*len(docs)
  - Mã giả:

```
frequency_matrix = zeros matrix(len(terms),len(docs))
for each doc_id, doc in docs:
        for each term in doc.split():
                term_index = get position of term in terms
                frequency_matrix[term_index][doc_id - 1] += 1
```

  - Ví dụ:

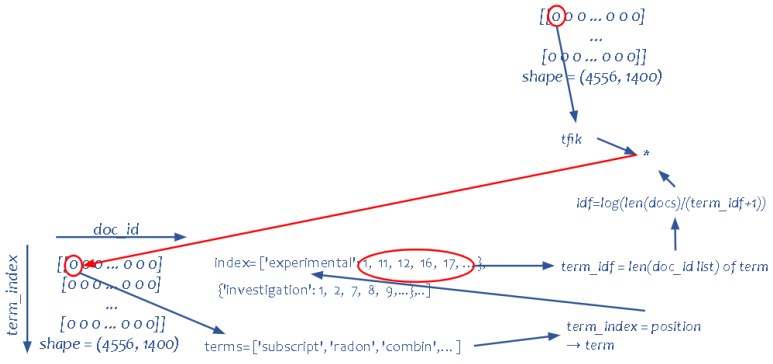


Hình 11 – Ví dụ Tạo ma trận term-doc LSI

- Tạo ma trận tf-idf: chuẩn hóa các giá trị trong ma trận term-doc
  - Công thức: $tf * idf$ với
    - tf là giá trị trong ma trận term-doc frequency_matrix
    - $idf = \log\left(\frac{len(docs)}{len(doc\ id\ list) of\ term+1}\right)$
  - Input:
    - Ma trận term-doc frequency_matrix với kích thước len(terms)*len(docs)
    - List terms chứa danh sách term
  - Output: ma trận term-doc tf_idf_matrix với kích thước len(terms)*len(docs) đã được cập nhật lại các giá trị
  - Mã giả

```
tf_idf_matrix = zeros matrix(len(terms),len(docs))
frequency_matrix = build_frequency_matrix
for doc_id in range(1, len(docs)+1):
        for term_index in range(len(terms)):
                term = get term from term_index
                term_idf = get num of docs have term in index
                idf = log(len(docs)/(1 + term_idf))
                tfik = frequency_matrix[term_index][doc_id - 1]
                tf_idf_matrix[term_index][doc_id - 1] = tfik * idf
```

  - Ví dụ:



Hình 12- Ví dụ Tạo ma trận tf-idf LSI

IV. Xử lý truy vấn

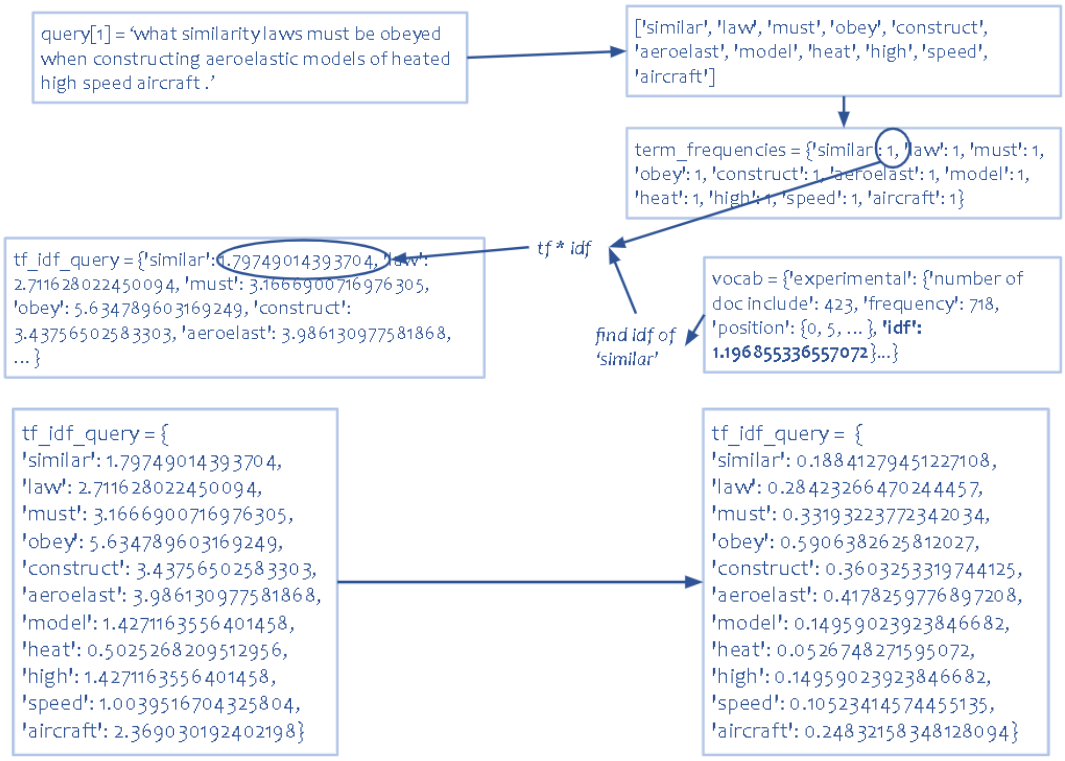
1. Vector Space Model

- Input:
  - Câu truy vấn
  - Tập vocab ở trên
- Output: dictionary tf_idf_query có dạng {word:w} với word là các term trogn query và w là trọng số sau khi đã chuẩn hóa như docs
- Tiền xử lý query như tiền xử lý docs:
  - Loại bỏ ký tự đặc biệt
  - Loại bỏ stopword
  - Tách từ
  - PorterStemmer/Lemmatizer
  - Tính tf của term trong query như của docs
  - Tìm idf của term trong query từ tập vocab
  - Tính $tf * idf$
  - Chuẩn hóa các trọng số của query bằng cosine như với docss
- Mã giả

```
preprocessing query
tf_idf_query = {}
term_frequencies = {}
for each term in query:
        increment term_frequencies[term] by 1
for each term, frequency in term_frequencies:
        if term is in order_term:
                tf = frequency
                idf = new_idf_vocab[term]['idf']
                tf_idf_query[term] = tf * idf
tf_idf_norm = sqrt(sum of sqr(values in tf_idf_query))
```

normalize tf_idf_query = each value/tf_idf_norm

- Ví dụ:



Hình 13 – Ví dụ xử lý truy vấn theo VSM

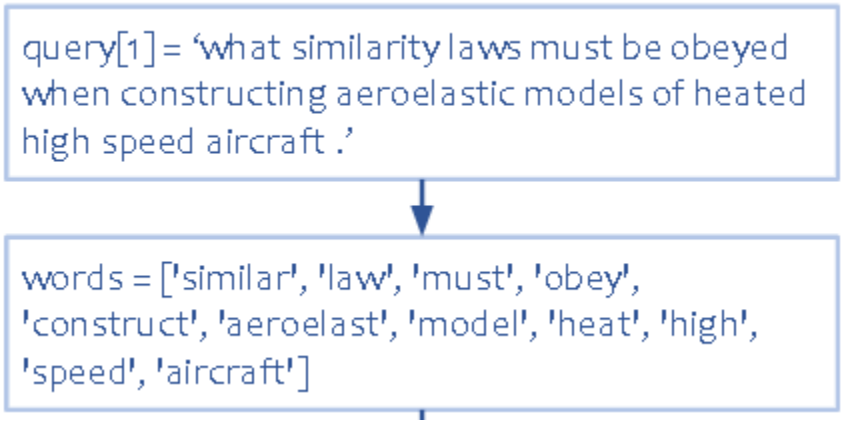
2. Latent Semantic Index

- Input: câu truy vấn
- Output: list words gồm các từ được tách ra từ câu truy vấn sau khi tiền xử lý và độ dài của list (len_query)
- Tiền xử lý query như tiền xử lý docs:
  - Loại bỏ ký tự đặc biệt
  - Loại bỏ stopword
  - Tách từ
  - PorterStemmer/Lemmatizer
- Mã giả

```
preprocessing query
words = extract_words_from_query(query)
```

```
len_query = length(words)
```

- Ví dụ:



Hình 14 – Ví dụ xử lý truy vấn LSI

V. Thực hiện truy xuất

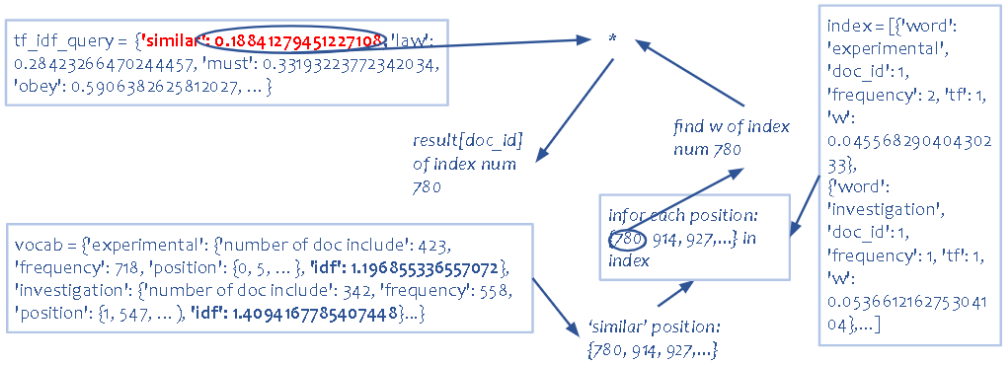
1. Vector Space Model

- Input:
  - Dictionary tf_idf_query phần tiền xử lý truy vấn
  - Tập vocab ở trên
  - Tập index ở trên
- Output: dictionary result có dạng {cosine_similarity:doc_id} được xếp theo thứ tự giảm dần của giá trị cosine_similarity, tức là càng xuống thấp thì độ tương đồng giữa tài liệu và câu truy vấn càng thấp
- Các bước:
  - Duyệt qua từng term tf_idf_query
  - Lấy danh sách vị trí xuất hiện của từng term bằng cách ánh xạ qua tập vocab
  - Với từng vị trí trong danh sách, ánh xạ qua tập index để lấy trọng số w tương ứng
  - Nhân w vừa tìm được với w của tf_idf_query term đó và thêm điểm vào cho doc_id tương ứng
  - Sắp xếp lại theo thứ tự

- Mã giả

```
result = create empty dictionary
for each term, query_weight in tf_idf_query:
        positions = new_idf_vocab[term]['position']
        for each position in positions:
                entry = new_w_sorted_indexs[position]
                retri_score = query_weight * entry['w']
        if entry['doc_id']∈ result:
                result[entry['doc_id']]+= retri_score
        else:
                result[entry['doc_id']] = retri_score
sort result.items()
```

- Ví dụ:



Hình 15 – Ví dụ thực hiện truy vấn theo VSM

2. Latent Semantic Index

- Input:
  - Ma trận term-doc tf_idf_matrix với kích thước len(terms)*len(docs)
  - List words và len của list từ phần xử lý truy vấn
  - List terms chứa danh sách term
  - Dictionary docs có dạng {doc_id: doc}, trong đó: doc_id là đánh số của tập tài liệu và doc là nội dung của tập tài liệu

- Output: dictionary result có dạng {cosine_similarity:doc_id} được xếp theo thứ tự giảm dần của giá trị cosine_similarity, tức là càng xuống thấp thì độ tương đồng giữa tài liệu và câu truy vấn càng thấp
- Các bước:
    - Tính ba ma trận s, z và ut từ ma trận tf-idf đã tìm được ở trên bằng phương thức svd của thư viện numpy

```
1 s
```

```
array([[-8.65958587e-04,  1.00832250e-03, -5.69349111e-04, ...,
        -5.83189762e-03,  2.12993843e-03,  1.46086720e-03],
       [-5.00362400e-04, -7.44945274e-06,  1.09137690e-03, ...,
        -9.97654397e-04, -3.09798547e-03,  1.07450894e-03],
       [-4.01751281e-02, -1.16289471e-02,  7.72449241e-02, ...,
         1.17393231e-03, -5.81795651e-03,  2.04808244e-04],
       ...,
       [-8.30891713e-04,  9.03143443e-04,  1.13839709e-03, ...,
         8.95492823e-01, -6.42173743e-04,  1.92768611e-03],
       [-6.18282554e-04,  5.62045581e-04, -1.15639918e-03, ...,
         1.45705495e-03,  9.10633930e-01,  1.62630671e-03],
       [-1.19495598e-03, -5.14741299e-03, -2.78667463e-03, ...,
         1.09362684e-04,  1.29695317e-03,  9.56812490e-01]])
```

```
1 z
```

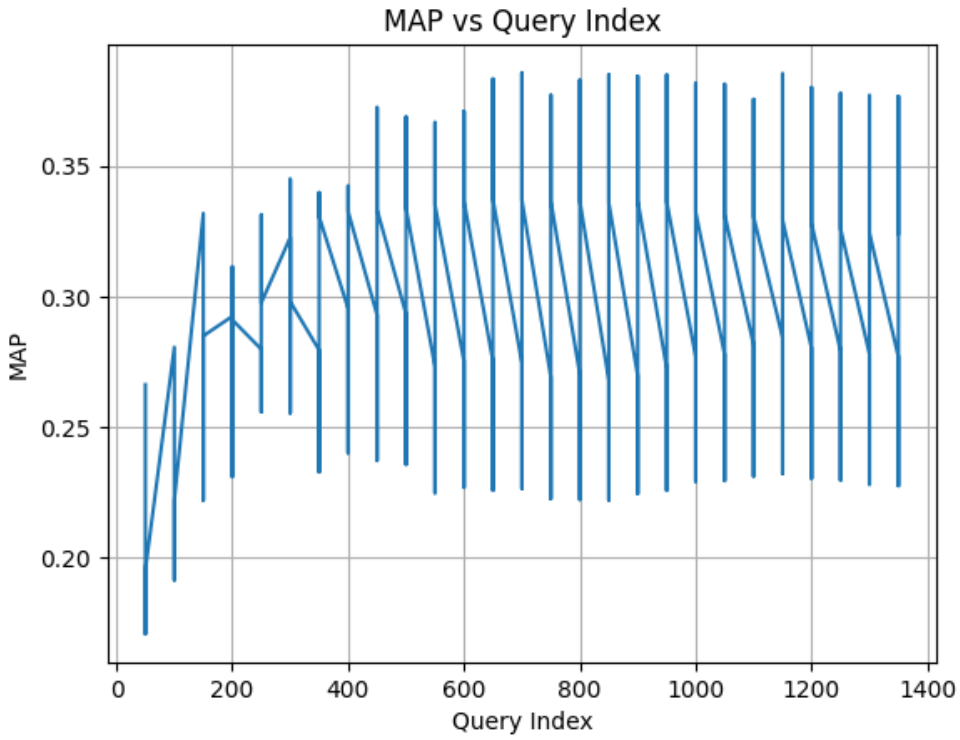
```
array([1.50644488e+02, 8.68339565e+01, 8.34845314e+01, ...,
       1.06719607e+00, 7.92618417e-15, 4.08695893e-15])
```

```
1 ut
```

```
array([[-2.11235223e-02, -3.10273171e-02, -5.03577421e-03, ...,
        -2.73527003e-02, -1.06030138e-02, -1.38151725e-02],
       [-1.25425780e-02,  2.05223395e-02,  5.21935419e-03, ...,
         1.55616545e-02,  9.45987985e-03,  1.16064057e-02],
       [ 3.25252266e-02, -1.99237375e-02, -5.57844387e-03, ...,
         2.99379742e-02,  6.23386974e-03,  1.83251884e-02],
       ...,
       [ 8.78329316e-04, -2.31683757e-02,  9.39631109e-01, ...,
        -8.73840925e-03, -1.07112583e-02,  4.42978829e-03],
       [ 0.00000000e+00,  5.97238973e-17, -7.07265610e-18, ...,
        -4.39303876e-17,  5.89289670e-17,  3.33389214e-17],
       [ 2.30323424e-37,  1.08889615e-16,  4.67771116e-18, ...,
         5.56074008e-17, -6.91143975e-17,  3.85493290e-17]])
```

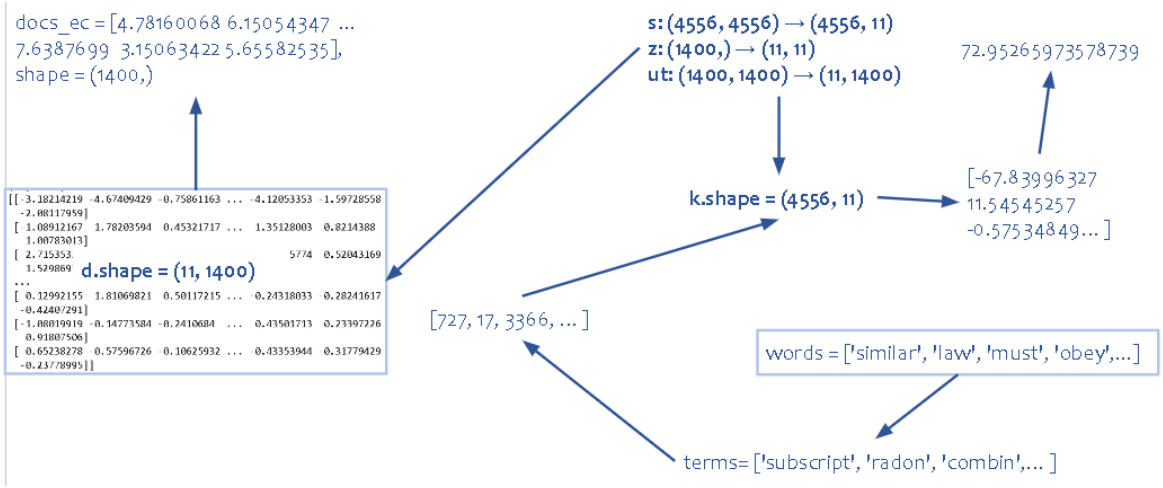
Hình 16 – Ba ma trận S, Z và UT

    - Cắt ba ma trận với số chiều bằng 700, do trước đó nhóm đã thực hiện khảo sát với 27 điểm dữ liệu từ 100 đến 1400 với số bước là 50 trên tập Lemma + Stop và lấy số chiều có kết quả cao nhất để thực hiện cho toàn bộ các tập khác trước khi tiến hành so sánh.



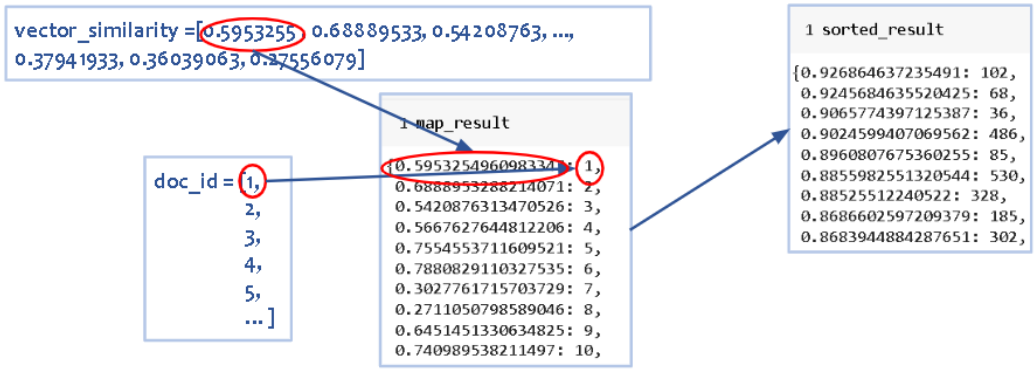
Hình 17 – Kết quả thử nghiệp với các chiều khác nhau

- Lần lượt tính ma trận k và d từ 3 ma trận trên
- Chuẩn hóa ma trận d
- Suy ra các vị trí các term của words bằng cách ánh xạ qua list term rồi lấy các dòng đó từ ma trận k ra sau đó tính tổng theo trục 0
- Chuẩn hóa sau đó tính cosine similarity



Hình 18 – ví dụ ba bước trên

- Nối kết quả cosine tương ứng với doc_id rồi sắp xếp giảm dần, ta có kết quả truy vấn cuối cùng



Hình 19 – Sắp xếp kết quả

- Mã giả

```
s, z, ut = np.linalg.svd(tf_idf_matrix)
s_truncated, z_truncated, ut_truncated  = truncate_matrix == 700
k = s_truncated @ z_truncated
d = z_truncated @ ut_truncated
docs_ec = np.linalg.norm(d, axis=0)
order = get position of word in term
query_arr = get value of each order in k and sum by axis 0
query_ec = np.linalg.norm(query_arr)
results = calculate_cosine_similarity(query_arr, d, docs_ec, query_ec)
map_result = {}
for result, doc_id in zip(vector_similarity, docs):
        map_result[result] = doc_id
sorted_result = map_result by relevant score
```

VI. Đánh giá mô hình

1. Các phương pháp đánh giá

a. P và R

- Precision (P) là tỷ lệ giữa số lượng kết quả đúng được trả về và tổng số kết quả được trả về. Công thức: $P = \frac{Số\ lượng\ kết\ quả\ đúng}{Tổng\ số\ kết\ quả\ được\ trả\ về}$

- Recall (R) là tỷ lệ giữa số lượng kết quả đúng được trả về và tổng số kết quả thực tế có sẵn. Công thức: $R = \frac{Số\ lượng\ kết\ quả\ đúng}{Tổng\ số\ kết\ quả\ thực\ tế}$

b. MAP nội suy

MAP nội suy là một phương pháp đánh giá đo lường hiệu suất của hệ thống truy xuất thông tin dựa trên sự xếp hạng của các kết quả trả về.

- Đầu tiên, tính toán Average Precision (AP) cho mỗi truy vấn. AP đo lường mức độ chính xác của hệ thống khi truy vấn được xếp hạng.
- Sau đó, tính toán trung bình của tất cả các AP để đạt được MAP. MAP nội suy là một phép nội suy giữa các AP.

2. Kết quả đánh giá

- Quá trình sử dụng VSM và Whoosh đối với tập Cranfield và tập NFCorpus diễn ra bình thường, tuy nhiên đối với LSI, chỉ có tập Cranfield thực hiện được còn tập NFCorpus thì không do tràn RAM trong quá trình lập chỉ mục nên các kết quả và nhận xét sẽ lấy phần lớn từ kinh nghiệm thực hiện đối với tập Cranfield, còn tập NFCorpus chỉ mang tính chất tham khảo.
- Kết quả của 11 điểm nội suy theo TREC

	Vector	LSI	Whoosh
0	0.7198	0.6740	0.2133
1	0.6962	0.6351	0.1865
2	0.5814	0.5163	0.1334
3	0.4737	0.4250	0.0753
4	0.3874	0.3484	0.0472
5	0.3348	0.3077	0.0353
6	0.2610	0.2381	0.0240
7	0.1907	0.1861	0.1321
8	0.1500	0.1533	0.0051
9	0.1100	0.1158	0.0011

10	0.0972	0.1062	0.0011

- Kết quả tổng hợp

	Vector			LSI			Whoosh		
	P	R	MAP	P	R	MAP	P	R	MAP
Không xử lí	0.0060	0.9961	0.0959	0.0058	1	0.3032	-	-	-
Stem	0.0060	0.9990	0.1054	0.0058	1	0.0167	0.0131	0.3519	0.0603
Lemma	0.0060	0.9992	0.1031	0.0058	1	0.2918	-	-	-
Stem + Bỏ Stopword	0.0096	0.9615	0.3638	0.0058	1	0.2119	0.0172	0.3719	0.0669
Lemma + Bỏ Stopword	0.0160	0.9573	0.3600	0.0058	1	0.3369	-	-	-

Bảng 6 – Kết quả tổng hợp của tập Cranfield

	Vector			Whoosh		
	P	R	MAP	P	R	MAP
Không xử lí	0.0167	0.9305	0.1629	-	-	-
Stem	0.0162	0.9534	0.1661	0.0183	0.8918	0.1712
Lemma	0.0163	0.9486	0.1676	-	-	-
Stem + Bỏ Stopword	0.0162	0.9529	0.2049	0.0184	0.8928	0.1749
Lemma + Bỏ Stopword	0.0163	0.9492	0.2052	-	-	-

Bảng 7 – Kết quả tổng hợp của tập NFCorpus

- Kết quả tổng hợp về thời gian và không gian

	Vector			LSI		
	Số lượng từ	Thời gian tạo chỉ mục	Thời gian truy vấn	Kích thước ma trận	Thời gian tạo chỉ mục	Thời gian truy vấn
Không xử lí	7908	1.6690	5.1733	(7908,1400)	84.4479	132.9668
Stem	4665	4.9419	5.1728	(4665,1400)	44.7292	74.8780

Lemma	6537	2.5258	5.0897	(6537,1400)	65.2688	101.5122
Stem + Bỏ Stopword	4556	3.1581	0.5524	(4556,1400)	42.4421	73.7299
Lemma + Bỏ Stopword	6424	3.2120	0.8435	(6424,1400)	61.5121	104.8263

Bảng 8 – Kết quả tổng hợp về thời gian và không gian của tập Cranfield

	Vector		
	Số lượng từ	Thời gian tạo chỉ mục	Thời gian truy vấn
Không xử lí	28286	4.3367	72.3422
Stem	21001	11.5396	138.7916
Lemma	20978	5.8448	87.2862
Stem + Bỏ Stopword	25946	14.1561	125.0903
Lemma + Bỏ Stopword	25925	7.7758	85.0072

Bảng 9 – Kết quả tổng hợp về thời gian và không gian của tập NFCorpus

3. Nhận xét kết quả

Về thời gian: thời gian tạo chỉ mục và tiến hành truy vấn của VSM rất nhanh chóng, gọn lẹ còn của LSI thì khá chậm và lâu (do áp dụng SVD để tách thành các ma trận con nhỏ hơn)

Về kết quả:

- VSM hoạt động tốt với cách tiền xử lý loại bỏ stopword kết hợp với PorterStemmer, bằng chứng là tổng thời gian tạo chỉ mục và truy vấn ít hơn so với các tổ hợp còn lại nhưng có kết quả MAP cao nhất
- LSI hoạt động tốt với cách tiền xử lý loại bỏ stopword kết hợp với Lemmatizer với kết quả truy vấn cao hơn các tổ hợp khác

Xem xét VSM và LSI, ta thấy kết quả của VSM cao hơn so với LSI, lý do có thể là tài liệu của bộ dữ liệu Cranfield có nội dung mang ý nghĩa chung hướng về khái niệm khoa học mà LSI lại biểu diễn các tài liệu và câu truy vấn bằng

cách tìm ra những mối quan hệ ngữ nghĩa tìm ẩn giữa các term nên việc nội dung hướng về một khái niệm gây khó khăn cho LSI hơn.

Bên cạnh đó, có một số câu truy vấn thuộc trường hợp các term của nó không tồn tại trong tập vocab của tài liệu nên VSM trả về kết quả truy xuất là một mảng rỗng, đó là lý do vì sao kết quả Recall của mô hình này đều không tới 1 mặc dù ta lấy hết các giá trị truy vấn được trả về. Về phần LSI, do tập trung vào ý nghĩa chứ không phải về sự xuất hiện của term như VSM nên LSI có thể trả về được các kết quả trong trường hợp đặc biệt này.

VII. Kết luận:

Qua đồ án trên, nhóm đã biết cách hiện thực hóa các mô hình truy xuất Vector Space Model và Latent Semantic Index: phân tích tài liệu, cách lập chỉ mục, xử lý truy vấn. Ngoài ra, nhóm còn biết thêm kiến thức về một số tập dữ liệu lớn trong bài toán truy xuất, có thêm kinh nghiệm trong việc áp dụng mô hình đã học để thử nghiệm và đánh giá trên tập dữ liệu có sẵn.