**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

****

**BÁO CÁO MÔN TRUY XUẤT THÔNG TIN**

**ĐỒ ÁN: THIẾT KẾ HỆ THỐNG TRUY VẤN THÔNG TIN TRÊN VĂN BẢN**

**Lớp: CS231.I21**

**Sinh viên thực hiện: Mai Quốc Kiệt – 15520400**

**Lê Thiện Duy – 15520158**

**Mục lục**

[Lời cảm ơn 3](#_Toc12389659)

[CHƯƠNG 1: GIỚI THIỆU 4](#_Toc12389660)

[1. Giới thiệu về truy xuất thông tin: 4](#_Toc12389661)

[2. Các ứng dụng của truy vấn thông tin đa phương tiện 4](#_Toc12389662)

[3. Giới thiệu về đồ án: 4](#_Toc12389663)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ 5](#_Toc12389664)

[1. Phân tích: 5](#_Toc12389665)

[2. Cấu trúc dữ liệu: 6](#_Toc12389666)

[3. Các phần mềm và thư viện cần thiết: 7](#_Toc12389667)

[CHƯƠNG 3: THỰC HIỆN 8](#_Toc12389668)

[1. Tiền xử lý dữ liệu: 8](#_Toc12389669)

[2. Xây dựng inverted index ban đầu: 9](#_Toc12389670)

[3. Tính toán các thông số còn lại: 9](#_Toc12389671)

[4. Xử lý query: 10](#_Toc12389672)

[5. Tính độ tương đồng 11](#_Toc12389673)

[6. Đánh giá hệ thống truy vấn: 12](#_Toc12389674)

[CHƯƠNG 4: KẾT QUẢ THỰC HIỆN 13](#_Toc12389675)

[1. Hướng dẫn sử dụng: 13](#_Toc12389676)

[2. Kết quả chạy chương trình: 13](#_Toc12389677)

[CHƯƠNG 5: KẾT LUẬN 14](#_Toc12389678)

[1. Kết quả đạt được: 14](#_Toc12389679)

[2. Hạn chế: 14](#_Toc12389680)

[3. Hướng phát triển: 14](#_Toc12389681)

[4. Phân công công việc: 14](#_Toc12389682)

[5. Biên bản làm việc nhóm: 14](#_Toc12389683)

[6. Tài liệu tham khảo: 15](#_Toc12389684)

[Link Github chứa đồ án của nhóm: 15](#_Toc12389685)

# Lời cảm ơn

Chúng em xin cảm ơn giảng viên hướng dẫn là thầy Nguyễn Trọng Chỉnh đã tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình học tập và thực hiện đồ án. Trong quá trình làm bài tập và đồ án, do thời gian và khả năng của bản thân còn hạn chế, nên chúng em không tránh khỏi sai sót. Vì vậy chúng em mong được sự bổ sung góp ý của thầy để hoàn thiện tốt hơn.

**Chúng em xin chân thành cảm ơn thầy và chúc thầy gặp nhiều thành công trong cuộc sống**

# CHƯƠNG 1: GIỚI THIỆU

1. **Giới thiệu về truy xuất thông tin:**

Truy xuất thông tin là tìm kiếm thông tin mong muốn trong các vật liệu chứa thông tin (tài liệu văn bản, …), các tài liệu phi cấu trúc hay bán cấu trúc (Free Text, XML), hoặc từ các tập lưu trữ lớn. Các dạng tài liệu đó có thể là email, tập tin trên máy tính cá nhân, hệ thống văn bản pháp lý, các cơ sở trí thức, hình ảnh và âm thanh, video,…

Một số công cụ, hệ thống truy xuất thông tin nổi tiếng: Google, Bing, Yahoo Search, …

Nhiệm vụ của truy xuất thông tin là: từ một bộ dữ liệu thông tin và một câu truy vấn của người sử dụng, ta xuất ra được kết quả là các thông tin liên quan nhất đến câu truy vấn đó.

1. **Các ứng dụng của truy vấn thông tin đa phương tiện**

Một số ứng dụng của truy xuất thông tin: truy xuất thông tin đa ngôn ngữ, truy xuất giọng nói, bản tin đài phát thanh, phân loại văn bản, tóm tắt văn bản, truy xuất thông tin có cấu trúc (XML), truy xuất thông tin địa lý, …

Một trong những ứng dụng nổi bật nhất của truy xuất thông tin là công cụ tìm kiếm web mà nổi tiếng nhất là Google với khả năng tìm kiếm văn bản, hinh ảnh, video, …

Bên cạnh đó còn có các ứng dụng khác như thư viện số, các hệ thống thông minh…

1. **Giới thiệu về đồ án:**

Xây dựng một hệ thống truy vấn trên văn bản, bao gồm các bước: đánh chỉ mục trên ổ cứng, cho người dùng nhập vào câu truy vấn, trả về các file liên quan đến truy vấn (có chứa những từ liên quan đến truy vấn).

Mục tiêu của đồ án: trả về các file văn bản đúng với câu truy vấn của người dùng nhất.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ

1. **Phân tích:**

Mô hình sẽ sử dụng: vector space model. Mô hình Vector Space Model sẽ biểu diễn các văn bản dưới dạng vector, các phần tử của vector sẽ thể hiện mức độ quan trọng của một từ và cả sự xuất hiện hay không xuất hiện của của nó trong một tài liệu.

Độ tương đồng sử dụng trong mô hình: độ tương đồng theo cosine

Term sẽ là một từ được tách bởi khoảng trắng, không phải là một từ không có ý nghĩa về mặt tìm kiếm (đại từ, giới từ, … hay còn được gọi là stopwords), và sẽ được đưa về dạng nguyên mẫu của từ (swimming 🡪 swim).

Việc chọn term như vậy sẽ dễ dàng cho việc tính toán và tìm kiếm, vector space model sẽ được tính toán dựa trên từng từ và văn bản, và mỗi từ tiếng anh đơn lẻ đều thể hiện được một ngữ nghĩa nhất định.

Một văn bản sẽ được thể hiện dưới dạng một vector chứa tần xuất xuất hiện của term đó trong văn bản (term frequency).

\*quá trình tính toán

Nhóm sử dụng inverted index thay cho index bình thường để xây dựng chỉ mục. Inverted index là sẽ map từ term đó sang các văn bản có chứa term đó. Việc xây dựng chỉ mục ngược sẽ tốn kém thời gian nhưng sẽ nhanh chóng hơn trong việc truy vấn vì chỉ cần map theo term sẽ suy ra được các thông tin chứa term đó. Việc nhanh chóng trong lúc truy vấn cũng sẽ phù hợp với nhu cầu của người dùng là đúng và chính xác

Xây dựng inverted index có dạng là một dictionary có chứa các thông tin như: term, tổng giá trị tf (tf\_overall), số lượng tài liệu chứa term đó (nums\_of\_docs), idf của term đó, văn bản chứa term đó, trọng số của term trong văn bản đã được chuẩn hóa …

Phân tích query tương tự như phân tích văn bản trước khi xây dựng inverted index, tách query thành các token dựa vào khoảng trắng, loại bỏ stopword, các ký tự đặc biệt, đưa về dạng nguyên mẫu của token đó, sau đó tiến hành tính w của query dựa trên tf của term đó trong query và idf của term đó trong inverted index

Tính độ tương đồng giữa query và tập văn bản dựa trên w của văn bản và w của query

Trả về kết quả theo độ tương đồng giảm dần

1. **Cấu trúc dữ liệu:**

Nhóm sử dụng một dictionary để lưu các giá trị khi lập chỉ mục bằng inverted index, dictionary sẽ có cấu trúc như sau:

{

<term>: {

“tf\_overall”: <tần số của term trên tất cả doc>

“num\_of\_docs”: <số lượng tài liệu chứa term>

“idf”: <idf của term>

“posting\_list”: {

<doc chứa term>: {

“tf”: <tf của term trong doc>

“w”: <trọng số của term trong doc đã được chuẩn hóa)

}

}

}

W của term sau khi được tính toán sẽ được lưu vào một dictionary có cấu trúc là:

{

<term>: “w” (trọng số w của term đó)

}

Độ tương đồng giữa văn bản và query được lưu trong một dictionary có cấu trúc:

{

“văn bản”: “độ tương đồng”

}

1. **Các phần mềm và thư viện cần thiết:**

Ngôn ngữ lập trình: Python 3.6.x

IDE: Sublime Text 3

Các thư viện:

Thư viện os để thực hiện các chức năng liên quan đến hệ điều hành

Thư viện re để thực hiện các chức năng liên quan đến biểu thức chính quy (Regex) như sub() và compile()

Thư viện collection để thực hiện các chức năng liên quan đến dictionary như xếp hạng dictionary

Hàm islice trong thư viện itertools để trả về các số lượng các phần tử cần thiết trong một collection

Thư viện nltk dùng cho lemmatization và stemmimg

Và các thư viện nhỏ khác để hỗ trợ như: math, …

# CHƯƠNG 3: THỰC HIỆN

1. **Tiền xử lý dữ liệu:**

Ý tưởng:

Theo cấu trúc dữ liệu đã đề ra, nhóm sẽ tạo một dictionary để chứa inverted index của các term. Vậy sẽ có một không gian tọa độ n chiều với mỗi chiều là một term trong inverted index.

Để xây dựng inverted index, cần phải có các bước xử lý trước:

Loại bỏ các kí tự đặc biệt như / \* - , …

Tách văn bản thành các token dựa vào các khoảng trắng

Loại bỏ các stopwords (các từ không có ngữ nghĩa về mặt tìm kiếm) tiếng Anh (như should, me, myself, …)

Lemmatization các token còn lại để đưa chúng về dạng nguyên mẫu từ, ví dụ: swimming 🡪 swim

Các token còn lại sẽ là term và được đưa vào để xây dựng inverted index

Thực hiện:

Bước 1: Tách văn bản thành các token dựa vào khoảng trắng:

Nhóm đọc nội dung file bằng lệnh read() và tách các token dựa vào khoảng trắng bằng lệnh split()

Bước 2: Loại bỏ stopwords và xây dựng inverted index:

Hàm thực hiện: build\_indexing(build\_inverted\_index(đường dẫn, các từ stopwords))

Đầu tiên nhóm tải file stopwords tiếng Anh “stopwords.txt” từ trên GitHub.

Sau đó nhóm đọc file stopword bằng hàm stopword(đường dẫn đến file stopwords) tự định nghĩa và dùng hàm split() để trả về một list các từ stopwords.

Nhóm tạo biểu thức chính quy (regex) để xử lý chung stopwords và các kí tự đặc biệt bằng re.compile().

Tiếp theo nhóm duyệt qua từng file trong dataset, đọc nội dung và chuyển về chữ thường và lưu vào string split\_words. Nhóm sử dụng hai lần hàm re.sub(): hàm re.sub() đầu tiên trong split\_words để thay thế các từ stopwords thành các ký tự khoảng trắng, hàm re.sub() thứ hai chuyển các kí tự đặc biệt bên trong split\_words thành khoảng trắng. Tiếp đến nhóm tách split\_words thành các token dựa vào khoảng trắng.

Cuối cùng các token sẽ được đưa vào hàm lemmatize() của thư viện nltk để tiến hành đưa các từ về nguyên mẫu của nó và các token còn lại sẽ là term và đưa vào xây dựng inverted index

1. **Xây dựng inverted index ban đầu:**

Hàm thực hiện: build\_inverted\_index(data\_path, stopwords)

Xây dựng inverted index ban đầu: nhóm tạo một dictionary có chứa key là các term, mỗi term như vậy sẽ có value là một dictionary nhỏ chứa file chứa term đó và số lần xuất hiện của term đó trong file đó. Cách xây dựng dictionary: nếu term đó chưa có trong dictionary, thì ta thêm term đó vào và tạo một dictionary mới làm value cho term đó. Nếu đã có term đó, ta xét đến dictionary nhỏ của term. Nếu chưa có file đang duyệt trong dictionary của term, ta thêm file đó vào thành key của dictionary nhỏ và gán số lần xuất hiện là 1, ngược lại ta tăng số lần xuất hiện lên thêm 1.

Cấu trúc của dictionary inverted\_index theo code: {term: {tên file chứa term đó: số lần term đó xuất hiện trong file đó}}

Hàm này sẽ trả về collection chứa inverted index ban đầu

1. **Tính toán các thông số còn lại:**

Với mỗi term trong tập từ vựng ta sẽ thực hiện: tính trọng số tf, idf, normalize, weight (w) của term đó

Tính toán tf – idf:

Trọng số tf: Một term sẽ quan trọng hơn trong một văn bản nếu term đó xuất hiện nhiều lần văn bản đó

Tf(t,d) = (với t là term, d là văn bản)

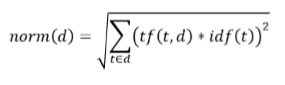
Trọng số tf\_overall = tổng trọng số tf của các doc

Trọng số num\_of\_docs = số lượng văn bản chứa term đó

Trọng số idf: Một term sẽ dễ dàng phân biệt hơn nếu nó xuất hiện trong ít văn bản hơn

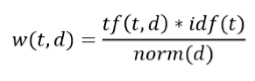
Idf(t) = (với t là term)

Trọng số normalize:



(với t là term, d là văn bản)

Trọng số weight:



(với t là term, d là văn bản)

Thực hiện:

Thực hiện tính toán các trọng số theo công thức ở trên.

Sau đó nhóm thực hiện lưu kết quả tính toán vào dictionary inverted index ban đầu theo cấu trúc dictionary đã định sẵn. Idf, num\_of\_doc sẽ nằm trong <term> và tf, w sẽ nằm trong <doc chứa term>

1. **Xử lý query:**

Hàm thực hiện: queryprocess(query, non\_words,indexing), trong đó query là query được truyền vào, non\_words là list danh sách các stopwords, indexing là inverted index đã được xây dựng

Query cũng sẽ được tiền xử lý tương tự như một văn bản trong tập dữ liệu

Đầu tiên nhóm sẽ tiến hành tách query thành từng token dựa vào khoảng trắng

Sau đó nhóm loại bỏ các token là các từ stopwords, loại bỏ các ký tự đặc biệt

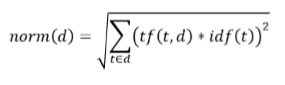
Quá trình tiền xử lý kết thúc bằng việc lemmatize các token để đưa chúng về dạng nguyên mẫu của từ, cũng như để đồng nhất chúng với các term trong inverted index. Những token còn lại sau bước này sẽ là term của query.

Sau đó ta tiến hành tính các trọng số của từng term của query tương tự như term trong inverted index: tf, normalize và weight của term

Trọng số tf được tính theo công thức:

Tf(t,d) =

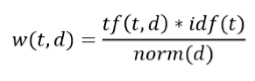
Trọng số normalize được tính theo công thức:



(với t là term, d là query)

Trọng số idf trong công thức normalize là trọng số idf của term đó trong inverted index

Trọng số weight của term được tính như sau:



(với t là term, d là query)

1. **Tính độ tương đồng**

Ý nghĩa:

Dựa vào độ tương đồng, ta có thể xác định được các văn bản liên quan đến query. Ta có thể sắp xếp độ liên quan của các file để có thể trả về các kết quả liên quan nhất cho query.

Ý tưởng:

Độ tương đồng của văn bản đó và query sẽ được tính dựa trên tích giữa weight của term đó trong inverted index và weight của term đó trong query

Thực hiện:

Hàm thực hiện: queryprocess(query, non\_words,indexing), trong đó query là query được truyền vào, non\_words là list danh sách các stopwords, indexing là inverted index đã được xây dựng

Độ tương đồng sẽ được tính theo công thức được đưa ra ở trên

Sau khi có kết quả độ tương đồng, nhóm lưu vào một dictionary có key là văn bản và value là độ tương đồng của văn bản đó với query: { <doc>: <similarity> }. Sau đó nhóm thực hiện sắp xếp các văn bản theo độ tương đồng giảm dần từ cao xuống thấp để trả về các kết quả có độ tương đồng cao nhất.

1. **Đánh giá hệ thống truy vấn:**

Bộ dữ liệu thử nghiệm: Bộ dữ liệu Cranfield do giáo viên cung cấp

Các hệ số sử dụng để đánh giá hệ thống truy vấn:

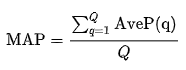
Precision: khả năng trả về những văn bản có liên quan nhất

Precision =

Recall: khả năng tìm kiếm tất cả các văn bản có liên quan trong tập dữ liệu

Recall =

mAP:



Precision, recall và mAP sẽ được tính toán ở vị trí văn bản trả về là văn bản có liên quan đến query dựa trên kết quả của giáo viên.

Số kết quả trả về của nhóm sẽ bằng số kết quả trả về trong file kết quả của giáo viên.

# CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

1. **Hướng dẫn sử dụng:**

Clone project tại: <https://github.com/QuocKiet/Information-retrieval>

Cài đặt python: https://www.python.org/downloads/windows/

Cài các thư viện cần thiết bằng cmd: pip install -r requirements.txt

Mở file search.py bằng sublime hoặc visual studio code, sau đó tiến hành build bằng ctrl + B cho sublime hoặc sử dụng cmd với lệnh python search.py cho visual studio code hoặc command prompt tại ngay thư mục chứa file search.py

Để chuyển đổi câu query cần thực hiện, thay thế câu query đó vào tham số đầu tiên của hàm queryprocess() tại hàm main() của chương trình:



1. **Kết quả chạy chương trình:**

Dictionary chứa inverted index: indexing

Dictionary chứa precision: precision

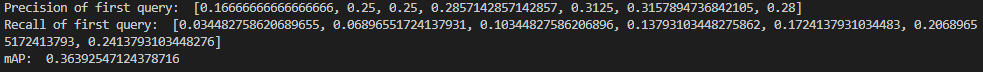
Dictionary chứa recall: recall

Thư mục chứa kết quả các truy vấn: “result/”

Số lương file liên quan trả về rất nhiều, nên nhóm chỉ lấy số lượng file liên quan bằng với số lượng file trong file kết quả của thầy (ví dụ file kết quả 1.txt của thầy có 25 kết quả thì file 1.txt trong thư mục result của nhóm sẽ có 25 kết quả)

Nhóm đã áp dụng stemming và lemmatizing trong xen kẽ nhau trong quá trình thực hiện và nhận thấy lemmatizing cho kết quả tốt hơn

Kết quả precision và recall của query đầu tiên và giá trị mAP của nhóm:



Kết quả 3 văn bản có độ tương đồng cao nhất của query thứ nhất:



# CHƯƠNG 5: KẾT LUẬN

1. **Kết quả đạt được:**

Chương trình chạy được và đúng

1. **Hạn chế:**

Kết quả độ đo mAP chưa được cao

Nhiều văn bản còn chưa trả về kết quả nào liên quan

1. **Hướng phát triển:**

Nghiên cứu thêm, tối ưu hóa code để cho kết quả nhanh hơn

Sử dụng một cách stemming và lemmatizing tốt hơn, có áp dụng xử lý ngôn ngữ tự nhiên tốt hơn để quá trình xây dựng inverted index và xử lý query trở nên tốt hơn

1. **Phân công công việc:**

Về việc lên ý tưởng: cả nhóm cùng họp nhóm và đóng góp ý tưởng, sau đó nhóm trưởng sẽ thống nhất và chia công việc cho các thành viên trong nhóm

Nhóm sẽ có các buổi họp nhóm, các thành viên sẽ đều tham gia. Thành viên nào được giao nhiệm vụ sẽ code chính nhiệm vụ đó, còn các thành viên còn lại sẽ đóng góp ý kiến và hỗ trợ giải quyết vấn đề

Mai Quốc Kiệt: xây dựng inverted index đầy đủ, xử lý query

Lê Thiện Duy: đánh giá hệ thống truy vấn, tìm hiểu lemmatizing, làm báo cáo, xử lý những vấn đề còn sót lại

1. **Biên bản làm việc nhóm:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chuẩn đánh giá | Tỉ trọng | Mai Quốc Kiệt | Lê Thiện Duy |
| Đóng góp về nội dung, chất lượng làm bài | **50%** | **100%** | **100%** |
| Giải quyết khó khăn khi làm bài, tư duy sáng tạo | **15%** | **100%** | **100%** |
| Hợp tác, nhiệt tình, tinh thần trách nhiệm cao | **15%** | **100%** | **100%** |
| Sự chuyên cần | **10%** | **100%** | **100%** |
| Tư duy phản biện, biết tôn trọng, lắng nghe | **10%** | **100%** | **100%** |

1. **Tài liệu tham khảo:**

Các slide bài giảng của giảng viên

File stopwords tiếng Anh: <https://gist.github.com/sebleier/554280>

Tìm hướng giải quyết các vấn đề chưa xử lý được: <https://stackoverflow.com/>

Tài liệu hướng dẫn đọc và ghi file JSON:

http://docs.python-guide.org/en/latest/scenarios/json/

Tài liệu hướng dẫn sử dụng regex: <https://regexr.com/>

Và một số trang web khác

## Link Github chứa đồ án của nhóm:

<https://github.com/QuocKiet/Information-retrieval>