MỞ ĐẦU

Ngày nay, sự truy tìm thông tin có vai trò rất quan trọng trong mọi lĩnh vực hoạt động của chúng ta – đặc biệt với sự xuất hiện của mạng toàn cầu thì khối lượng thông tin trên các máy tính đã tăng theo hàm mủ; việc tìm kiếm những thông tin hữu ích ngày càng tăng và trở nên thiết yếu, kéo theo những bài toán cần giải quyết để phục vụ cho vấn đề nêu trên - là xây dựng các hệ thống phục vụ cho việc tìm kiếm và tra cú thông tin một cách chính xác và nhanh nhất các thông tin mà họ cần trên kho tư liệu khổng lồ này.

Các kỹ thuật truy vấn thông tin hiện nay thường dùng [6]:

- Dựa trên các mô hình: mô hình boolean, mô hình xác suất và mô hình không gian vector.
- Dùng các kỹ thuật gom cụm dữ liệu.

Luận văn sẽ trình bày sự cần thiết của mô hình không gian vector và trọng số của từ chỉ mục – các văn bản, câu truy vấn và từ chỉ mục được biểu diễn thành các vector trong không gian vector. Hiện nay, mô hình không gian vector và mô hình Latin Semantec Index (LSI) đang được nghiên cứu cho việc xây dựng các hệ truy tìm thông tin (Information Retrievel System) – gọi tắt là IR, đạt hiệu quả hơn rất nhiều so với hệ thống sử dụng mô hình Boolean [3].

Với mô hình không gian vector, các văn bản, câu truy vấn và từ chỉ mục được biểu diễn thành các vector trong không gian vector. Mỗi tập văn bản được đại diện bởi một tập các từ chỉ mục và được gọi là không gian văn bản. Trong không gian vector văn bản, mỗi thành phần của vector văn bản biểu diễn độ đo trọng số của tập từ chỉ mục tương ứng với văn bản đó. Sử dụng các phép toán trên không gian vector để tính toán độ đo tương tự giữa câu truy vấn và các văn bản hoặc các từ chỉ mục, kết quả sau khi tính toán có thể được xếp hạng theo độ đo tương tự với vector truy vấn. Ngoài ra, mô hình không gian vector còn hướng dẫn người dùng biết được

những văn bản độ tương tự cao hơn có nội dung gần với nội dung họ cần hơn so với các văn bản khác[2], [4].

Mô hình LSI sử dụng phép chiếu trực giao ma trận biểu diễn tập văn bản có hạng r vào không gian k chiều (k<<r).Hiệu quả truy tìm sử dụng mô hình LSI được đánh giá trong các bài báo [2], [3], [7] cao hơn so với mô hình không gian vector chuẩn. Mục tiêu của việc dùng mô hình LSI là để khắc phục những hạn chế của mô hình không gian vector và làm sao cho hệ thống hoạt động tối ưu hơn.Tuy nhiên việc chọn hệ số k trong mô hình LSI cho tới hiện nay vẫn còn là một bài toán chưa có lời giải tổng quát. Cho tới hiện tại việc chọn *k* cho mô hình LSI chỉ thực hiện dựa trên các phương pháp thử nghiệm, cụ thể một phương pháp mới nhất được đề nghị trong bài báo [8].

Mục tiêu của luận văn này sẽ đề xướng một phương pháp gom nhóm các tài liệu văn bản trước khi truy vấn thông tin. Cụ thể là: sử dụng thuật toán gom cụm K-means để gom nhóm các tài liệu văn bản HTML tiếng Anh. Thuật toán K-means và các biến thể của nó đều nhằm mục đích tăng độ hội tụ và cách tính các khoảng cách từ đối tượng đến các trọng tâm của cụm. Trong luận văn này cũng trình bày hai cách cải tiến cho thuật toán gom cụm K-means như sau:

- Tiền xử lý tập dữ liệu vào dùng mô hình LSI: đối với hệ truy tìm thông tin thì tập văn bản rất lớn, việc xử lý tập dữ liệu vào được coi là hết sức quan trọng vì nó liên quan đến hiệu quả của việc truy tìm thông tin như: thời gian truy tìm, các văn bản liên quan đến truy vấn (mô hình LSI sẽ được trình bày chi tiết trong chương 2).
- Đề nghị một độ đo khoảng cách thích hợp cho hệ truy tìm văn bản (sẽ được trình bày chi tiết trong chương 3).

Tiếp theo luận văn sẽ đem kết quả đạt được sau khi cải tiến so sánh với hệ truy tìm thông tin dùng mô hình không gian vector và mô hình cải tiến LSI trong các bài báo [4], [6], [8].

Bố cục của luận văn bao gồm các chương sau:

Chương 1: Tổng quan về hệ truy tìm thông tin.

Chương 2: Mô hình không gian vector (VSM) và Mô hình Latin Semantec Index (LSI).

Chương 3: Kết hợp thuật toán gom cụm K-means và mô hình LSI vào bài toán gom cụm văn bản.

Chương 4: Cài đặt thử nghiệm hệ truy tìm thông tin (IR).

Kết luận và hướng phát triển.

Phần tài liệu tham khảo và phụ lục.

CHƯƠNG 1
TỔNG QUAN VỀ HỆ TRUY TÌM THÔNG TIN

Trong chương này trình bày các nội dung sau:

- Khái quát về các mô hình hệ truy tìm thông tin.
- Gom cụm văn bản
- Một số công trình nghiên cứu trong và ngoài nước.
- Kết luận.

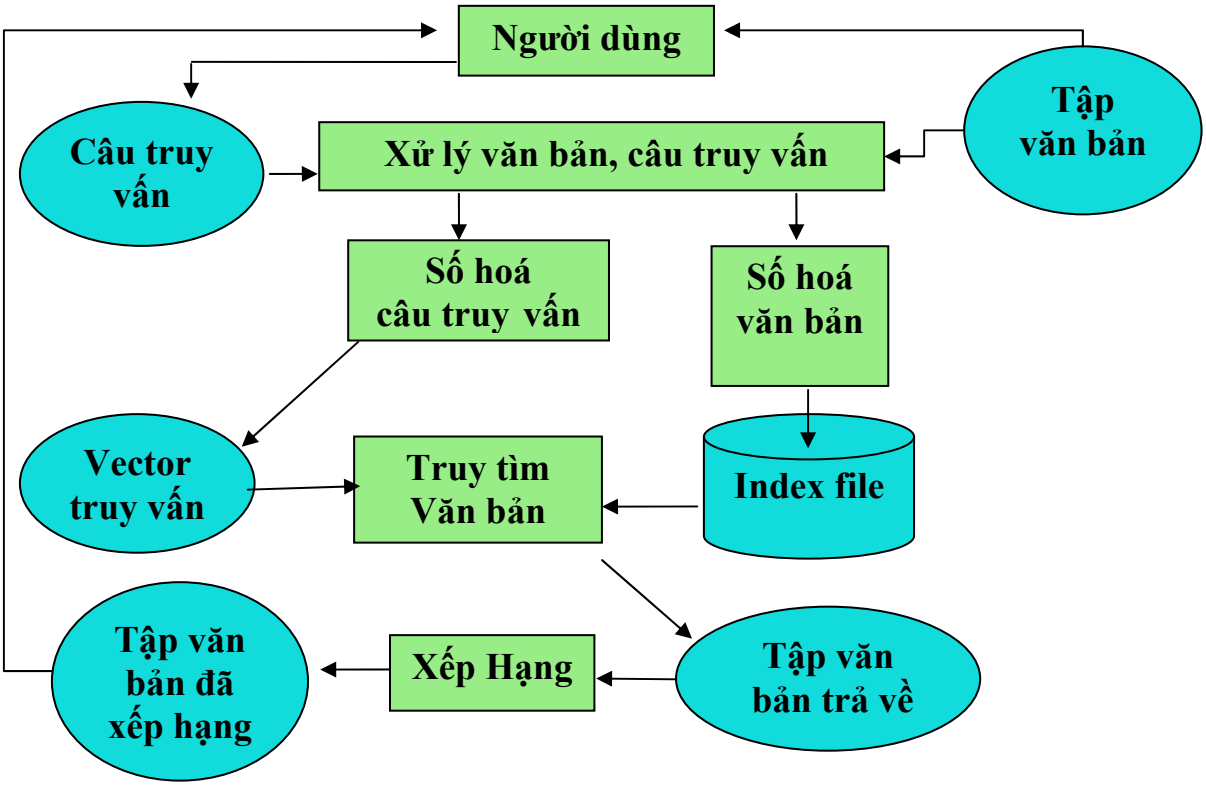
1.1 Hệ truy tìm thông tin (information retrieval system)

1.1.1 Giới thiệu:

Hệ truy tìm thông tin (IR) là một hệ thống được nảy sinh để giải quyết cho việc truy tìm những thông tin liên quan đến nhu cầu trong mọi lĩnh vực của người dùng. Quy trình của hệ tìm kiếm thông tin như sau:

- Người dùng muốn tìm một tài liệu liên quan đến một chủ đề nào đó.
- Người dùng cung cấp một mô tả chủ đề đó dưới dạng câu truy vấn.
- Từ câu truy vấn này, hệ thống sẽ lọc ra những cụm từ chỉ mục.
- Những cụm từ chỉ mục này sẽ được so khớp với những từ chỉ mục của văn bản đã được xử lý.
- Hệ thống sẽ trả về những văn bản có độ liên quan cao nhất.

Sau đây là kiến trúc của hệ truy tìm thông tin



Hình 1.1 Kiến trúc của hệ IR

Theo truyền thống, việc tìm kiếm thông tin được thực hiện bằng tay, phần lớn thường gặp trong các mẫu liệt kê những quyển sách trong thư viện hay trong chính bảng mục lục của quyển sách…Những mẫu liệt kê hay bảng mục lục này thường có chứa một số lượng nhỏ các từ chỉ mục như là: tiêu đề, tác giả và một số tiêu đề chính.

Những vấn đề trên trải qua suốt hàng thập kỷ, mãi đến thế kỷ 20 khi có sự xuất hiện của máy tính thì việc tìm kiếm thông tin đã thay đổi hoàn toàn – tạo ra một cuộc cách mạng lớn trong việc truy tìm thông tin.

Ngày nay, hệ truy tìm thông tin đóng một vai trò rất lớn trong các lĩnh vực của chúng ta - Đặc biệt với sự xuất hiện của hệ thống Internet và mạng toàn cầu. Trong 10 năm gần đây, số lượng thông tin ở các dạng mẫu khác nhau trên các trang điện tử đã tăng vọt theo hàm mủ. Thông tin có thể là văn bản, ảnh số, video, thư viện phần

mềm, bách khoa toàn thư trực tuyến, thông tin thương mại, v.v… từ các kho dữ liệu. Trong bài luận này chỉ tập trung vào trình bày thông tin văn bản.

Hệ truy tìm thông tin xuất hiện tại thời điểm nóng bỏng này là một cuộc cách mạng và là một điều kiện cần thiết cho việc ứng dụng khoa học máy tính vào tất cả các lĩnh vực trên toàn cầu, điển hình như các hệ truy tìm được người dùng quan tâm nhiều nhất hiện nay là google, yahoo, v.v…

Thành phần chính của mô hình trên là việc số hóa văn bản, thành phần này có nhiệm vụ chuyển tập văn bản ở ngôn ngữ tự nhiên thành các tập tin chỉ mục có cấu trúc bằng cách sử dụng mô hình không gian vector.

1.1.2 Mục tiêu của hệ truy tìm thông tin

Mục tiêu chính của hệ truy tìm thông tin (IR) là truy tìm những văn bản trong tập văn bản của hệ thống liên quan đến thông tin mà người sử dụng hệ thống cần. Những thông tin được người dùng đưa vào hệ thống bởi các câu truy vấn (*query*). Những tài liệu – văn bản "liên quan" (*relevant*) với câu truy vấn sẽ được hệ thống trả về. Như vậy, mục đích của hệ IR là để tự động quy trình kiểm tra tài liệu bằng cách tính độ đo tương quan giữa câu truy vấn và tài liệu.

1.2 Các mô hình của hệ truy tìm thông tin

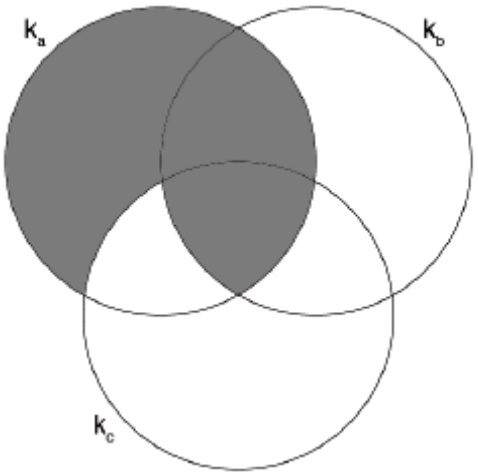
1.2.1 Mô hình Boolean

Mô hình Boolean là mô hình cổ điển và đơn giản đã được sử dụng trước đây và cho đến nay vẫn còn được sử dụng trong các hệ thống IR. Mô hình Boolean dựa trên lý thuyết tập hợp (*set theory*) và đại số Boolean (*Boolean algebra*). Mô hình Boolean phổ biến bởi vì cả lý thuyết tập hợp và đại số Boolean có mối quan hệ đơn giản và dễ hiểu, vì vậy các hệ IR được xây dựng trên mô hình nầy, người dùng dễ dàng sử dụng.

Với mô hình Boolean văn bản được biểu diễn bởi một vector nhị phân, tức là các vector có các phần tử thuộc {0, 1}. Từ chỉ mục thứ k_i xuất hiện trong văn bản d_j thì trọng số $w_{ij} = 1$, ngược lại $w_{ij} = 0$.

Tất cả các truy vấn được biểu diễn bởi các biểu thức Boolean, sử dụng ba phép toán cơ bản: *not, and, or*, được biểu diễn trong hình 1.2.

Văn bản truy vấn sử dụng mô hình này được xem như: hoặc liên quan đến nội dung truy vấn hoặc không, ở đây không có cách để để tìm các văn bản chỉ liên quan cục bộ hay còn gọi là liên quan một phần (*partially relevant*) của
câu truy vấn. Ví dụ cho văn bản *d*, *d* có từ chỉ mục k_b, tuy nhiên *d* được xem như không liên quan tới câu truy vấn $q = k_a$ *AND* $(k_b$ *or* $k_c)$. Bởi vì *d* không có
từ chỉ mục k_A nên không liên quan (*irrelevant*) đến câu truy vấn.



Hình 1.2 trình bày kết quả truy vấn $q = k_a$ *AND* $(k_b$ *or* $k_c)$.

Ưu điểm của mô hình Boolean:

- Đơn giản và dễ sử dụng.

Nhược điểm của mô hình Boolean:

- Vì dựa trên phép toán logic nhị phân nên một văn bản được tìm kiếm chỉ xác định hai trạng thái: liên quan hoặc không với câu truy vấn.

- Việc chuyển một câu truy vấn của người dùng sang dạng biểu thức Boolean không đơn giản.

1.2.2 Mô hình không gian vector

Mô hình không gian vector khắc phục những nhược điểm của mô hình boolean là việc sử dụng trọng số cho từ chỉ mục khác trọng số nhị phân (*non-binary*). Trọng số từ chỉ mục không giới hạn bởi hai trị 0 hoặc 1, các trọng số này được sử dụng để tính toán độ đo tương tự của mỗi văn bản với câu truy vấn. Với mô hình không gian vector, các văn bản, câu truy vấn và từ chỉ mục được biểu diễn thành các vector trong không gian vector. Sử dụng các phép toán trên không gian vector để tính toán độ đo tương tự giữa câu truy vấn và các văn bản hoặc các từ chỉ mục, kết quả sau khi tính toán có thể được xếp hạng theo độ đo tương tự với vector truy vấn. Ngoài ra, mô hình không gian vector còn hướng dẫn người dùng biết được những văn bản độ tương tự cao hơn có nội dung gần với nội dung họ cần hơn so với các văn bản khác.

Mô hình không gian vector dựa trên giả thiết là nội dung của văn bản có thể được hiểu như sự kết hợp của các từ chỉ mục. Một văn bản *d* được biểu diễn như một vector của các từ chỉ mục $\mathrm{d} = (\mathrm{t}_1, \mathrm{t}_2 \ldots, \mathrm{t}_\mathrm{n})$ với t_i là từ chỉ mục thứ i ($1 \le i \le n$) (*các giá trị có thể là số lần xuất hiện của term t_i trong văn bản d*). Mỗi từ chỉ mục trong văn bản biểu diễn một chiều (*dimension*) trong không gian. Tương tự, câu truy vấn cũng được biểu diễn như một vector $\mathrm{q} = \left(\hat{\mathrm{t}}_1, \hat{\mathrm{t}}_2, \ldots, \hat{\mathrm{t}}_\mathrm{n} \right)$.

Sau khi đã biểu diễn tập văn bản và câu truy vấn thành các vector trong không gian vector, ta có thể sử dụng độ đo *cosines* để tính độ đo tương tự giữa các vector văn bản và vector truy vấn.

Ưu điểm của mô hình không gian vector:

- Đơn giản, dễ hiểu

- Cài đặt đơn giản
- Khắc phục các hạn chế trên mô hình Boolean

Nhược điểm mô hình không gian vector:

- Số chiều biểu diễn cho tập văn bản có thể rất lớn nên tốn nhiều không gian lưu trữ.

1.2.3 Mô hình xác suất

Cho câu truy vấn của người dùng *q* và văn bản *d* trong tập văn bản. Mô hình xác suất tính xác suất mà văn bản *d* liên quan đến cấu truy vấn của người dùng. Mô hình giả thiết xác suất liên quan của một văn bản với câu truy vấn phụ thuộc cách biểu diễn chúng. Tập văn bản kết quả được xem là liên quan và có tổng xác suất liên quan với câu truy vấn lớn nhất.

Ưu điểm của mô hình xác suất:

- Văn bản được sắp xếp dựa vào xác suất liên quan đến câu truy vấn

Nhược điểm mô hình xác suất:

- Mô hình không quan tâm đến số lần xuất hiện của từ chỉ mục trong văn bản
- Việc tính toán xác suất khá phức tạp và tốn nhiều chi phí.

Bảng PLA.1 trong phụ lục A trình bày chi tiết ưu nhược điểm của mô hình Boolean, Không gian vector và mô hình xác suất.

1.3 Gom cụm văn bản

Ngoài việc sử dụng các mô hình trên thì kỷ thuật gom cụm văn bản cũng được ứng dụng rất nhiều trong hệ truy tìm thông tin. Việc ứng dụng gom cụm không chỉ dùng cho văn bản mà còn cho các bài toán khác như: gom cụm hình ảnh, đồ thị, video…

Mục tiêu của việc gom cụm là để gom tập các đối tượng thành các nhóm, dựa trên cách thức phân loại dựa trên các vector đặc trưng. Các đối tượng dữ liệu

cùng loại thì được gom về cùng cụm – các đối tượng dữ liệu tương tự với một đối tượng khác trong cùng cụm và không tương tự với các đối tượng khác trong cụm khác - Gom cụm phụ thuộc vào việc định nghĩa các độ đo khoảng cách.

1.3.1 Phương pháp dựa trên phân hoạch

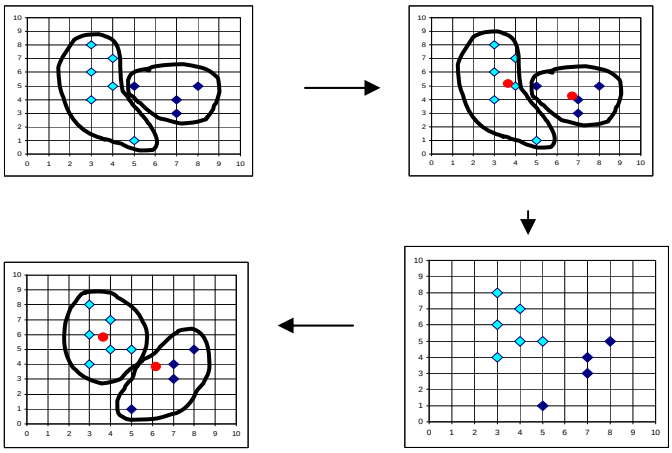
Tạo một phân hoạch của CSDL *D* chứa *n* đối tượng thành tập gồm *k* cụm sao cho:

- Mỗi cụm chứa ít nhất là một đối tượng
- Mỗi đối tượng thuộc về đúng một cụm

Có 2 phương pháp:

- K-means: mỗi cụm được đại diện bằng tâm của cụm (**centroid**)
- K-mediods: mỗi cụm được đại diện bằng một trong các đối tượng của cụm (**medoid**)

Cả hai phương pháp trên đều phải cho biết trước số cụm k.



Hình 1.3 phương pháp gom cụm k-means

Ưu điểm:

- Scalable tương đối: trong khi xử lý các tập dữ liệu lớn
- Hiệu suất tương đối: $O(tkn)$, với n là số đối tượng, k là số cụm, và t là số lần lặp. Thông thường $k, t << n$.
- Thường kết thúc ở điểm tối ưu cục bộ; có thể tìm được tối ưu toàn cục

dùng các kỹ thuật như thuật toán di truyền

Nhược điểm:

- Có thể áp dụng chỉ khi xác định được trị trung bình của các đối tượng
- Cần chỉ định trước *k,* số các cụm
- Không thể xử lý dữ liệu chuỗi và outliers
- Không phù hợp để khám phá các cụm với dạng không lồi hay cụm có kích thước khác nhau.

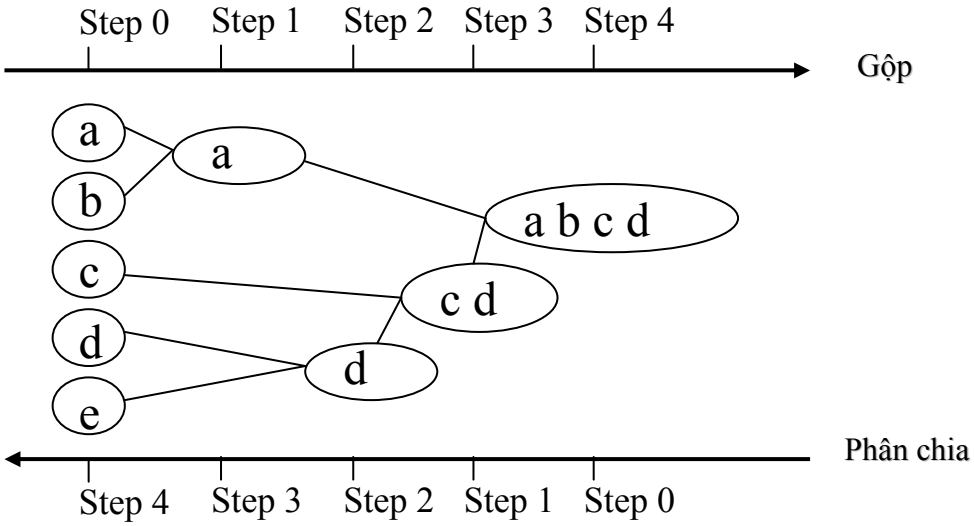
1.3.2 Phương pháp phân cấp

Tạo phân cấp cụm, chứ không phải là một phân hoạch đơn thuần các đối tượng, phương pháp này không cần phải cho biết trước số cụm k. Phân cấp cụm thường tạo cây các cụm hay còn được gọi là **dendrogram.** Trong đó:

- Các lá của cây biểu diễn các đối tượng riêng lẻ.
- Các nút trong của cây biểu diễn các cụm.

Có 2 loại gom cụm phân lớp:

- **Gộp-agglomerative** (từ dưới lên):
  - Đưa từng đối tượng vào cluster riêng của nó (a singleton)
  - Trộn ở mỗi bước hai cụm tương tự nhất cho đến khi chỉ còn một cụm hay thỏa điều kiện kết thúc
- **Phân chia -divisive** (từ trên xuống):
  - Bắt đầu bằng một cụm lớn chứa tất cả đối tượng.
  - Phân chia cụm phân biệt nhất thành các cụm nhỏ hơn và xử lý cho đến khi co *n* cụm hay thỏa điều kiện kết thúc



Hình 1.4 phương pháp gom cụm phân cấp

Ưu điểm:

- Khái niệm đơn giản.
- Lý thuyết tốt.
- Khi cụm được trộn/tách, quyết định là vĩnh cửu => số các phương án khác nhau cần được xem xét bị rút giảm.

Nhược điểm:

- Trộn/tách các cụm là vĩnh cửu => các quyết định sai là không thể khắc phục về sau.
- Các phương pháp phân chia là cần thời gian tính toán.
- Các phương pháp là không scalable cho các tập dữ liệu lớn.

1.3.3 Phương pháp dựa trên mật độ.

Bắt đầu bằng việc tìm kiếm các đối tượng lõi (**core**), dựa vào những lõi này để hình thành các cụm. Một số nghiên cứu liên quan:

- DBSCAN: được Ester giới thiệu vào năm 1996, khi nghiên cứu các thuật toán phân cụm dữ liệu không gian. DBSCAN được khẳng định qua thực nghiệm là tốt hơn các thuật toán khác. Cụ thể so với thuật toán CLARANS thì

DBSCAN phát hiện ra các cụm bất kì nhiều hơn và thực hiện tốt trên 100 tiêu chuẩn đánh giá hiệu quả thuật toán [Ester 1996].

Ưu điểm:

- Phát hiện ra các cụm với hình dạng bất kì, kể cả hình không lồi.
- Khử nhiễu tốt.

Ưu điểm

- Nếu các cụm có mật độ khác nhau nhiều thì DBSCAN sẽ không giữ được tính hiệu quả. Trên những dữ liệu như thế ta phải áp dụng mật độ của cụm có mật độ thấp nhất cho tất cả các cụm khác. Với các cụm có mật độ rất cao thì DBSCAN tốn nhiều thời gian để xác định lân cận của các điểm một cách không cần thiết.
- Nếu có quan tâm đến các thuộc tính phi không gian (non-spatial) thì sử dụng DBSCAN không thích hợp vì DBSCAN không chú ý đến các thuộc tính đó.

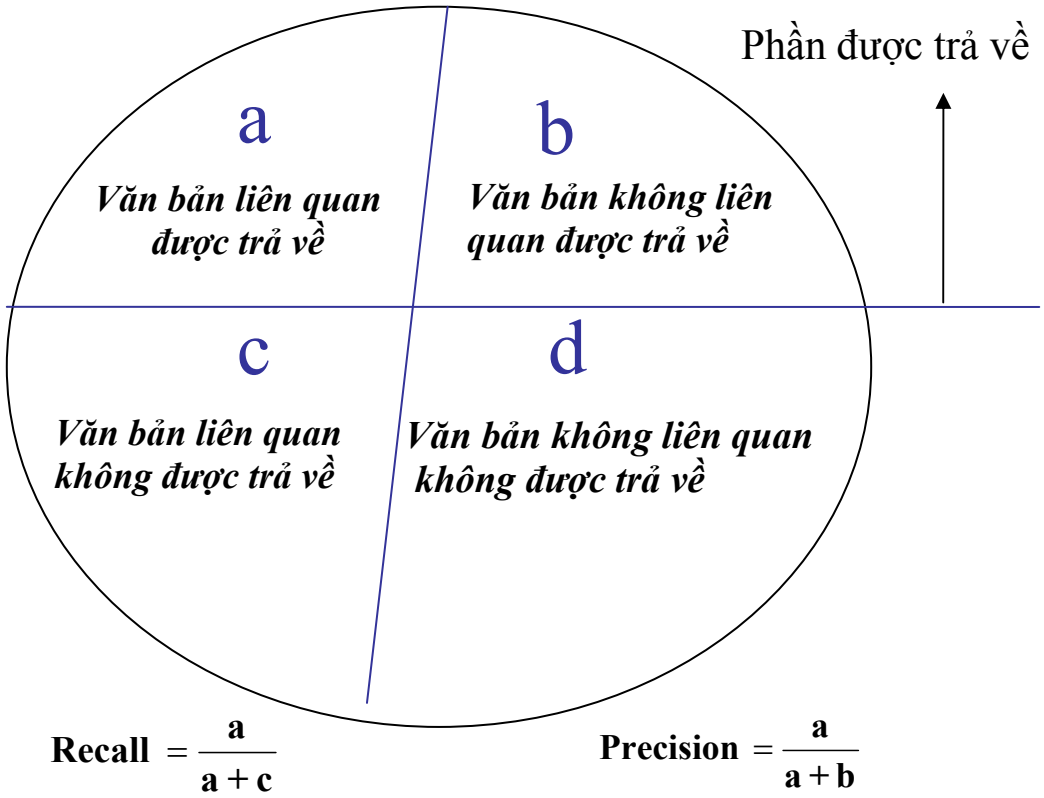
Bảng PLA.2 trong phụ lục A trình bày chi tiết ưu nhược điểm của các thuật toán gom cụm.

1.4 Đánh giá hiệu quả hệ truy tìm thông tin

Để đánh giá hiệu quả của hệ truy tìm thông tin có thể dựa theo các tiêu chuẩn sau [6]:

- Dựa trên hai độ đo: "độ chính xác" (precision) và "độ bao phủ" (recall).

Độ chính xác là tỉ lệ các văn bản liên quan được trả về trên tổng số các văn bản trả về tương ứng với câu truy vấn, và độ bao phủ là tỉ số của số văn bản liên quan được trả về trên tổng số các văn bản liên quan đến câu truy vấn trong tập văn bản. Như vậy, p*recision* đo hiệu quả của hệ thống theo quan điểm người dùng, và *recall* khả năng truy tìm những văn bản liên quan đến câu truy vấn của hệ thống. Thông thường khi độ đo *precision* tăng thì *recall* giảm và ngược lại. Miền giá trị của *precision* và *recall* nằm trong khoảng [0,1].



Hình 1.5 tính độ hiệu quả của hệ truy tìm thông tin

- Hiệu quả thực thi của hệ thống(*Execution efficiency*) được đo bởi thời gian thực hiện thủ tục tìm kiếm các văn bản liên quan đến câu truy vấn được cho.
- Hiệu quả lưu trữ được đo bởi dung lượng bộ nhớ cần thiết để lưu trữ dữ liệu (*cả bộ nhớ ngoài lưu trữ dữ liệu chỉ mục và bộ nhớ RAM khi hệ thống thực thi*).

1.5 Một số công trình nghiên cứu trong và ngoài nước:

1.5.1 Ở Việt Nam:

Hiện nay, ở nước ta có công trình nghiên cứu về mô hình Latin semantec Index như sau:

Đỗ Trung Hiếu (2005), *Số hóa văn bản theo mô hình không gian vector và ứng dụng, luận* văn thạc sĩ, Trường Đại Học Khoa Học Tự Nhiên.

1.5.2 Ở nước ngoài:

Ở nước ngoài, có công trình nghiên cứu về mô hình Latin semantec Index như sau:

Kevin Erich Heinrich (2007), *Automated Gene Classification using Nonnegative Matrix Factorization on Biomedical Literature*, Doctor of Philosophy Degree, The University of Tennessee, Knoxville.

Dawid Weiss (2006), *Descriptive Clustering as a Method for Exploring Text Collections*, Pozna´n University of Technology Institute of Computing Science.

1.6 Kết luận và phạm vi luận văn

Do tính hiệu quả thấp của mô hình Boolean (*Boolean Model*), mô hình xác suất (*Probabilistic Model*), nên hiện nay mô hình không gian vector và mô hình LSI đang được nghiên cứu phục vụ cho việc xây dựng các hệ thống IR hiện đại hoạt động hiệu quả hơn thay thế các hệ thống cũ [4].

Tuy nhiên, trong mô hình không gian vector việc sử dụng ma trận hóa vector văn bản làm cho số chiều của ma trận rất lớn, ảnh hưởng đến hiệu quả của việc truy tìm thông tin. Hơn nữa, việc tìm kiếm các văn bản liên quan đến câu truy vấn có độ tin cậy thấp – nghĩa là có những văn bản liên quan mà không được trả về cho người dùng.

Do đó, mô hình LSI được đưa ra để khắc phục những hạn chế của mô hình không gian vector. HIệu quả của mô hình LSI được đánh giá là cao hơn so với mô hình không gian vector [2], [4], [7].

Phạm vi luận văn

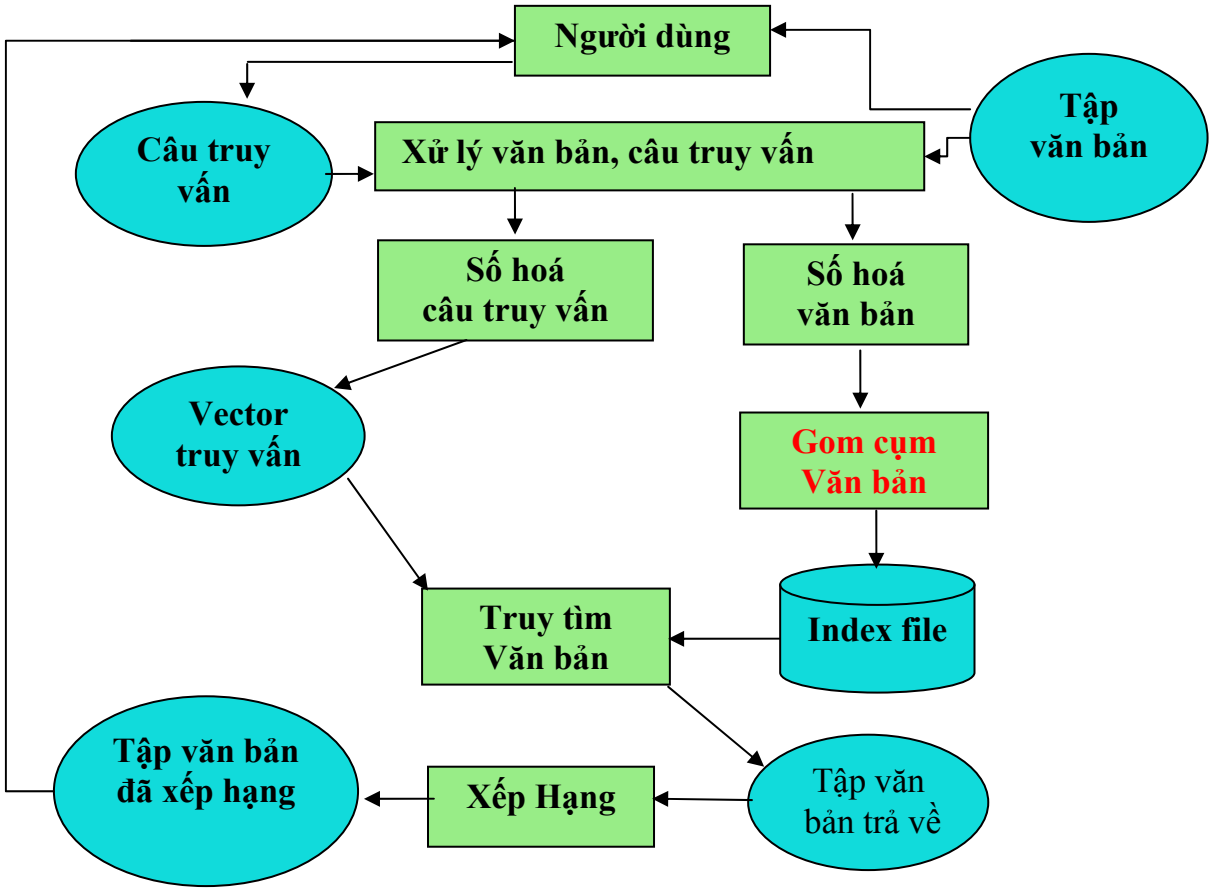
Trong mô hình LSI, việc phân tích SVD cho ma trận từ của văn bản (term document A) trong mô hình không gian vector làm giảm đi số chiều của ma trận A

rất nhiều và việc giải quyết được các văn bản liên quan đến câu truy vấn mà được xem là điểm yếu trong mô hình không gian vector, nên mô hình LSI được đánh giá rất cao. Tuy vậy, để trả về các văn bản liên quan thì ta cũng phải đi so sánh với tất cả các văn bản trong tập dữ liệu. Điều này dẫn đến việc hạn chế tốc độ tìm kiếm của giải thuật.

Để khắc phục điều này, Trong luận văn này đề nghị một phương pháp, là trước khi thực hiện tính Cosines giữa vector truy vấn với các vector văn bản trong ma trận A_k ta tiến hành gom cụm văn bản trước trong ma trận A_k. Bài toán gom cụm ở đây được chọn là thuật toán K-means được cải tiến qua 2 bước:

- Tiền xử lý tập dữ liệu vào dùng mô hình LSI.
- Chọn một độ đo thích hợp để tính độ tương tự cho các văn bản.

Sau khi tiến hành gom cụm văn bản trên ma trận A_k thì lúc này mỗi cụm văn bản sẽ có một vector trọng tâm đặc trưng cho từng cụm. Lúc này thay vì tính độ đo Cosin của câu truy vấn với tất cả các vector văn bản trong ma trận A_k theo mô hình LSI thì ta tính độ đo Cosines của vector truy vấn với từng vector trọng tâm của từng cụm. Khi đó, ta trả về các cụm mà có độ đo thỏa một ngưởng cho trước và thực hiện lại việc tính độ đo Cosines của vector truy vấn với các vector văn bản nằm trong các cụm đó. Điều này sẽ giúp cải thiện một cách hiệu quả việc truy tìm thông tin.



Hình 1.6 Kiến trúc của hệ IR dùng mô hình LSI kết hợp thuật toán gom cụm

Đưa ra kiến trúc cơ bản và xây dựng thử nghiệm ba hệ truy tìm thông tin dựa trên mô hình không gian vector, mô hình LSI và mô hình mô hình kết hợp LSI và thuật toán gom cụm văn bản loại HTML bằng ngôn ngữ tiếng Anh.

CHƯƠNG 2
MÔ HÌNH KHÔNG GIAN VECTOR (VSM)
MÔ HÌNH LATENT SEMANTIC INDEX (LSI)

Trong chương này trình bày các nội dung sau:

- Giới thiệu mô hình không gian vector (*VSM*).
- Số hóa văn bản trong mô hình không gian vector và truy vấn.
- Giới thiệu mô hình LSI.
- Phân tích Singular Value Decomposition (*SVD*) trong mô hình LSI.
- Chọn hệ số k và cập nhật lại hệ số k.
- Truy vấn văn bản trong mô hình LSI.

2.1 Mô hình không gian vector (VSM)

2.1.1 Giới thiệu

Mô hình tổng quát của hệ IR là một bộ bốn [D, Q, F, $R(q_i, d_j)$]. Trong đó:

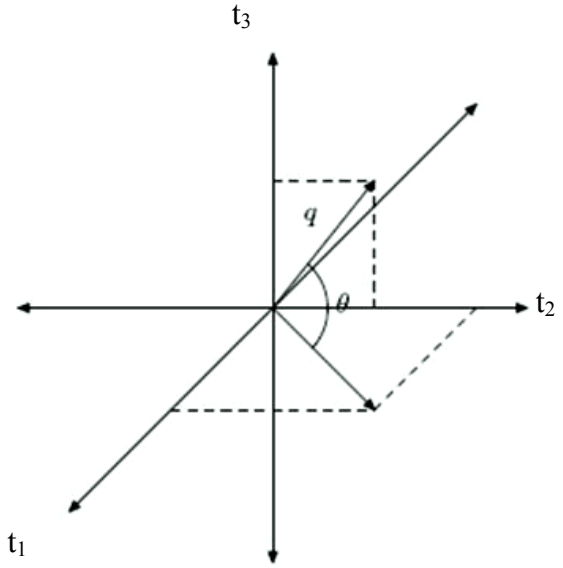
- D là tập văn bản.
- Q là các câu truy vấn.
- F là mô hình biểu diễn tập văn bản, câu truy vấn và các quan hệ của chúng.
- $R(q_i, d_j)$ là hàm xếp hạng theo đo độ tương tự giữa câu truy vấn $q_i \in Q$

và văn bản $d_j \in D$. Hàm xếp hạng xác định một thứ tự về mức độ liên quan của các văn bản với câu truy vấn q_i.

Mô hình không gian vector sẽ làm nhiệm vụ đưa tất cả các văn bản trong tập văn bản được mô tả bởi một tập các từ khoá hay còn gọi là các từ chỉ mục (*index terms*) sau khi đã loại bỏ các từ ít có ý nghĩa (*stop word)*. Các từ chỉ mục này cũng chính là các từ chứa nội dung chính của tập văn bản. Mỗi từ chỉ mục này được gán một trọng số, trọng số của một từ chỉ mục nói lên sự liên quan của nó đến nội dung của một văn bản. Sử dụng các phép toán trên không gian vector để tính toán độ đo tương tự giữa câu truy vấn và các văn bản hoặc các từ chỉ mục, kết quả sau khi tính

toán có thể được xếp hạng theo độ đo tương tự với vector truy vấn.

Mỗi văn bản d được biểu diễn bằng một vector một chiều của các từ chỉ mục $\vec{d}$ =(t1, t2,…, tn) với t_i là từ chỉ mục thứ i (1=<i<=n) trong văn bản d. Tương tự câu truy vấn cũng được biểu diễn bằng một vector $\vec{q}$(q1, q2,…, qn). Lúc đó độ đo tương tự của văn bản d và câu truy vấn q chính là độ đo cosines của chúng.



Hình 2.1 góc giữa vector truy vấn và vector văn bản

2.1.2 Số hóa văn bản theo mô hình không gian vector

2.1.2.1 Cách tổ chức dữ liệu

Trong mô hình không gian vector, mỗi tập văn bản được đại diện bởi một tập các từ chỉ mục, tập từ chỉ mục xác định một "không gian" mà mỗi từ chỉ mục tượng trưng một chiều trong không gian đó. Trong không gian vector văn bản biểu diễn độ đo trọng số (weight) của tập từ chỉ mục tương ứng với văn bản đó.

Ví dụ 2.1: Giả sử tập *A* có *n* văn bản và tập $T = \{t_1, t_2, \ldots, t_m\}$ có *m* từ chỉ mục biểu diễn cho tập văn bản. Vậy không gian vector biểu diễn tập văn bản có số chiều là *m* và mỗi văn bản được biểu diễn bởi một vector *m* chiều. Nếu tập có *m* văn bản sẽ được biểu diễn bởi tập $A = \{d_1, d_2, \ldots, d_n\}$ vector trong không gian vector *n* chiều..

$$A = \begin{pmatrix} d_{11} & d_{21} & \bullet & \bullet & \bullet & d_{1n} \\ d_{12} & d_{22} & \bullet & \bullet & \bullet & d_{2n} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ d_{m1} & d_{m2} & \bullet & \bullet & \bullet & d_{mn} \end{pmatrix}$$

2.1.2.2 Hàm tính trọng số của từ chỉ mục

Định nghĩa một hàm tính trọng số của từ chỉ mục như sau:

$$w_{ij} = l_{ij} \times g_i \times n_j$$

Trong đó:

- l_{ij} là trọng số cục bộ của từ chỉ mục *i* trong văn bản *j* - là hàm đếm số lần xuất hiện của mỗi từ chỉ mục trong một văn bản.

-g_i là trọng số toàn cục của từ chỉ mục *i* - là hàm đếm số lần xuất hiện của mỗi từ chỉ mục trong toàn bộ tập văn bản

- n_j là hệ số được chuẩn hoá của văn bản *j* - là hệ số cân bằng chiều dài của các văn bản trong tập văn bản..

Hàm	Tên hàm	Viết tắt
1 if $f_{ij} > 0$ 0 if $f_{ij} = 0$	*Binary*	BNRY
f_{ij}	*Within_document frequency*	FREQ
$1+ \log f_{ij}$ if $f_{ij} > 0$ 0 if $f_{ij} = 0$	*Log*	LOGA
$(1 + \log f_{ij})/(1+ \log a_j)$ if $f_{ij} > 0$ 0 if $f_{ij} = 0$	*Normalized log*	LOGN
$0.5 + 0.5(f_{ij}/x_j)$ if $f_{ij} > 0$ 0 if $f_{ij} = 0$	*Augumented normalized term frequency*	ATF1

Bảng 2.1 Bảng các hàm tính trọng số cục bộ

Hàm tính trọng số cục bộ được gọi là tốt nếu nó tuân theo nguyên lý: một từ chỉ mục có tần số xuất hiện cao trong một văn bản thì "liên quan" đến văn bản đó hơn. Danh sách các hàm tính trọng số cục bộ trong bảng 3.1.

Hàm tính trọng số cục bộ đơn giản nhất là hàm nhị phân (BNRY) và hàm tính số lần xuất hiện của từ chỉ mục trong văn bản (FREQ):

$$L_{ij} = \begin{cases} 1, & f_{ij} > 0 \\ 0, & f_{ij} = 0 \end{cases} \quad \text{(BNRY) và}$$

$$L_{ij} = f_{ij} \quad \text{(FREQ)}$$

trong đó f_{ij} là số lần xuất hiện của từ chỉ mục i trong văn bản j. Các trọng số nầy thường được sử dụng để tính trọng số câu truy vấn, trong câu truy vấn các từ chỉ mục chỉ xuất hiện một đến hai lần.

Việc sử dụng các hàm này để tính trọng số cục bộ cho văn bản sẽ không tốt bởi vì hàm BNRY không phân biệt sự xuất hiện một lần và nhiều lần của một từ chỉ mục, còn hàm FREQ có trọng số quá lớn với một từ chỉ mục có số lần xuất hiện lớn.

Hàm *logarithms* được sử dụng để điều chỉnh lại số lần xuất hiện của một từ chỉ mục trong một văn bản, bởi vì một từ chỉ mục xuất hiện 10 lần trong một văn bản không hẳn có độ đo quan trọng gấp 10 lần so với một từ chỉ xuất hiện 1 lần. Hai hàm *logarithms* tính trọng số cục bộ trong bảng trên:

$$L_{ij} = \begin{cases} 1 + \log f_{ij} & if \quad f_{ij} > 0 \\ 0 & if \quad f_{ij} = 0 \end{cases} \text{(LOGA) và}$$

$$L_{ij} = \begin{cases} \dfrac{1 + \log f_{ij}}{1 + \log a_j} & if \quad f_{ij} > 0 \\ 0 & if \quad f_{ij} = 0 \end{cases} \text{(LOGN)}$$

trong đó a_j là số lần xuất hiện trung bình của các từ chỉ mục trong văn bản j. Bởi vì hàm LOGN được chuẩn hoá bởi LOGA nên trọng số được cho bởi LOGN sẽ luôn thấp hơn trong số được cho bởi LOGA trong cùng từ chỉ mục và văn bản. Khi trọng số toàn cục không sử dụng, hàm LOGN được sử dụng để chuẩn hoá trọng số cục bộ. Một công thức tính trọng số cục bộ khác là sự kết hợp giữa BNRY và FREQ để tạo thành hàm ATF1:

$$L_{ij} = \begin{cases} 0.5 + 0.5\left(\frac{f_{ij}}{x_j}\right) & if \quad f_{ij} > 0 \\ 0 & if \quad f_{ij} = 0 \end{cases} \quad \text{(ATF1)}$$

trong đó x_j là số lần xuất hiện lớn nhất của các từ chỉ mục trong văn bản j. Với công thức trên, L_{ij} thay đổi từ 0.5 đến 1.0 cho các từ chỉ mục xuất hiện trong văn bản.

Trọng số toàn cục (*global weight*) chỉ giá trị "phân biệt" (*discrimination value*) của mỗi từ chỉ mục trong toàn bộ tập văn bản. Các hàm tính trọng số toàn cục dựa trên ý nghĩa: số lần xuất hiện ít của một từ chỉ mục trong toàn bộ văn bản có giá trị phân biệt cao hơn. Một hàm tính trọng số toàn cục thông dụng là IDF (*inverted document frequency* [10].

Hàm	**Tên hàm**	**Viết tắt**
$\log\left(\frac{N}{n_i}\right)$	*Inverse document frequency*	*IDFB*
$\log\left(\frac{N-n_i}{n_i}\right)$	*Probabilistics inverse*	*IDFP*
$1+\sum_{j=1}^{N}\frac{\frac{f_{ij}}{F_i}\log\frac{f_{ij}}{F_i}}{\log N}$	*Entropy*	*ENPY*
$\frac{F_i}{n_i}$	*Global frequency IDF*	*IGFF*
1	*No global weight*	*NONE*

Bảng 2.2 Bảng các hàm trọng số toàn cục

Ý nghĩa của các tham số trong các hàm:

- *N* là số văn bản trong tập toàn bộ văn bản
- n_i là số văn bản mà từ chỉ mục *i* xuất hiện
- *Fi* là số lần xuất hiện của từ chỉ mục *i* trong toàn bộ văn bản

Một công thức quen thuộc nhất của hệ số chuẩn hoá trong mô hình không gian vector là công thức chuẩn hoá cosines (COSN):

$$N_j = \frac{1}{\sqrt{\sum_{i=0}^{m} \left(G_i L_{ij}\right)^2}}$$

Với hàm COSN, văn bản có nhiều từ chỉ mục sẽ có hệ số chuẩn hoá nhỏ hơn so với các văn bản có từ chỉ mục ít hơn, bởi vì trong tập văn bản chiều dài của các văn bản khác nhau, hệ số này làm cân bằng trọng số của các từ chỉ mục trong tập văn bản.

Mỗi sự kết hợp của 3 hàm tính trọng số cục bộ, toàn cục và hệ số chuẩn hoá có ưu và nhược điểm riêng nên việc chọn lựa sự kết hợp nào phụ thuộc vào người thiết kế hệ thống.

2.1.2.3 Ma trận biểu diễn tập văn bản

Trong mô hình không gian vector một tập có n văn bản được biểu diễn bởi m từ chỉ mục được vector hóa thành ma trận A – ma trận này được gọi là ma trận từ chỉ mục (*term document)*. Trong đó n văn bản trong tập văn bản được biểu diễn thành n vector cột, m từ chỉ mục được biểu diễn thành m dòng. Do đó phần tử d_{ij} của ma trận A chính là trong số của từ chỉ mục i xuất hiện trong văn bản j. Thông thường, trong một tập văn bản số từ chỉ mục lớn hơn rất nhiều so với văn bản m >> n.

Ví dụ 2.1: Giả sử ta có *n = 5* văn bản, mỗi văn bản chỉ có một câu là tiêu đề của một cuốn sách:

D1: How to **Bake** **Bread** without **Recipes**

D2: The Classic Art of Viennese **Pastry**

D3: Numerical **Recipes**: The Art of Scientific Computing

D4: **Breads**, **Pastries**, **Pies** and **Cakes** : Quantity **Baking** **Recipes**

D5: **Pastry**: A Book of Best French **Recipes**

Giả sử có *m = 6* từ chỉ mục cho các văn bản trên – các từ gạch chân

T1: bak(e, ing)

T2: recipes

T3: bread

T4: cake

T5: pastr(y, ies)

T6: pie

Với 5 văn bản và 6 từ chỉ mục ta biểu diễn ma trân term document A_{6x5} như sau:

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

2.1.3 Truy vấn văn bản

Trong mô hình không gian vector, việc truy vấn tập dữ liệu văn bản để tìm những văn bản liên quan với câu truy vấn dựa vào các kỹ thuật tính toán trên mô hình không gian vector. Một câu truy vấn được xem như tập các từ chỉ mục và được biểu diễn như các văn bản trong tập văn bản.Vì câu truy vấn rất ngắn nên có rất nhiều từ chỉ mục của tập văn bản không xuất hiện trong câu truy vấn, có nghĩa là hầu hết các thành phần của vector truy vấn là zero. Thủ tục truy vấn chính là tìm các

văn bản trong tập văn bản liên quan với câu truy vấn hay còn gọi là các văn bản có độ đo tương tự "cao" với câu truy vấn. Theo cách biểu diễn hình học, các văn bản được chọn là các văn bản gần với câu truy vấn nhất theo một độ đo (*measure*) nào đó.

Độ đo thường được sử dụng nhất là độ đo *cosines* của góc giữa vector truy vấn và vector văn bản. Nếu ma trận term – document *A* có các cột được ký hiệu là d_j , $j = 1, ..., n$ thì *n* độ đo *cosines* của vector truy vấn *q* với *n* văn bản trong tập văn bản được tính theo công thức:

$$\cos\theta_j = \frac{d_j{}^T q}{\|d_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^{m} d_{ij} q_i}{\sqrt{\sum_{i=1}^{m} d_{ij}{}^2}\sqrt{\sum_{i=1}^{m} q_i{}^2}} \qquad (2.1)$$

Sử dụng tập văn bản trong ví dụ 2.1 ở trên để ví dụ cho thủ tục truy vấn, dựa trên công thức (2.1) tính góc của các vector trong không gian vector 6 chiều ($\Re^6$). Giả sử người sử dụng cần những thông tin về nấu ăn và muốn tìm kiếm các cuốn sách về *baking bread*. Với câu truy vấn trên tương ứng với vector truy vấn là:

$$q^{(1)} = (1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)^T$$

với các phần tử khác không cho hai từ *baking* và *bread*. Việc tìm kiếm các văn bản liên quan được thực hiện bằng cách tính *cosines* của các góc θ_j giữa vector truy vấn $q^{(1)}$ với các vector văn bản d_j bằng công thức (2.1). Một văn bản được xem như liên quan (*relevant*) và được trả về nếu *cosines* của góc được tạo bởi vector truy vấn và vector văn bản đó lớn hơn một ngưỡng (*threshold*) cho trước. Trong cài đặt thực tế ngưỡng được kiểm nghiệm và quyết định bởi người xây dựng hệ thống. Nhưng đối với ví dụ nhỏ này chỉ sử dụng ngưỡng là 0.5.

Với vector truy vấn $q^{(1)}$, chỉ có giá trị *cosines* của các góc khác *zero*: $\cos\theta_1 = 0.8165$ và $\cos\theta_4 = 0.5774$. Vậy các văn bản liên quan đến *baking* và

bread D1 và *D4* được trả về, các văn bản *D2*, *D3* và *D5* không liên quan và được bỏ qua.

Nếu người sử dụng chỉ muốn tìm các cuốn sách về *baking*, thì kết quả sẽ khác, trong trường hợp này vector truy vấn là:

$$q^{(2)} = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T ,$$

và *cosines* của các góc giữa vector truy vấn và 5 vector văn bản theo thứ tự là: **0.5774**, **0**, **0**, **0.4082**, và **0**. Vì vậy chỉ văn bản *D1*, là cuốn sách về *baking bread* thoả ngưỡng cho trước 0.5 và được trả về. Văn bản thứ tư *D4* thực sự là có liên quan đến chủ đề *baking* mà người sử dụng cần nhưng không được trả về.

Đây là một điểm yếu của mô hình không gian vector. Để khắc phục điểm yếu này của mô hình không gian vector, một mô hình rất hiệu quả gần đây được đề nghị - mô hình Latent Semantic Indexing (*LSI*).

2.2 Mô hình Latent Semantic Index(*LSI*).

2.2.1 Giới thiệu

Mô hình không gian vector được nếu như số lượng từ chỉ mục tăng rất lớn thì kích thước của ma trận từ chỉ mục (term document) A cũng tăng theo rất lớn. Hơn nữa độ đo Cosines giữa vector truy vấn và vector văn bản là phải khác Zero nếu và chỉ nếu tồn tại ít nhất từ chỉ mục giữa 2 vector trên.

Latent Semantic Indexing (*LSI*) là phương pháp tạo chỉ mục tự động dựa trên khái niệm để khắc phục hai hạn chế tồn tại trong mô hình không gian vector chuẩn về hai vấn đề ***synoymy*** và ***polysemy*** [7], [8], [9]. Với synoymy, nhiều từ có thể được sử dụng để biểu diễn một khái niệm, vì vậy hệ thống không thể trả về những văn bản liên quan đến câu truy vấn của người dùng khi họ sử dụng những từ trong câu truy vấn đồng nghĩa với những từ trong văn bản. Với polysemy, một từ có thể có nhiều nghĩa, vì vậy hệ thống có thể trả về những văn bản không liên quan. Điều

này thực tế rất thường xảy ra bởi vì các văn bản trong tập văn bản được viết bởi rất nhiều tác giả, với cách dùng từ rất khác nhau. Một cách tiếp cận tốt hơn cho phép người dùng truy vấn văn bản dựa trên khái niệm (*concept*) hay nghĩa (*meaning*) của văn bản.

Mô hình LSI cố gắng khắc phục hai hạn chế trên trong mô hình không gian vector bằng cách chỉ mục khái niệm được tạo ra bởi phương pháp thống kê (*phân tích SVD ma trận term – document A*) thay cho việc sử dụng các từ chỉ mục đơn. Mô hình LSI dựa trên giả thiết là có các ngữ nghĩa tiềm ẩn (*latent semantic*) trong việc sử dụng từ: có nhiều từ biểu diễn cho một khái niệm và một khái niệm có thể được biểu diễn bởi nhiều từ. Mô hình LSI sử dụng phân tích SVD (*Singular Value Decomposition*) ma trận term – document *A* để phát hiện ra các quan hệ ngữ nghĩa trong cách dùng từ trong toàn bộ văn bản .

2.2.2 Phân tích *Singular Value Decomposition* (SVD) của ma trận từ chỉ mục (term document A)

Vấn đề cơ bản của mô hình LSI là phân tích SVD của ma trận term document A. Nó được biểu diễn như sau:

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

Trong đó:

- *U* là ma trận trực giao cấp *m* x *r* (*m số từ chỉ mục*) các vector dòng của U là các vector từ chỉ mục.

- $\sum$ là ma trận đường chéo cấp r x r có các giá trị suy biến (*singular value*) $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$, với *r = rank(A)*.

- *V* là ma trận trực giao cấp *r* x *n* (n *số văn bản trong tập văn bản*) - các vector cột của V là các vector văn bản.

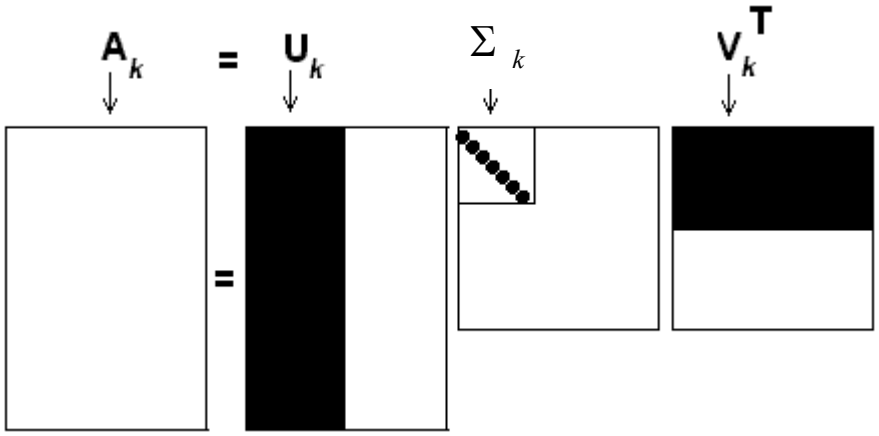
Ví dụ 2.2.2.1: Ta quay lại ví dụ 2.1.3.1 ở trên, phân tích SVD của ma trận term – document $A = U\Sigma V^T$ trong đó:

$$U = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 & -0.7071 & 0 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 & 0 & 0 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 & -0.7071 & 0 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 & 0 & -0.7071 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 & 0 & 0 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 & 0 & 0.7071 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1.6950 & 0 & 0 & 0 \\ 0 & 1.1158 & 0 & 0 \\ 0 & 0 & 0.8403 & 0 \\ 0 & 0 & 0 & 0.4195 \end{pmatrix}$$

$$V = \begin{pmatrix} 0.4366 & -0.4717 & 0.3688 & -0.6715 & 0 \\ 0.3067 & 0.7549 & 0.0998 & -0.2760 & -0.5000 \\ 0.4412 & -0.3568 & -0.6247 & 0.1945 & -0.5000 \\ 0.4909 & -0.0346 & 0.5711 & 0.6571 & 0 \\ 0.5288 & 0.2815 & -0.3712 & -0.0577 & 0.7071 \end{pmatrix}$$

Ma trận xấp xỉ $A_k = U_k \Sigma_k V_k^T$ có hạng là *k* với k << r . Trong đó, các cột của U_k là k cột đầu tiên của U, các cột của V_k là k cột đầu tiên của của V và Σ_k là ma trận đường chéo cấp *k* x *k* với các phần tử nằm trên đường chéo là *k* giá trị suy biến lớn nhất của *A*.



Hình 2.2 Biểu diễn ma trận xấp xỉ A_k có hạng là k

Ví dụ 2.2: cho k=3, Tìm ma trận xấp xỉ $A_3 = U_3 \sum_3 V_3^T$:

$$A_3 = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 \\ 0.7479 & -0.3981 & -0.5249 \\ 0.2670 & -0.2567 & 0.5308 \\ 0.1182 & -0.0127 & 0.2774 \\ 0.5198 & 0.8423 & -0.1158 \\ 0.1182 & -0.0127 & 0.6394 \end{pmatrix} \begin{pmatrix} 1.6950 & 0 & 0 \\ 0 & 1.1158 & 0 \\ 0 & 0 & 0.8403 \end{pmatrix} \begin{pmatrix} 0.4366 & -0.4717 & 0.3588 \\ 0.3067 & 0.7549 & 0.0998 \\ 0.4412 & -0.3568 & -0.6247 \\ 0.4909 & -0.0346 & 0.5711 \\ 0.5288 & 0.2815 & -0.3712 \end{pmatrix}^T$$

$$A_3 = \begin{pmatrix} 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.6003 & 0.0094 & 0.9933 & 0.3858 & 0.7091 \\ 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \\ -0.0326 & 0.9866 & 0.0094 & 0.4402 & 0.7043 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \end{pmatrix}$$

Trong mô hình LSI, ma trận A_k là xấp xỉ của ma trận từ chỉ mục (term – document *A)* được tạo ra có ý nghĩa rất quan trọng: phát hiện sự kết hợp ngữ nghĩa giữa các từ chỉ mục được sử dụng trong toàn bộ tập văn bản, loại bỏ những thay đổi trong cách sử dụng từ gây ảnh hưởng xấu đến phương pháp truy tìm theo từ chỉ mục[7], [8], [9]. Vì sử dụng không gian LSI *k* chiều, nhỏ hơn rất nhiều so với số từ chỉ mục (*m từ chỉ mục*) nên sự khác nhau không quan trọng trong các từ "đồng nghĩa" được loại bỏ. Những từ chỉ mục thường xuyên xuất hiện cùng nhau trong các văn bản sẽ nằm gần nhau khi biểu diễn trong không gian LSI *k* chiều, ngay cả các từ chỉ mục không đồng thời xuất hiện trong cùng một văn bản. Vì vậy, các văn bản không chứa các từ chỉ mục xuất hiện trong câu truy vấn cũng có thể có độ đo tương tự cao với câu truy vấn [9].

Lấy một ví dụ nhỏ, ta xét các từ chỉ mục *car*, *automobile*, *driver* và *elephant.* Từ *car* và *automobile* đồng nghĩa, *driver* cũng có quan hệ về nghĩa với *car* và *automobile*, còn *elephant* thì hoàn toàn không. Trong các hệ thống truy tìm thông

tin truyền thống, truy tìm các văn bản sử dụng từ *automobile* hệ thống không thể truy tìm các văn bản về *car* hơn các văn bản về *elephant* nếu văn bản đó không sử dụng từ *automobile*, cho dù *car* đồng nghĩa với *automobile*. Điều nầy làm giảm độ đo ***precision*** và ***precall*** của hệ thống. Hệ thống hoạt động hiệu quả hơn nếu truy vấn văn bản về *automobile* cũng truy xuất các văn bản về *car* và ngay cả các văn bản về *driver*. Việc sử dụng mô hình LSI có thể biểu diễn mối quan hệ hữu ích nầy giữa các từ chỉ mục trong toàn bộ văn bản, giúp cho hệ thống hoạt động hiệu quả hơn. Các từ *car* và *automobile* xuất hiện cùng đồng thời xuất hiện với nhiều từ (ví dụ: *motor*, *model*, *vehicle*, *chassis*, *carmakers*, *sedan*, *engine*,…) sẽ được biểu diễn gần nhau trong không gian LSI k chiều [7]. Mục tiêu chính của mô hình LSI là biểu diễn tường minh mối quan hệ tiềm ẩn của các từ chỉ mục nhằm tăng hiệu truy tìm của hệ thống .

2.2.3 Truy vấn trong mô hình LSI

Để truy vấn trong mô hình LSI, vector truy vấn q được so sánh với các vector cột trong ma trận A_k của ma trận term – document A. Gọi e_j là vector đơn vị thứ j có số chiều n (*cột thứ j của ma trận đơn vị n* x n), vector cột thứ j của ma trận A_k là $A_k e_j$. Độ đo *cosines* của các góc giữa vector truy vấn q và các vector văn bản trong ma trận A_k được tính:

$$\cos\theta_j = \frac{\left(A_k e_j\right)^T q}{\left\|A_k e_j\right\|_2 \|q\|_2} = \frac{\left(U_k \Sigma_k V_k^T e_j\right)^T q}{\left\|U_k \Sigma_k V_k^T e_j\right\|_2 \|q\|_2} = \frac{e_j^T V_k \Sigma_k \left(U_k^T q\right)}{\left\|\Sigma_k V_k^T e_j\right\|_2 \|q\|_2} \quad (2.2)$$

với $j = 1, \ldots, n$. Nếu ta gọi vector $s_j = \Sigma_k V_k^T e_j$, công thức (2.2) được viết lại:

$$\cos\theta_j = \frac{s_j^T \left(U_k^T q\right)}{\left\|s_j\right\|_2 \|q\|_2}, \qquad j = 1, \ldots, n \qquad (2.3)$$

Các chuẩn $\left\|s_j\right\|_2$ được tính toán chỉ 1 lần cho mỗi ma trận term – document A và sau đó được sử dụng cho tất cả các truy vấn. Trong công thức (2.3) k thành phần

đầu tiên của vector s_j là toạ độ của cột thứ j của ma trận A_k trong cơ sở được xác định bởi các vector cột trong ma trận U_k. Ngoài ra k thành phần của vector $U_k^T q$ là toạ độ của phép chiếu $U_k U_k^T q$ của vector truy vấn q vào không gian cột của ma trận A_k. Vì vậy, thay vì sử dụng vector truy vấn q, ta sử dụng vector $U_k^T q$ là một hình chiếu của q.

$$\cos\theta_j' = \frac{s_j^T \left(U_k^T q\right)}{\left\|s_j\right\|_2 \left\|U_k^T q\right\|_2}, \quad j = 1,\ldots,n. \qquad (2.4)$$

Trong công thức (2.4), tính *cosines* chỉ sử dụng vector k chiều sau một lần tính $U_k^T q$. Bởi vì vector truy vấn q thường rất thưa (*đa số phần tử bằng zero*), nên chi phí tính toán $U_k^T q$ thấp. Đối với tất cả các vector văn bản, $\cos\theta_j' \geq \cos\theta_j$ vì vậy độ đo ***recall*** có thể tăng lên và độ đo ***precision*** có thể giảm khi sử dụng (2.4) thay cho (2.3).

Khi cài đặt hệ IR thực tế ta chỉ lưu và tính toán trên ba ma trận U_k, Σ_k và V_k, cải thiện rất nhiều chi phí lưu trữ và tính toán.

Quay lại câu truy vấn $q^{(1)}$(***baking bread***) trong ví dụ 2.1. Sử dụng ma trận xấp xỉ A_3 ($k=3$) và công thức (2.2), các độ đo *cosines* là: **0.7327**, **- 0.0469**, **0.0330**, **0.7161** và **- 0.0097**. Hai văn bản *D1* và *D4* được trả về, có độ đo *cosines* với q rất gần nhau (*D1*: **0.7327**, *D4*: **0.7161**). Độ đo *cosines* của các vector văn bản khác không còn zero nữa, nhưng vẫn còn rất nhỏ so với ngưỡng 0.5, nghĩa là các văn bản không liên quan vẫn không được trả về. Sử dụng vector truy vấn $q^{(2)}$(***baking***) và A_3 kết quả là: **0.5181**, **-0.1107**, **0.5038**, **0.3940**, và **0.2362**, vì vậy cả hai văn bản về **baking** là *D1* và *D3* được trả về, với 2 độ đo *cosines* với vector q rất gần nhau (*D1*: **0.5181**, *D3*: **0.5038**). Với kết quả ở ví dụ nhỏ trên ta thấy kết quả trên mô hình LSI tốt hơn so với mô hình không gian vector (*VSM*) chuẩn.

Có một cách tiếp cận khác cho thủ tục truy vấn trong mô hình LSI, các văn bản có thể được so sánh với nhau bằng cách tính độ đo *cosines* các vector văn bản trong "không gian văn bản" (*document space*) – chính là so sánh các vector cột trong ma trận V_k^T. Một câu truy vấn *q* được xem như là một văn bản và giống như một vector cột được thêm vào ma trận V_k^T. Để thêm *q* như một cột mới vào V_k^T ta phải chiếu *q* vào không gian văn bản *k* chiều.

Từ công thức ma trận $A_k = U_k \Sigma_k V_k^T$ ta suy ra $\Sigma_k^{-1} U_k^T A_k = V_k^T$ (vì $U_k U_k^T = I_k$) vậy ta có $V_k = A_k^T U_k \Sigma_k^{-1}$.

Áp dụng tương tự cho vector truy vấn *q*: $q_k = q^T U_k \Sigma_k^{-1}$

Cuối cùng ta tính độ đo *cosines* của vector q_k với các vector văn bản trong ma trận V_k^T:

$$\cos \theta_j = \frac{q_k \left(V_k^T\right)_j}{\|q_k\|_2 \left\|\left(V_k^T\right)_j\right\|_2} \qquad (2.5)$$

với $\left(V_k^T\right)_j$ là vector cột thứ *j* của ma trận V_k^T.

2.2.4 Cập Nhật Singular Value Decomposition (SVD)

2.2.4.1 Cập Nhật Văn Bản (*SVD- Updating document)*:

Giả sử $A \in \Re^{m \times n}$ là ma trận từ chỉ mục (term – document) đã được tạo, và $A_k = U_k \Sigma_k V_k^T$ là xấp xỉ tốt nhất của *A* có hạng *k*. Cho $D \in \Re^{m \times p}$ là *p* văn bản mới được thêm vào tập văn bản. Công việc ở đây chính là tính xấp xỉ tốt nhất có hạng *k* của ma trận: $B \equiv \left(A_k, D\right)$ [5], [9].

Sử dụng công thức phân tích A_k ma trận *B* có thể được viết lại như sau:

$$B = \left(U_k \Sigma_k V_k^T, D\right)$$

$$= \left(U_k, \left(I_m - U_k U_k^T\right) D\right) \begin{pmatrix} \Sigma_k & U_k^T \\ 0 & I_p \end{pmatrix} \begin{pmatrix} V_k^T & 0 \\ 0 & I_p \end{pmatrix}$$

trong đó $\left(I_m - U_k U_k^T\right)$ là ma trận biểu diễn cho phép chiếu trực giao các cột của ma trận *D* vào không gian con (*subspace*) $P_k^{\perp}$ trực giao với không gian được tạo bởi các cột của ma trận P_k. Gọi $\left(I_m - U_k U_k^T\right)D = \hat{U}_p R$ là phân tích QR của ma trận $\left(I_m - U_k U_k^T\right)D$ thì

$$B = \left(U_k, \hat{U}_p\right)\begin{pmatrix}\Sigma_k & U_k^T D\\ 0 & R\end{pmatrix}\begin{pmatrix}V_k^T & 0\\ 0 & I_p\end{pmatrix} \qquad (2.6)$$

với $\left(U_k, \hat{U}_p\right)$ là ma trận trực chuẩn. Phân tích SVD của

$$\hat{B} = \begin{pmatrix}\Sigma_k & U_k^T D\\ 0 & R\end{pmatrix} = \left(P_k, P_k^{\perp}\right)\begin{pmatrix}\hat{\Sigma}_k & 0\\ 0 & \hat{\Sigma}_p\end{pmatrix}\left(Q_k, Q_k^{\perp}\right)^T$$

trong đó $P_k, Q_k \in \Re^{(k+p)\times k}$ và $\hat{\Sigma}_k \in \Re^{k\times k}$. Ma trận xấp xỉ tốt nhất có hạng *k* của *B* là:

$$B_k = \left(U_k \quad \hat{U}_k\right) P_k \hat{\Sigma}_k \left(\begin{pmatrix}V_k & 0\\ 0 & I_p\end{pmatrix} Q_k\right)^T \qquad (2.7)$$

Thuật toán cập nhật SVD văn bản [4]:

1. Input: *k*, $U_k \in \Re^{m\times k}, \Sigma_k \in \Re^{k\times k}, V_k \in \Re^{n\times k}, D \in \Re^{m\times p}$
2. Tính phép chiếu: $\hat{D} = \left(I_m - U_k U_k^T\right)D$
3. Tính phân tích QR: $\hat{D} = \hat{U}_p R$, với $\hat{U}_p \in \Re^{m\times p}, R \in \Re^{p\times p}$
4. Tính SVD ma trận: $\hat{B} \equiv \begin{pmatrix}\Sigma_k & U_k^T D\\ 0 & R\end{pmatrix} \in \Re^{(k+p)(k+p)}$

   $$\hat{B} = \left(P_k, P_k^{\perp}\right)\begin{pmatrix}\hat{\Sigma}_k & 0\\ 0 & \hat{\Sigma}_p\end{pmatrix}\left(Q_k, Q_k^{\perp}\right)^T,$$

   trong đó $P_k, Q_k \in \Re^{(k+p)\times k}$ và $\hat{\Sigma}_k \in \Re^{k\times k}$
5. Output: Xấp xỉ tốt nhất của ma trận $B = (A_k, D)$ là:

$$B_k \equiv \left[\left(U_k, \hat{U_k}\right)P_k\right].\hat{\Sigma_k}\left[\begin{pmatrix} V_k & 0 \\ 0 & I_p \end{pmatrix}Q_k\right]^T$$

2.2.4.2 Cập Nhật từ chỉ mục (terms):

Giả sử $T \in \Re^{q\times n}$ là q vector từ chỉ mục (*vector terms*) mới được thêm vào. Tính ma trận ma trận xấp xỉ tốt nhất có hạng k của ma trận [5], [9]:

$$B \equiv \begin{pmatrix} A_k \\ T \end{pmatrix} \quad (2.8)$$

Trước hết thay thế SVD của A_k vào công thức (2.8):

$$\begin{aligned} B &= \begin{pmatrix} A_k \\ T \end{pmatrix} = \begin{pmatrix} U_k \Sigma_k V_k^T \\ T \end{pmatrix} \\ &= \begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}\left[\begin{pmatrix} \Sigma_k \\ TV_k \end{pmatrix}V_k^T + \begin{pmatrix} 0 \\ T\left(I_n - V_k V_k^T\right) \end{pmatrix}\right] \\ &= \begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}\begin{pmatrix} \Sigma_k & 0 \\ TV_k & I_q \end{pmatrix}\begin{pmatrix} V_k^T \\ T\left(I_n - V_k V_k^T\right) \end{pmatrix} \\ &= \begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}\begin{pmatrix} \Sigma_k & 0 \\ TV_k & I_q \end{pmatrix}\left(V_k \left(I_n - V_k V_k^T\right)T^T\right)^T \end{aligned}$$

phân tích QR ma trận $\left(I_n - V_k V_k^T\right)T^T = \hat{V_k}.L_q$

Vậy ma trận B có thể viết lại như sau:

$$B = \begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}\begin{pmatrix} \Sigma_k & 0 \\ TV_k & L_q \end{pmatrix}\left(V_k, \hat{V_k}\right)^T$$

Phân tích SVD ma trận $\hat{B} = \begin{pmatrix} \Sigma_k & 0 \\ TV_k & L_q \end{pmatrix} = \left(P_k, P_k^{\perp}\right)\begin{pmatrix} \hat{\Sigma_k} & 0 \\ 0 & \hat{\Sigma_q} \end{pmatrix}\left(Q_k, Q_k^{\perp}\right)^T$

trong đó $P_k, Q_k \in \Re^{(k+q)\times k}$ và $\hat{\Sigma}_k \in \Re^{k\times k}$.

Vậy xấp xỉ tốt nhất của B có hạng k là:

$$B_k = \left[\begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}P_k\right].\hat{\Sigma_k}\left[\left(V_k, \hat{V_k}\right)Q_k\right]^T \quad (2.9)$$

Thuật toán cập nhật từ chỉ mục(SVD- Updating Terms):

1. Input: k, $U_k \in \Re^{m\times k}, \Sigma_k \in \Re^{k\times k}, V_k \in \Re^{n\times k}, T \in \Re^{m\times q}$

2. Tính phép chiếu: $\hat{T} = \left(I_n - V_k V_k^T\right)T^T \in \Re^{n\times q}$

3. Tính phân tích QR: $\hat{T} = \hat{V}_k L_q$, với $\hat{V}_q \in \Re^{m\times q}, L_q \in \Re^{q\times q}$

4. Tính SVD ma trận: $\hat{B} \equiv \begin{pmatrix} \Sigma_k & 0 \\ TV_k & L_q \end{pmatrix} \in \Re^{(k+q)(k+q)}$

$$\hat{B} = \left(P_k, P_k^{\perp}\right)\begin{pmatrix} \hat{\Sigma}_k & 0 \\ 0 & \hat{\Sigma}_q \end{pmatrix}\left(Q_k, Q_k^{\perp}\right)^T,$$

trong đó $P_k, Q_k \in \Re^{(k+q)\times k}$ và $\hat{\Sigma}_k \in \Re^{k\times k}$

5. Output: Xấp xỉ tốt nhất của ma trận $B = \begin{pmatrix} A_k \\ T \end{pmatrix}$ là :

$$B_k = \left[\begin{pmatrix} U_k & 0 \\ 0 & I_q \end{pmatrix}P_k\right].\hat{\Sigma}_k\left[\left(V_k, \hat{V}_k\right)Q_k\right]^T$$

2.2.4.3 Loại bỏ từ chỉ mục (Downdating) Trong Mô Hình LSI

Thao tác xoá từ chỉ mục hay văn bản còn gọi là folding-out, trong mô hình LSI chỉ đơn giản xoá các vector từ chỉ mục trong ma trận U_k và các vector văn bản trong ma trận V_k^T [9].

2.2.5 Chọn hệ số k trong mô hình LSI

Trong mô hình LSI, việc chọn hệ số k là một việc hết sức quan trọng đến hiệu quả của thuật toán. Việc chọn hệ số k như thế nào là tối ưu vẫn còn là một bài toán mở, chọn hệ số k quá nhỏ hay quá lớn cũng ảnh hưởng đến hiệu quả truy tìm của thuật toán. Theo các tài liệu nghiên cứu về LSI [2], [7] qua thực nghiệm trên các tập dữ liệu văn bản cụ thể, các tác giả chọn *k* từ 50 đến 100 cho các tập dữ liệu nhỏ và từ 100 đến 300 cho các tập dữ liệu lớn.

Tuy nhiên các nghiên cứu trên chỉ đưa ra con số *k* cụ thể dựa vào thực nghiệm trên các tập dữ liệu mẫu cụ thể. Về tổng quát không thể sử dụng các con số trên cho các ứng dụng thực tế khi mà tập dữ liệu có thể chưa xác định trước (*có thể tập dữ liệu rất nhỏ hoặc rất lớn*). Một phương pháp đề nghị chọn hệ số *k* gần đây nhất (2003) được đưa ra bởi Miles Efron trong tài liệu [8], tác giả sử dụng phương pháp phân tích giá trị riêng (*Eigenvalue*) của ma trận từ chỉ mục (Term – Document *A)* và sử dụng kiểm định thống kê để chọn hệ số *k* tốt nhất trên dãy các hệ số *k* được chọn thử nghiệm. Tác giả đã thử nghiệm phương pháp của tác giả trên hai tập dữ liệu mẫu chuẩn là MEDLINE và CISI.

Tập văn bản mẫu thử nghiệm	Số văn bản	Số câu truy vấn mẫu
MEDLINE	1033	30
CISI	1460	100

Bảng 2.3: tập dữ liệu thử nghiệm MEDLINE và CISI

.

Ta có thể tính độ sai số của phép xấp xỉ tạo ma trận A_k từ ma trận từ chỉ mục (Term – Document) *A* bằng công thức [1]:

$$\min_{rank(B)=k} \|A-B\|_F^2 = \|A-A_k\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_{rank(A)}^2 \quad (2.10)$$

từ công thức (2.10) ta có thể tính được tỉ lệ thay đổi của ma trận A_k có hạng *k* so với ma trận *A* ban đầu là

$$s = \frac{\|A - A_k\|_F^2}{\|A\|_F^2} = \frac{\sum_{i=k+1}^{rank(A)} \sigma_i^2}{\sum_{j=1}^{rank(A)} \sigma_j^2} \qquad (2.11)$$

Vậy thay vì chọn hệ số *k* thủ công, hệ thống có thể tự động chọn hệ số *k* dựa vào tỉ số thay đổi của A_k so với *A* theo chuẩn *F*. Bài toán bây giờ cần giải quyết là chọn sai số *s* nào là tốt.

Gọi *error* là độ đo sai số cho phép giảm hạng ma trận term – document *A* thành ma trận A_k có hạng *k* thì:

$$error = \begin{cases} \|A - A_k\|_F = \left(\sum_{i=k+1}^{rank(A)} \sigma_i^2\right)^{\frac{1}{2}} & \textit{Sai số tuyệt đối} \\ \dfrac{\|A - A_k\|_F}{\|A\|_F} & \textit{Sai số tương đối} \end{cases}$$

Thuật toán chọn hệ số *k* theo tỉ lệ sai số tương đối *error* cho trước.

Input: Ma trận đường chéo Σ , sai số *error*

Output : hệ số *k*

Thuật toán:

For(*k* = rank(*A*); *k* > 0 ; *k* --)

{

$$s = \frac{\|A - A_k\|_F^2}{\|A\|_F^2} = \frac{\sum_{i=k+1}^{rank(A)} \sigma_i^2}{\sum_{j=1}^{rank(A)} \sigma_j^2};$$

if (*s* < *error*)

continue;

else

return k + 1 ;

}

Thuật toán trên có chi phí không cao, bởi vì việc tính các chuẩn *F* ta chỉ tính tổng các phần tử trên đường chéo chính của ma trận Σ.

CHƯƠNG 3
KẾT HỢP THUẬT TOÁN K-MEANS VÀ MÔ HÌNH LSI VÀO BÀI TOÁN GOM CỤM VĂN BẢN

Trong chương này trình bày các nội dung sau:

- Giới thiệu các kỹ thuật gom cụm.
- Thuật toán gom cụm K-means.
- Bước cải tiến thuật toán K-means trong luận văn.
- Truy vấn văn bản trong gom cụm.
- Đánh giá hiệu quả của việc kết hợp thuật toán gom cụm với mô hình LSI.
- Cập nhật văn bản cho cụm

3.1 Giới thiệu:

Với các kỹ thuật áp dụng trong hệ truy tìm thông tin được trình bày tổng quát ở chương 1 thì trong chương 2 trình bày mô hình không gian vector và mô hình cải tiến LSI áp dụng vào hệ truy tìm thông tin. Do câu truy vấn rất ngắn nên có rất nhiều từ chỉ mục của tập văn bản không xuất hiện trong câu truy vấn, có nghĩa là hầu hết các thành phần của vector truy vấn là zero. Điều này có nghĩa là có một số văn bản có liên quan đến câu truy vấn nhưng không được trả về. Đây là một điểm yếu của mô hình không gian vector.

Để khắc phục nhược điểm của mô hình không gian vector thì trong mô hình LSI – một mô hình rất hiệu quả mà gần đây đã được quan tâm rất nhiều cho việc ứng dụng vào hệ truy tìm thông tin. Trong mô hình này không những trình bày giảm số chiều rất nhiều của ma trận từ chỉ mục (term document) A, mà quan trọng là tìm chính xác về nghĩa của tập văn bản và trả về một cách chính xác các văn bản mà người dùng cần tìm kiếm [7], [8], [9]. Tuy nhiên để trả về các tập văn bản mà người dùng cần tìm thì mô hình LSI phải đi tính độ đo Cosines của tất cả các tập văn bản trong ma trận xấp xỉ A_k. Điều này dẫn đến việc hạn chế tốc độ tìm kiếm của giải

thuật.

Trong luận văn này đề nghị áp dụng bài toán gom cụm vào hệ truy tìm thông tin cụ thể là thuật toán K-means với hai phương pháp cải tiến: Tiền xử lý ma trận từ chỉ mục (term – document) A và áp dụng hợp lý độ đo khoảng cách. Sau đó, đem kết quả đạt được so sánh với mô hình không gian vector và mô hình cải tiến LSI.

3.2 Thuật toán gom cụm K-Means

Kỹ thuật sử dụng các bài toán gom cụm vào hệ truy tìm thông tin trong những năm gần đây khá phổ biến. Hiện nay có rất nhiều thuật toán gom cụm đã giới thiệu trong chương một như: K-means, Fuzzy K-means, hierachical, DBScan, gom cụm phẳng bằng mạng Kohonen… Tuy nhiên, tất cả các giải thuật toán này về cơ bản đều dựa trên cách thức phân loại các vector đặc trong không gian đa chiều. Hay nói cách khác là chúng đều gom cụm trên tập ma trận từ (terms) với số từ (terms) của tập văn bản rất lớn.

3.2.1 Thuật toán K-means

Đầu vào của thuật toán: số cụm *k,* và CSDL có *n* đối tượng

Thuật toán gồm 4 bước :

- Chọn bất kỳ *k* đối tượng làm các tâm (centroids) ban đầu.
- Gán hoặc gán lại từng đối tượng vào cụm với khoảng cách gần nhất.
- Cập nhật centroids.
- Quay về bước 2, dừng khi không còn phép gán mới.

3.2.2 Thời gian và độ phức tạp của thuật toán K-means.

- Độ phức tạp của việc chọn k cụm ban đầu là O(k).
- Tính khoảng cách của n đối tượng với trọng tâm của từng cụm là O(kn).
- Thời gian cập nhật lại centriods là O(n).
- Thời gian của bước 4 là O(n)

Vậy độ phức tạp của thuật toán K-means là:

$$O(k) + t(O(kn) + O(n) + O(n)) = O(tkn)$$

3.2.3 Ưu điểm và nhược điểm của K-means áp dụng vào hệ IR

Ưu điểm:

- Hiệu suất tương đối: do t, k << n (t là số lần lặp, k là số cụm, n là tập văn bản) cho nên có sự thực thi rất tốt trong hầu hết các ứng dụng.
- Scalable tương đối: trong khi xử lý các tập dữ liệu lớn.

Nhược điểm:

- Vì phải xác định trước số cụm trong không gian từ chỉ mục của tập văn bản (terms document) nên dẫn tới không thể chọn một cách tối ưu số cụm. Nếu chọn số cụm quá ít thì dẫn đến các văn bản có độ tương tự và không tương tự nhau sẽ thuộc về cùng một cụm. Nếu chọn số cụm quá nhiều dẫn đến độ hội tụ của các văn bản tương tự bị giảm.
- Do sử dụng cách tính khoảng cách trọng tâm từ các đối tượng đến cụm không phù hợp để khám phá các cụm với dạng không lồi hay cụm có kích thước khác nhau. Điều này, sẽ làm gây trở ngại cho việc xác định độ tương tự cho các văn bản trả về khi người dùng thực hiện truy vấn.

3.2.4 Bước cải tiến thuật toán gom cụm K-means trong luận văn so với thuật toán gom cụm K-means cổ điển.

K-Means là một thuật toán được áp dụng khá nhiều trong gom cụm dữ liệu vì hiệu năng và tính hiện thực khá tốt. Tuy nhiên ngoài việc cần cho trước số cụm, K-Means còn đòi hỏi phải chọn trước k điểm làm trọng tâm, việc chọn ngẫu nhiên này có thể cho ra các kết quả khác nhau. Do đó, đã xuất hiện hàng loạt những thuật toán cải tiến nhằm tạo ra các cụm có hiệu quả cao và hiệu quả về mặt thời gian. Các biến thể K-means này khác nhau ở chổ:

- Chọn *k* centroids ban đầu.
- Tính toán sự bất tương tự.

- Các chiến lược tính centroids cụm.

Từ những ưu nhược điểm của thuật toán K-means cổ điển áp dụng vào hệ truy tìm thông tin, luận văn đền xuất các biện pháp cải tiến cho thuật toán K-means áp dụng vào hệ truy tìm thông tin:

3.2.4.1 Tiền xử lý tập dữ liệu vào

Đối với hệ truy tìm thông tin tập văn bản rất lớn kéo theo từ chỉ mục cũng rất lớn. Nếu có tập N văn bản và M từ chỉ mục (M,N >>) thì lúc này tập ma trận từ chỉ mục (terms document) A_{MxN} rất lớn. Điều này dẫn đến các hạn chế sau:

- Do số từ chỉ mục M>>N nên ma trận từ chỉ mục (terms document) A sẽ bị thưa rất nhiều. Việc gom cụm trên tập ma trận quá lớn tốn nhiều chi phái và thời gian.
- Việc sử dụng thuật toán gom cụm trong hệ truy tìm thông tin khi truy vấn văn bản cũng sẽ bị hai hạn chế về ***synoymy*** và ***polysemy***[7], [8], [9]. Với synoymy, nhiều từ có thể được sử dụng để biểu diễn một khái niệm, vì vậy hệ thống không thể trả về những văn bản liên quan đến câu truy vấn của người dùng khi họ sử dụng những từ trong câu truy vấn đồng nghĩa với những từ trong văn bản. Với polysemy, một từ có thể có nhiều nghĩa, vì vậy hệ thống có thể trả về những văn bản không liên quan. Điều này thực tế rất thường xảy ra bởi vì các văn bản trong tập văn bản được viết bởi rất nhiều tác giả, với cách dùng từ rất khác nhau. Một cách tiếp cận tốt hơn cho phép người dùng truy vấn văn bản dựa trên khái niệm (*concept*) hay nghĩa (*meaning*) của văn bản.

Mô hình LSI khắc phục hai hạn chế trên và giảm số chiều của ma trận từ chỉ mục (terms document) A bằng cách chỉ mục khái niệm được tạo ra bởi phương pháp thống kê (*phân tích SVD ma trận term document A*) thay cho việc sử dụng các từ chỉ mục đơn [8]. Mô hình LSI dựa trên giả thiết là có các ngữ nghĩa tiềm ẩn (*latent semantic*) trong việc sử dụng từ: có nhiều từ biểu diễn cho một khái niệm và một

khái niệm có thể được biểu diễn bởi nhiều từ. Mô hình LSI sử dụng phân tích SVD (*Singular Value Decomposition*) ma trận term – document *A* để phát hiện ra các quan hệ ngữ nghĩa trong cách dùng từ trong toàn bộ văn bản.

Do những ưu điểm trên của mô hình LSI (đã được trình bày chi tiết trong chương 2), luận văn đề nghị dùng mô hình LSI để tiền xử lý lại ma trận từ chỉ mục (terms document) A trước, sau đó mới thực hiện gom cụm.

Thuật toán tiền xử lý ma trận từ chỉ mục (term document) A

Input:

- Term document A (term là số dòng, document là số cột)
- Hạng của ma trận

Output:

Tập các ma trận có hạng là k: A_k, U_k, $\sum_k$, V_k.

1: Chuẩn hóa chiều dài của các vector cột của A (các vector văn bản) bằng cách chia các vector cột với chiều dài $|a_j|$.

$$| a_j |= \sqrt{\sum_{i=0}^{n} | a_{i,j}^2}$$

2: Phân tích SVD trên ma trận A

$$A = U\sum V^T$$

3: Tính ma trận xấp xỉ A_k

$$A_k = U_k\sum_k V_k^T$$

3.2.4.2 Chọn độ đo khoảng cách thích hợp

Nhìn chung, thuật toán K-means và các biến thể của nó đều tính khoảng cách từ đối tượng đến trong tâm cụm. Việc tính khoảng cách này cũng tùy vào đối tượng dữ liệu mà chọn các biến thang đo khoảng cách khác nhau. Các công thức tính khoảng cách theo Euclidean hay Manhattan đều phụ thuộc vào chiều dài của đối tượng dữ

liệu, nghĩa là các đối tượng có chiều dài càng lớn thì càng xa nhau. Điều này sẽ không phù hợp với các đối tượng tài liệu văn bản. Trong văn bản, một tài liệu dài và một tài liệu ngắn có thể có độ tương tự rất lớn nếu tập chỉ mục của hai văn bản là như nhau, nên việc áp dụng khoảng cách theo Euclidean hay Manhattan thì sẽ không trả về các tài liệu liên quan với nhau một cách chính xác.

Do đó, trong luận văn này sẽ đề nghị tính độ tương tự giữa các tập văn bản trong hệ truy tìm thông tin dùng độ đo Cosines. Độ đo Cosines tính độ tương tự giữa các văn bản không dựa vào chiều dài của nó mà tính góc giữa 2 vector văn bản đó. Hai vector văn bản có độ tương tự cần cao (hay gọi là gần nhau) nếu góc tạo bởi chúng càng nhỏ.

- **Khoảng cách Euclidean**

Khoảng cách Euclidean giữa 2 điểm u và v có p tọa độ được tính bởi công thức sau:

$$dist(u,v) = \sqrt{\sum_{i=1}^{p}(u_i - v_i)^2}$$

- **Khoảng cách Manhattan**

Khoảng cách Manhattan giữa 2 điểm u và v có p tọa độ được tính bởi công thức sau:

$$D_M(u,v) = \sum_{i=1}^{p}|u_i - v_i|$$

- **Độ đo tương tự Cosines**

Công thức tính độ đo tương tự Cosines giữa 2 vector x, y có dạng:

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

Để hiểu rõ hơn về tính chất của các độ đo, ta hãy xem ví dụ sau đây:

Ví dụ 3.1: giả sử ta có 3 văn bản và 1 câu truy vấn như sau:

q = (1, 2, 0)

d1= (2, 3, 1)

d2= (20, 30, 10)

d3= (0, 1, 3)

Sử dụng các độ đo khoảng cách ở trên ta tính được khoảng cách của vector truy vấn q với 3 văn bản như sau:

Văn bản	**Khoảng cách**			**Xếp hạng**		
	D_E	D_M	Cosines(q,d)	D_E	D_M	Cosines(q,d)
d1	1.73	3	0.96	1	1	1
d2	35.28	58	0.96	3	3	2
d3	3.32	5	0.28	2	2	3

Bảng 3.1: trình bày kết quả truy vấn sử dụng 3 độ đo.

Từ ví dụ 3.1 ta thấy rằng d2 có độ tương tự với d1 cao hơn d3, từ kết quả của bảng 3.1 thì chỉ có độ đo Coinses mới trả về kết quả truy vấn chính xác.

3.2.4.3 Chọn số cụm cho thuật toán K-means

Từ những khuyết điểm của thuật toán k-means cổ điển được trình bày trong mục 3.2.3 về việc chọn trước số cụm. Trong bài luận cũng đề nghị cách chọn trước số cụm để không làm ảnh hưởng đến độ hội tụ của văn bản là sẽ lấy từ hệ số k của phân tích SVD để tạo ma trận xấp xỉ A_k.

Như đã giới thiệu ở chương 2 mục 2.2.5, việc giảm hạng của ma trận term - document A bằng phân tích SVD(A) và tạo ma trận $A_k = U_k \Sigma_k V_k^T$ có hạng k chính là chiếu ma trận A có hạng r_A vào không gian LSI k chiều nhỏ hơn. Không gian k

chiều này còn gọi là không gian "ngữ nghĩa" hay không gian "khái niệm"(*concept*). Các khái niệm này được hiểu như các khái niệm nhân tạo (*artificial concepts*) [8], và không thể biểu diễn được trong ngôn ngữ tự nhiên. Những gì có thể hiểu đó là cho văn bản *d*, văn bản *d* có độ đo "tương tự" gần với khái niệm 1, hoàn toàn "khác" với khái niệm 2 và ít gần hơn với khái niệm 3….[9]. Việc chọn hệ số *k* đóng vai trò rất quan trọng trong mô hình LSI, theo các tài liệu nghiên cứu về LSI [2] [8], qua thực nghiệm trên các tập dữ liệu văn bản cụ thể.

Vì *k* mang tính chất là *k* ngữ nghĩa trong mô hình LSI phù hợp với việc chọn *k* cụm văn bản theo ngữ nghĩa nên luận văn đề xuất cách chọn số cụm theo hệ số *k* trong mô hình LSI. Điều này sẽ được kiểm chứng lại trong chương 4 trên tập văn bản thử nghiệm 7379 văn bản.

Thuật toán K-means của luận văn

Input:

- Ma trận xấp xỉ A_k.
- Số cụm k (= hệ số k trong mô hình LSI).

Thuật toán

1. Chọn bất kỳ *k* đối tượng làm các tâm (centroids) ban đầu.
2. Gán hoặc gán lại từng đối tượng vào cụm với khoảng cách gần nhất (tính theo độ đo Cosines).
3. Cập nhật centroids.
4. Quay về bước 2, dừng khi không còn phép gán mới.

Ví dụ 3.2: trở lại ví dụ trong chương 2 ta chọn số cụm là 3. Khi đó ta tiến hành gom cụm như sau:

Khởi tạo số cụm:

Tọa độ	D1	D2	D3	D4	D5	Cluster 1	Cluster 2	Cluster 3
X1	0.4366	0.3067	0.4412	0.4909	0.5288	0.4366	0.3067	0.4412
X2	-0.4717	0.7549	-0.3568	-0.0346	0.2815	-0.4717	0.7549	-0.3568
X3	0.3688	0.0998	-0.6247	0.5711	-0.3712	0.3688	0.0998	-0.6247

Kết quả được thể hiện như sau:

						Tổng theo điều kiện "Yes"
Cluster 1	Yes	No	No	Yes	No	2
Cluster 2	No	Yes	No	No	Yes	2
Cluster 3	No	No	Yes	No	No	1

3.3 Truy vấn trong gom cụm

Sau khi đã tiến hành gom cụm , để truy vấn văn bản, đầu tiên ta sẽ đi tính độ đo Cosines của vector truy vấn q với từng vector trọng tâm của cụm c. Độ đo Cosines giữa vector truy vấn q và vector trọng tâm c được tính như sau:

$$Cos\theta = \frac{q.c}{|q|.|c|} \qquad (3.1)$$

Kế tiếp, sau khi đã tính được độ đo Cosines của vector truy vấn q và với các vector trọng tâm của cụm theo công thức (3.1), ta so sánh các độ đo này với một ngưỡng nào đó. Lúc này, để trả về các văn bản mà người dùng cần ta đi tính tiếp độ đo Cosines của các văn bản trong cụm mà thỏa điều kiện.

Ví dụ 3.3: quay lai ví dụ 3.2 sau khi đã thực hiện gom cụm xong ta sẽ thực hiện lại câu truy vấn $q^{(1)}$(***baking bread***). Kết quả được thể hiện như sau:

DISTANCE	**Q**
Cluster 1	0.918215469
Cluster 2	-0.22716859
Cluster 3	-0.353485478

Nếu chọn ngưỡng là 0.5 thì lúc này chỉ có cluster 1 thỏa điều kiện nên cluster 1 được trả về. Khi đó ta chỉ tính Cosines của vector q với các văn bản trong cluster 1. Nhìn lại ví dụ 3.2 ta thấy trong cluster 1 chỉ có 2 văn bản D1 và D4, ta tính tiếp như sau:

DISTANCE	D1	D4
Q	0.589175403322357	0.651164569635012

Với ngưỡng là **0.5** thì lúc đó văn bản D1 và D4 được trả về.

Đối với mô hình LSI, để trả về các văn bản liên quan đến câu truy vấn thì phải đi tính cosines giữa vector truy vấn với tất cả các vector của văn bản. Điều này đã làm hạn chế tốc độ truy tìm thông tin.

Nhìn lại ví dụ 3.3, việc trả về tập văn bản liên quan đến câu truy vấn được cải thiện một cách hiệu quả. Lúc này ta chỉ lấy vector truy vấn tính Cosines với các văn bản trong các cụm được trả về. Số cụm và số văn bản dùng để tính Cosines với vector truy vấn q rất ít hơn tập văn bản ban đầu.

3.4 Đánh giá hiệu quả của việc kết hợp thuật toán gom cụm với mô hình LSI

Giả sử có tập N văn bản. (N rất lớn). Khi đó ta thấy:

Đối với mô hình LSI, sau khi phân tích SVD trên ma trận từ chỉ mục (terms document) A làm giảm đi số chiều của ma trân A rất nhiều (k<<N). Tuy nhiên, để thực hiện truy vấn thì trong mô hình LSI cũng vẫn phải đi tính Cosines của vector truy vấn q với tất cả các văn bản trong tập văn bản N. Điều này làm giảm hiệu quả của mô hình LSI.

Vậy thời gian tính Cosines của câu truy vấn q với các văn bản trong tập văn bản N là: O(N).

Đối với mô hình dùng thuật toán gom cụm K-means đã cải tiến sau khi phân tích SVD, ta thấy:

- Gọi k là số cụm ban đầu. Thời gian tính Cosines của câu truy vấn q với các vector trong tâm là O(k).
- Gọi n là số văn bản trong từng cụm (n<<N). Thời gian tính Cosines của câu truy vấn q với các vector văn bản trong từng cụm thỏa ngưởng trả về là O(kn).

Vậy thời gian trả về số văn bản cần lấy là: O(k) + O(kn) = O(kn)
Như vậy, O(kn) << O(N) vì (k,n << N)

Lúc này, mô hình cải tiến chỉ đi tính Cosines của câu truy vấn q với các vector văn bản trong từng cụm thỏa ngưỡng trả về nên làm tăng hiệu quả so với mô hình LSI.

Trường hợp xấu nhất, nếu tập văn bản N thỏa hết điều kiện cần trả về thì lúc này mô hình cải tiến sẽ chậm hơn mô hình LSI k lần.

3.5 Cập nhật lại văn bản cho cụm

Để cập nhật lại văn bản cho cụm ta chiếu văn bản đó vào không gian LSI, sau đó tính Cosines của văn bản với tất cả các cụm và gán văn bản vào cụm có độ đo Cosines cao nhất (văn bản gần với cụm đó nhất).

Thuật toán cập nhật văn bản cho cụm

1. Chiếu tập văn bản vào không gian LSI:

$$\hat{d} = d^T U_k \Sigma_k^{-1}$$ ($\hat{d}$ là hình chiếu của d)

2. Tính Cos($\hat{d}_i$,c) i=1,2…n
3. Cập nhật lại văn bản d cho cụm gần nhất.
4. Quay lại bước 2 cho đến khi nào cập nhật tất cả văn bản vào cụm.

CHƯƠNG 4
CÀI ĐẶT THỬ NGHIỆM HỆ TRUY TÌM THÔNG TIN (IR)

Trong chương này trình bày các nội dung sau:

- Xây dựng hệ IR trên: mô hình không gian vector, mô hình ngữ nghĩa LSI, gom cụm trên không gian LSI bằng thuật toán K-means. Thử nghiệm trên tập dữ liệu thực được lấy từ các tài liệu điện tử dạng tập tin HTML , tập dữ liệu có 7379 văn bản, đều sử dụng ngôn ngữ tiếng Anh.
- Dựa trên dữ liệu kiểm tra kết quả trong cách chọn hệ số *k* trong bài báo [1] và của tác giả Miles Efron trình bày trong bài báo [8] được trình bày trong chương 2 mục 2.2.5 để đưa ra hệ số k tối ưu cho mô hình LSI và thuật toán K-means.
- . So sánh kết quả thử nghiệm của hệ IR trên mô hình không gian vector, mô hình LSI và thuật toán gom cụm k-means kết hợp mô hình LSI.

4.1 Xử lý văn bản tiếng Anh

Theo mô hình kiến trúc ở chương 1 thì giai đoạn đầu tiên là xử lý văn bản. Giai đoạn này phụ thuộc vào ngôn ngữ sử dụng. Luận văn sử dụng tập văn bản ngôn ngữ tiếng Anh nên cách xử lý gồm các bước sau:

- Đọc nội dung từng văn bản, loại bỏ các không có ý nghĩa (*function word)* như: the, of, a, ...
- Giữ lại các từ có nội dung (*content word*) sử dụng như các từ chỉ mục,ngoài ra trước khi sử dụng nó như các từ chỉ mục cho tập văn bản ta sử dụng thuật toán *stemming* để giữ lại từ gốc của từ đó. Ví dụ: trong văn bản có các từ *computer, computing*, *computed* sử dụng thuật toán *stemming* để lấy từ gốc *comput*. Sử dụng thuật toán *stemming* làm giảm số từ chỉ mục hay giảm số chiều cho vector biểu diễn văn bản nhưng vẫn giữ được nội dung của văn bản.
- Thao tác tiếp theo làm giảm số từ chỉ mục bằng cách loại bỏ các từ chỉ mục chỉ xuất hiện 10 lần trong toàn bộ văn bản.

Tập các từ còn lại được sử dụng như các từ chỉ mục cho toàn bộ văn bản, số từ chỉ mục này chính là số chiều cho mỗi vector biểu diễn văn bản. Tập từ chỉ mục này được hệ thống lưu lại thành tập tin: *Term_ Index.out* trong thư mục chứa dữ liệu chỉ mục của hệ thống.

4.2 Cấu hình hệ thống

Tập văn bản 7379 được thử nghiệm trên hệ thông có cấu hình như sau:

- Pentium Core 2 Duol.
- CPU 3 GHz.
- Ram 4 GHz.
- HDD 160 Gb.

Kích thước của các tập tin trong hệ thống IR:

STT	Tên tập tin	Kích thước
1	Term_index.out	104 kb
2	Doc_index.out	377 kb
3	Term_Doc_Matrix.out	1,18 gb
4	Matrix_Uk.out	78,859 kb
5	Matrix_Sk.out	2,572 kb
6	Matrix_Vk.out	42,157 kb
7	Matrix_C.out	879 kb
8	Tập 7379 docs	33,4 mb

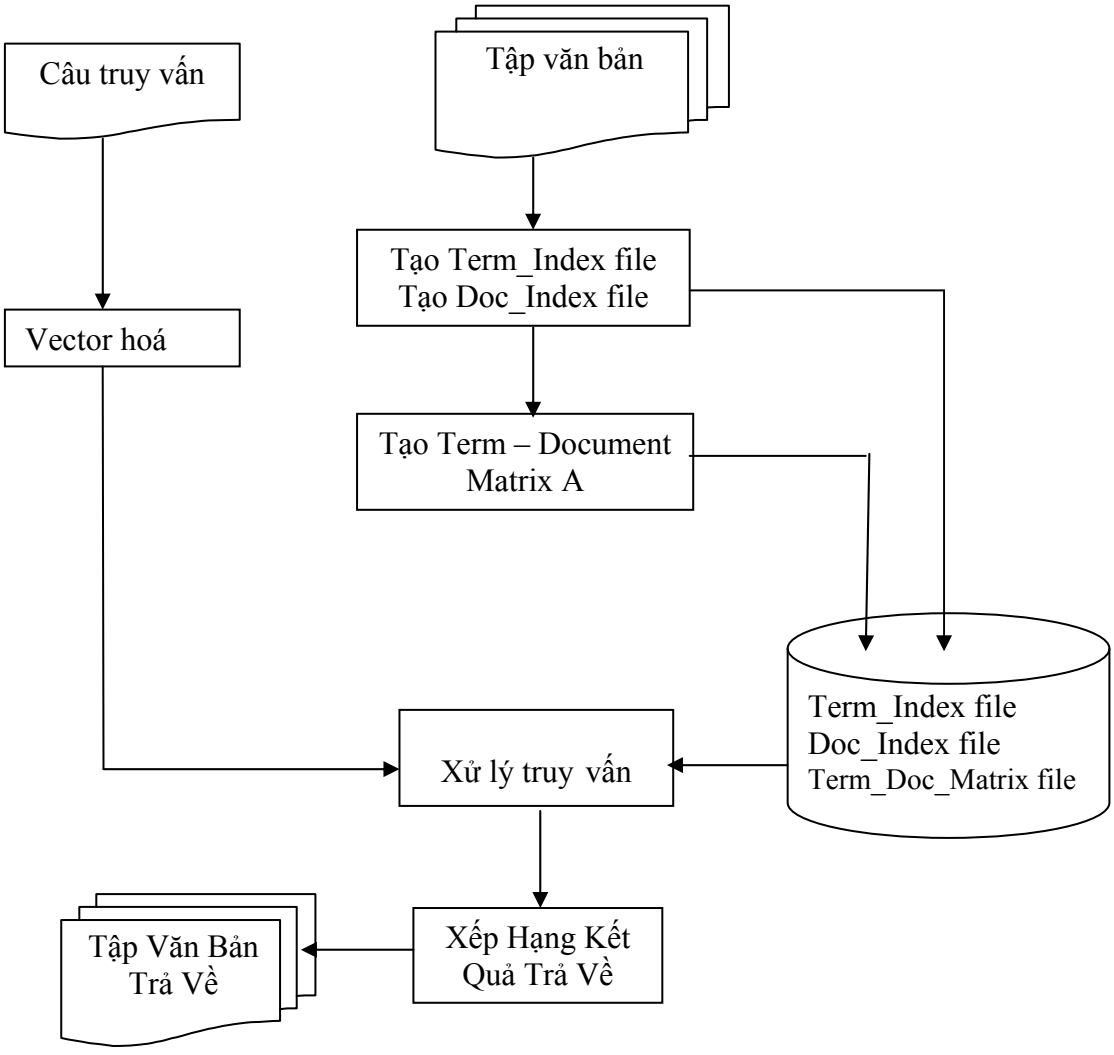
Bảng 4.1: liệt kê các tập tin trong hệ thống IR

4.3 Vector hoá văn bản và tạo ma trận từ chỉ mục (Term – Document) A biểu diễn tập văn bản

Sau khi đã tạo tập từ chỉ mục (*giả sử có m từ chỉ mục*) cho tập văn bản, mỗi văn bản được vector hoá thành một vector *m* chiều. Các thành phần của mỗi vector được

đánh trọng số sử dụng hàm: $w_{ij} = f_{ij} \times \log \frac{N}{n_i}$ như đã nêu trong chương 2. Vậy một tập có *n* văn bản được biểu diễn như một ma trận term – document *A* cấp *m* x *n*. Tập văn bản được biểu diễn như một ma trận trong đó mỗi cột là vector của một văn bản.

4.4 Hệ VSM_IR (*Vector Space Model Information Retrieval System*)



Hình 4.1 Kiến trúc VSM_IR

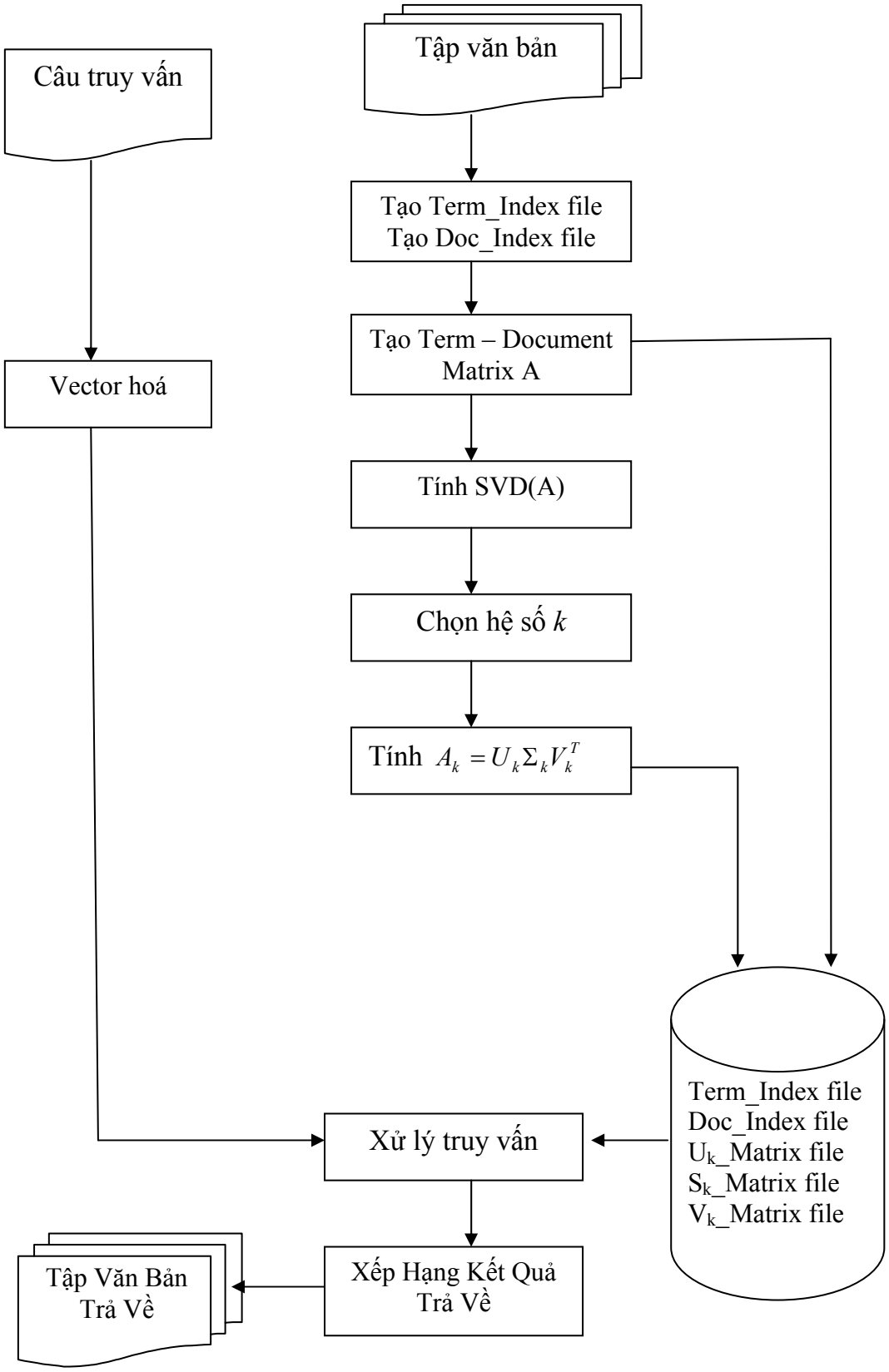
Hình 4.1 là kiến trúc của một hệ VSM_IR gồm các module:

- Xử lý văn bản và tạo các tập tin *Term_ Index.out* và *Doc_ Index.out*
- Tạo ma trận Term – Document *A*
- Xử lý truy vấn
- Xếp hạng kết quả trả về theo thứ tự giảm dần độ đo *cosines*
- Giao diện hiển thị kết quả truy vấn.

Các tập tin chính trong thư mục chứa dữ liệu chỉ mục:

- *Term_ Index.out* lưu tập từ chỉ mục cho tập văn bản
- *Doc_ Index.out* lưu tên tập tin và đường dẫn đến tập văn bản
- *Term_Doc_Marix.out* lưu ma trận Term – Document A biểu diễn tập văn bản.

4.5 Hệ LSI_IR (*Latent Semantic Indexing Information Retrieval System*)



Hình 4.2 Kiến trúc hệ LSI_IR

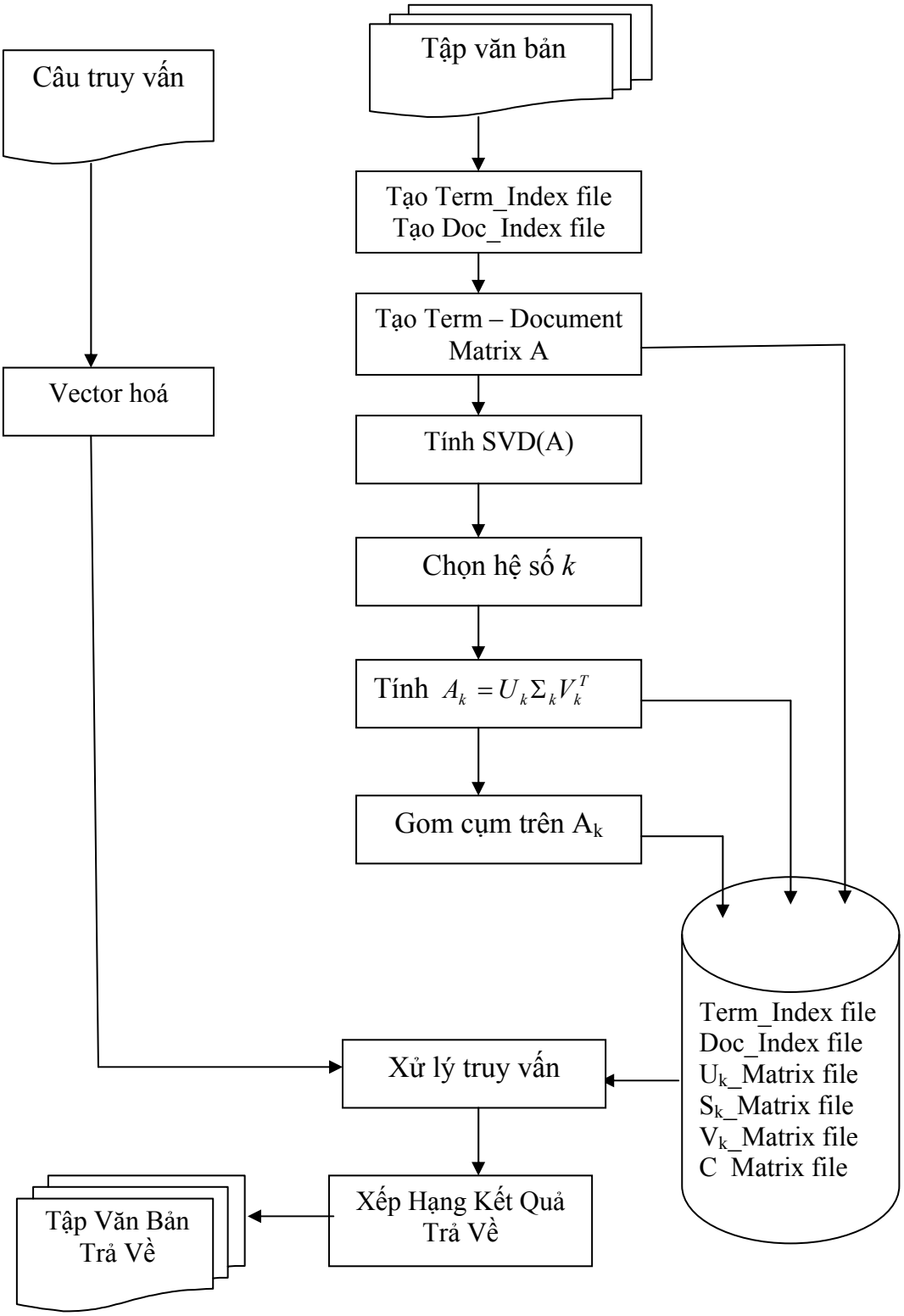
Hình 4.2 là kiến trúc của hệ LSI_IR ngoài các module có trong hệ VSM_IR còn có thêm các module khác, dưới đây là toàn bộ các module:

- Xử lý văn bản tạo các tập tin *Index_Term.out* và *Doc_ Index.out*
- Tạo ma trận Term – Document *A*
- Tính SVD ma trận Term – Document $A = U\Sigma V^T$
- Chọn hệ số k tạo ma trận $A_k = U_k \Sigma_k V_k^T$ với ngưỡng sai số tương đối so với ma trận *A* cho trước.
- Xử lý truy vấn
- Xếp hạng kết quả trả về theo thứ tự giảm dần độ đo *cosines*
- Giao diện hiển thị kết quả truy vấn

Các tập tin trong thư mục chứa dữ liệu chỉ mục của hệ LSI_IR:

- *Term_ Index.out* lưu tập từ chỉ mục cho tập văn bản
- *Doc_ Index.out* lưu tên tập tin và đường dẫn đến tập văn bản
- Ba file *Matrix_Uk.out, Matrix_Sk.out, Matrix_Vk.out* lưu ba ma trận U_k, Σ_k, V_k^T

4.6 Hệ IR kết hợp mô hình LSI và thuật tóan gom cụm K-means



Hình 4.3 Kiến trúc hệ IR cải tiến áp dụng mô hình LSI và thuật toán K-means

Hình 4.3 là kiến trúc của hệ IR này ngoài các module có trong hệ VSM_IR và LSI_IR còn có thêm các module khác, dưới đây là toàn bộ các module:

- Xử lý văn bản tạo các tập tin *Term_ Index.out* và *Doc_ Index.out*
- Tạo ma trận Term – Document *A*
- Tính SVD ma trận Term – Document $A = U\Sigma V^T$
- Chọn hệ số *k* tạo ma trận $A_k = U_k \Sigma_k V_k^T$ với ngưỡng sai số tương đối so với ma trận *A* cho trước.
- Chọn hệ số k cho thuật toán K-means và tập đối tượng dùng để gom cụm là ma trận A_k.
- Xử lý truy vấn.
- Xếp hạng kết quả trả về theo thứ tự giảm dần độ đo *cosines.*
- Giao diện hiển thị kết quả truy vấn.

Các tập tin trong thư mục chứa dữ liệu chỉ mục của hệ LSI_IR:

- *Term_ Index.out* lưu tập từ chỉ mục cho tập văn bản
- *Doc_ Index.out* lưu tên tập tin và đường dẫn đến tập văn bản
- Ba file *Matrix_Uk.out, Matrix_Sk.out, Matrix_Vk.out* lưu ba ma trận U_k, Σ_k, V_k^T
- *MaTrix_C.out dùng để lưu tên cụm, trong tâm của cụm và số văn bản của cụm.*

4.7 Kết quả thử nghiệm

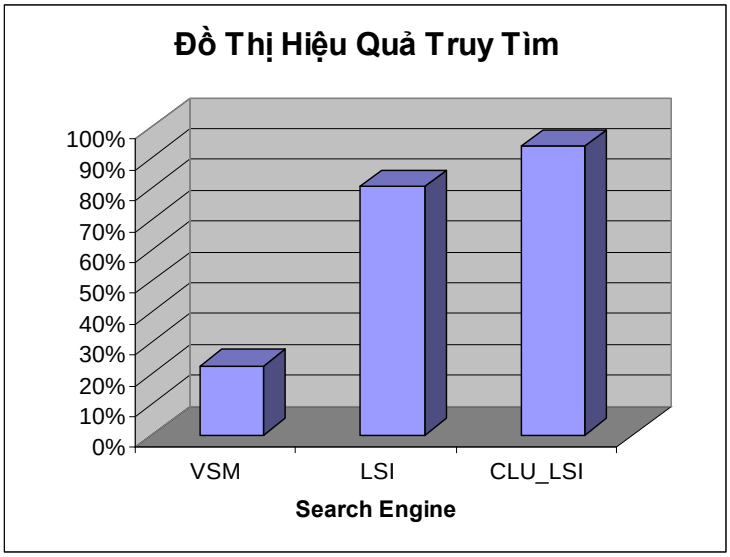
Như đã trình bày trong chương 1, hiệu quả của một hệ IR cơ bản được đánh giá dựa trên 3 tiêu chuẩn:

- Khả năng truy tìm của hệ thống, thông thường được đánh giá bởi trung bình độ đo *precision*
- Hiệu quả lưu trữ dữ liệu chỉ mục của hệ thống
- Thời gian thực hiện thủ tục truy vấn.

4.7.1 So sánh và đánh giá hiệu quả của hệ VSM_ IR , hệ LSI_IR và hệ cải tiến dùng mô hình LSI kết hợp thuật toán K-means trên tập dữ liệu thực

Sau khi cài đặt hai hệ IR trên mô hình VSM, LSI và mô hình cải tiến gom cụm trên LSI , chạy thử trên tập dữ liệu thực gồm 7379 văn bản về Công Nghệ Thông Tin và Toán Học, sử dụng 13274 từ chỉ mục cho tập văn bản trên, kết quả thống kê cho thấy rằng hệ IR sử dụng LSI hiệu năng truy tìm cao hơn khoảng từ 30 – 40 % so với mô hình VSM (hình 4.7) và hệ IR cải tiến bằng cách gom cụm trên không gian LSI hiệu năng truy tìm khoảng 5 – 10% so với LSI (hình 4.4).

Hình 4.4 Dưới đây là đồ thị về hiệu quả truy tìm của các hệ IR, được thử nghiệm trên tập dữ liệu thử nghiệm 7379 văn bản và 13274 từ với 30 câu truy vấn. Chọn hệ số $k = 300$ cho mô hình LSI. Tập dữ liệu của 30 câu truy vấn được trình bày trong trong phần phụ lục A bảng PLA.3.



Hình 4.4 Đồ thị hiệu quả truy tìm của hệ VSM_IR, LSI_IR và CLU_LSI_IR trên tập 7379 văn bản

Đối với hệ LSI_IR, ngưỡng sai số được chọn là 10% và có hệ số *k* tương ứng 300 trên tập văn bản thử nghiệm, là hệ số tốt hơn so với các hệ số khác sau khi đã chạy thử với các tỉ lệ sai số tương ứng trong bảng 4.1.

Sai số tương đối	*16%*	*13%*	*12%*	*11.5%*	*11%*	*10%*	*9.5%*	*8%*	*6.5%*
K	50	100	150	200	250	300	350	400	450

Bảng 4.2 Hệ số k tương ứng với các sai số tương đối của tập 7379 văn bản

Hệ số k này cũng được chọn để khởi tạo số cụm cho thuật toán K-means. Trên thực tế việc sử dụng hai độ đo *precision* và *recall* để đánh giá hiệu quả của hệ thống bất kỳ là rất khó, vì thực tế không thể xác định được số văn bản liên quan đến câu truy vấn cụ thể trong tập văn lớn là bao nhiêu, chỉ có thể thực hiện điều này trên tập văn bản nhỏ, được chọn lựa và phân loại chi tiết. Một khó khăn nữa gặp phải là trong việc đánh giá kết quả trả về của tập văn bản liên quan đến câu truy vấn phụ thuộc rất nhiều vào tính chủ quan của người đánh giá.

Với những phân tích trên việc đánh giá và so sánh hiệu quả thực thi của hai

hệ IR trên là rất khó nếu sử dụng hai độ đo *preision* và *recall.* Luận văn chỉ đánh giá và so sánh hiệu quả của hệ IR bằng cách so sánh tổng số văn bản liên quan được trả về của hai hệ VSM_IR, LSI_IR, và hệ cải tiến CLU_LSI_IR khi thử nghiệm trên cùng một tập câu truy vấn.

Một tập văn bản trả về của một truy vấn cụ thể gồm có hai phần: Phần liên quan đến câu truy vấn – ký hiệu là R và phần không liên quan đến câu truy vấn – ký hiệu NR. Vậy hiệu quả truy tìm (*HQTT*) của hệ thống với một câu truy vấn cụ thể được tính:

$$HQTT = \frac{R}{R + NR}$$

Công thức trên chính là tỉ lệ số văn bản liên quan đến câu truy vấn trên tổng số văn bản trả về, công thức trên cũng chính là độ đo *precision*.

Giả sử M là số văn bản trả về của hệ VSM_IR của cùng câu truy vấn với hệ LSI_IR, vậy $R_{VSM} = M - NR_{VSM}$, từ đây ta có thể tính được HQTT của VSM_IR so với LSI_IR

$$HQTT(VSM) = \frac{R_{SVM}}{R_{LSI}}.$$

Giả sử N là số văn bản trả về của một câu truy vấn của hệ LSI_IR.

Vậy $R_{LSI} = N - NR_{LSI}$ và *hiệu quả truy tìm của hệ LSI_IR*.

$$HQTT(LSI) = \frac{R_{LSI}}{N}$$

Giả sử P là số văn bản trả về của một câu truy vấn của hệ IR cải tiến bằng cách gom cụm sau khi đã phân tích SVD (CLU_LSI_IR).

Vậy $R_{CLU\_LSI} = P - NR_{CLU\_LSI}$ và *hiệu quả truy tìm của hệ CLU_LS _IR*.

$$HQTT(CLU\_LSI) = \frac{R_{CLU\_LSI}}{P}$$

Ví dụ: chọn ngưỡng 0.2 và hệ số k = 300 cho LSI_IR, kết quả truy vấn của một câu truy vấn cụ thể trên VSM_IR M = 24 văn bản, LSI_IR có N = 46 văn bản và CLU_LSI _IR có P = 46 văn bản.

Với VSM_IR có R_{VSM} = 24 văn bản liên quan đến câu truy vấn và không có văn bản nào không liên quan

$$HQTT(SVM) = \frac{24}{40} \times 100\% = 60\% .$$

Với LSI_IR có R_{LSI} = 40 tức có 6 văn bản không liên quan đến câu truy vấn. Từ đây ta có:

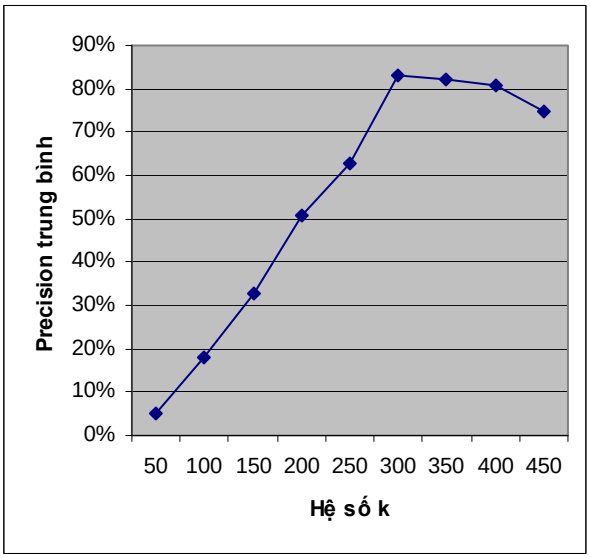
$$HQTT(LSI) = \frac{40}{46} \times 100\% = 87\%$$

Với LSI_CLU_LSI_IR có R_{CLU_LSI} = 40 tức có 5 văn bản không liên quan đến câu truy vấn. Từ đây ta có:

$$HQTT(LSI\_CLU) = \frac{40}{45} \times 100\% = 89\%$$

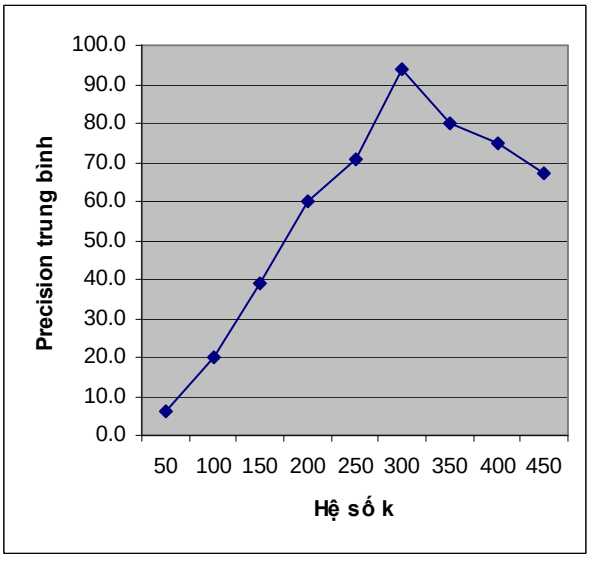
Vậy tuy kết quả truy vấn của VSM_IR không có văn bản nào không liên quan đến câu truy vấn nhưng hiệu quả chỉ đạt 60% so với 87% của LSI_IR và CLU_ LSI _IR là 89% so với LSI.

Các đồ thị sau đây biểu diễn trên tập dữ liệu thử nghiệm 7379 văn bản và 13274 từ.



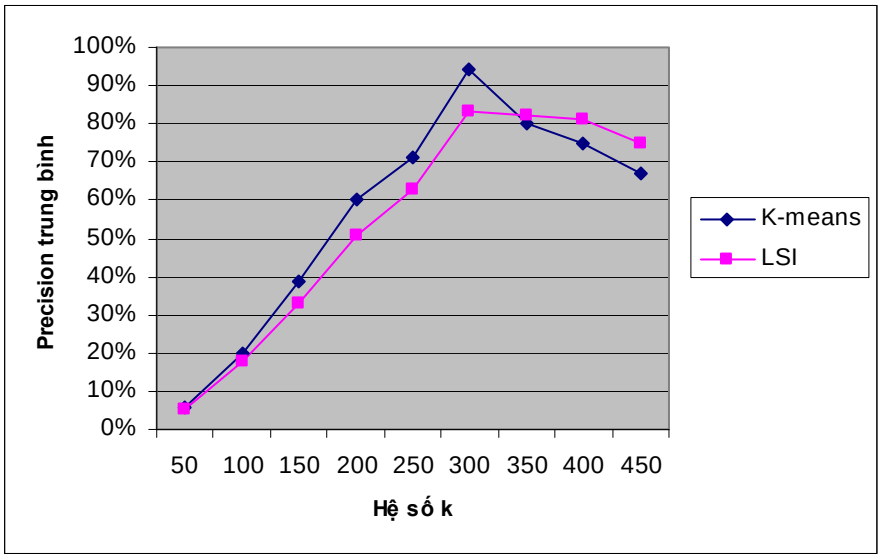
Hình 4.5 Đồ thị độ đo precision trung bình của LSI

Số liệu của tập dữ liệu thử nghiệm 30 câu truy vấn của đồ thị hình 4.5 được trình bày trong phụ lục A bảng PLA.4.



Hình 4.6 Đồ thị độ đo precision trung bình sau khi gom cụm

Số liệu của tập dữ liệu thử nghiệm 30 câu truy vấn của đồ thị hình 4.6 được trình bày trong phụ lục A bảng PLA.4.



Hình 4.7 Đồ thị độ đo precision trung bình của LSI và K-means

Số liệu của tập dữ liệu thử nghiệm 30 câu truy vấn của đồ thị hình 4.7 được trình bày trong phụ lục A bảng PLA.5.

4.7.2 Dung lượng lưu trữ dữ liệu chỉ mục của ba hệ thống trên tập dữ liệu thực 7379 văn bản

Dung lượng bộ nhớ RAM cho mỗi hệ IR lưu trữ dữ liệu chỉ mục khi thực thi được đo bởi ma trận biểu diễn tập văn bản và các tập tin *Term_Index.out* và *Doc_Index.out*.

Công thức tính dung lượng bộ nhớ cho ma trận biểu diễn tập văn bản như sau:

RAM = (<*số văn bản*> x <*số từ chỉ mục*>) x (sizeof(<*kiểu dữ liệu*>))

- Với hệ VSM_IR, ma trận Term – Document *A* cấp 7379 x 13274 mỗi phần tử ma trận có kiểu *float* trong Java chiếm 4 byte.

Vậy RAM = (7379 x 13274) x 4(byte) = 374MB

- Với LSI_IR lưu ba ma trận $U_{13274x300}$, $\Sigma_{300x300}$, và $V^T_{300\times7379}$.

Vậy RAM =(13274 x 300 + 300 x 300 + 300 x 7379) x 4(byte) = 24 MB

- Với CLU_ LSI _IR lưu ba ma trận $U_{13274x300}$, $\Sigma_{300x300}$, và $V^T_{300\times7379}$, $C_{300x300.}$

Vậy RAM = (13274 x 300 + 300 x 300 + 300 x 7379+ 300 x 300) x 4(byte)

= 24,32 MB

Hệ IR	**Dung Lượng Lưu Trữ Trên RAM**
VSM	374 MB
LSI	24 MB
CLU_LSI	24,32 MB

Bảng *4.3 Dung lượng bộ nhớ lưu trữ dữ liệu chỉ mục của tập dữ liệu thực 7379 văn bản*

Với kết quả như trên bảng 4.2 ta nhận xét rằng dung lượng lưu trữ dữ liệu chỉ mục của mô hình LSI giảm hơn 90% so với VSM và mô hình cải tiến gom cụm kết hợp mô hình LSI tăng 10% so với LSI vì phải lưu thêm ma trận trọng tâm C của cụm.

Điều này cho ta thấy mô hình cải tiến gom cụm sau khi phân tích SVD về dung lượng lưu trữ dữ liệu không tốt hơn nhiều so với mô hình LSI và cả hai mô hình đều tốt hơn rất nhiều so với mô hình VSM.

4.7.3 Thời gian thực thi thủ tục truy vấn của ba hệ thống trên tập dữ liệu thực 7379 văn bản

Trong mô hình LSI, lượng dữ liệu lưu trữ chỉ mục ít hơn nhiều so với VSM nên việc tính toán cho thủ tục truy vấn nhanh hơn rất nhiều. Nhưng trong mô hình LSI việc tính toán còn bị hạn chế là phải tính lại Cosines của câu truy vấn với tất cả các văn bản nên thời gian tính toán chậm hơn so với mô hình cải tiến gom cụm văn bản sau khi đã phân tích SVD. Với việc thử nghiệm trên cùng một tập câu truy vấn cho cả ba hệ IR, thời gian cho thủ tục tìm kiếm trên LSI_IR nhanh hơn trên dưới 30 lần so với VSM_IR và LSI_IR chậm hơn 2 lần so với mô hình cải tiến CLU _LSI.

Bảng 4.2 trình bày thời gian thực thi của 3 mô hình khi thực hiện câu truy vấn "sql"

Hệ IR	**Thời Gian Tìm Kiếm(giây)**
VSM	13.344
LSI	0.407
CLU_LSI	0.391

Bảng 4.4 Thời gian thực thi trung bình của thủ tục truy vấn

4.7.4 So sánh kết quả thử nghiệm giữa các độ đo khoảng cách Cosines và Euclidean của thuật toán gom cụm K-means.

Để so sánh hiệu quả của việc dùng độ đo khoảng cách Cosines và Euclidean của thuật toán gom cụm K-means ta chỉ so sánh các tập văn bản trả về của 2 độ đo.

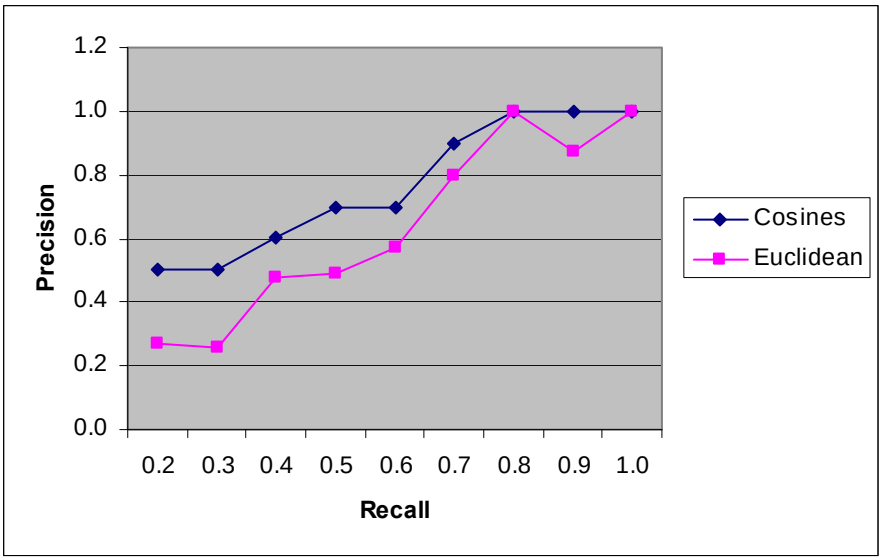
Theo như trình bày ở trên, với CLU_LSI _IR_{Cos} , R_{CLU_LSI}= 40 có 5 văn bản không liên quan đến câu truy vấn. Từ đây ta có:

$$HQTT(CLU\_LSI\_\mathrm{IR}_{\mathrm{Cos}}) = \frac{40}{45} \times 100\% = 89\%$$

Với LSI_CLU_LSI_IR_E có R_{CLU_LSI} = 32 văn bản liên quan đến câu truy vấn và không có văn bản nào không liên quan

$$HQTT(LSI\_CLU\_\mathrm{IR}_{\mathrm{E}}) = \frac{32}{40} \times 100\% = 80\%$$

Hình 4.8 Dưới đây là đồ thị so sánh 2 độ đo Cosines và Euclidean trong hệ IR, được thử nghiệm trên tập dữ liệu thử nghiệm 7379 văn bản và 13274 từ Chọn hệ số k = 300 cho mô hình LSI.



Hình 4.8 Đồ thị precision/recall của hai độ đo Euclidean và Cosines

Số liệu của tập dữ liệu thử nghiệm 30 câu truy vấn của đồ thị hình 4.8 được trình bày trong phụ lục A bảng PLA.6

4.8 Chương trình cài đặt của hệ VSM_IR , LSI_IR, CLU_LSI

Ba hệ IR trên được cài đặt trên ngôn ngữ Java, sử dụng phiên bản JDK1.6, gồm các Module:

- Module tiền xử lý văn bản thực hiện các công việc: đọc và chuyển các tập tin văn bản dạng HTML sang dạng tập tin văn bản .txt, lọc ra các token trong tập tin văn bản, loại bỏ các stopwords sau đó stemming các token, loại bỏ các token xuất hiện ít hơn 10 lần trong toàn bộ tập văn bản, các token còn lại sẽ sử dụng như tập các từ chỉ mục (*index terms*) cho tập văn bản. Tập từ chỉ mục được lưu vào các tập tin Term_Index.out và Doc_Index.out lưu tên và đường dẫn các tập tin văn bản trong tập văn bản.
- Module tạo ma trận Term_Document *A*: sau khi tập từ chỉ mục đã tạo, mỗi văn bản được vector hoá thành một vector và tập văn bản sẽ được biểu diễn thành một ma trận. Mỗi cột của ma trận biểu diễn vector của mỗi văn bản.

- Module đại số ma trận: gồm các lớp tạo và tính toán trên ma trận, các lớp phân tích SVD và QR của ma trận, tính độ đo *cosines* của hai vector.
- Module hệ VSM_IR: thực hiện công việc tìm kiếm của hệ thống dựa trên mô hình không gian vector khi người dùng thục hiện truy tìm thông tin. Xếp hạng kết quả truy tìm của hệ thống theo thứ tự giảm dần độ đo tương tự của văn bản với câu truy vấn.
- Module LSI_IR: thực hiện việc tìm kiếm thông tin trên mô hình LSI, và các thao tác cập nhật trên mô hình này. Xếp hạng kết quả tìm được của hệ thống theo thứ tự giảm dần độ đo tương tự của các văn bản với câu truy vấn.
- Module CLU_LSI _IR: thực hiện việc tìm kiếm thông tin trên mô hình gom cụm văn bản sau khi phân tích SVD, và các thao tác cập nhật trên mô hình này. Xếp hạng kết quả tìm được của hệ thống theo thứ tự giảm dần độ đo tương tự của các văn bản với câu truy vấn.
- Module giao diện và hiển thị kết quả tìm kiếm: hiển thị giao diện người dùng và kết quả tìm kiếm.

Các lớp trong module tiền xử lý văn bản:

- ✓ *Doument.java,*
- ✓ *DocumentReference.java,*
- ✓ *FileDocument.java,*
- ✓ *HTMLFileDocument.java,*
- ✓ *DocumentIterator.java,*
- ✓ *TextFileDocument.java*
- ✓ *Porter.java*

Lớp trong module tạo ma trận Term_Document A:

- ✓ *Create_Term_Doc_Matrix.java*

Các lớp trong module đại số ma trận:

- ✓ *Matrix.java*
- ✓ *SingularValueDecomposition.java*

- ✓ *QRDecomposition.java*
- ✓ *Maths.java*

Lớp trong module VSM_IR:

- ✓ *Search_VSM.java*

Các lớp trong modul LSI_IR:

- ✓ *LSI.java*
- ✓ *Search_LSI.java*

Các lớp trong modul CLU_LSI _IR:

- ✓ *Cluster.java*
- ✓ *Centroid.java*
- ✓ *Datapoint.java*
- ✓ *Search_Clustering.java*

Các lớp trong module giao diện và hiện thị kết quả tìm kiếm của ba hệ VSM_IR, LSI_IR, CLU_LSI:

- ✓ *DataTableModel.java*
- ✓ *SearchResult.java*
- ✓ *SearchGUI.java*

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Luận văn đã tập trung nghiên cứu mô hình ngữ nghĩa Latent Semantic Indexing (LSI) và ứng dụng phát triển thuật toán gom cụm văn bản theo hai cách : Tiền xử lý ma trận từ chỉ mục ban đầu và đề nghị áp dụng độ đo hợp lý để tính độ tương tự giữa các văn bản bằng độ đo Cosines. Từ những nghiên cứu về lý thuyết này đã đưa ra được kiến trúc cơ bản của một hệ IR dựa trên mô hình không gian vector. , xây dựng và thử nghiệm ba hệ IR trên ba mô hình: mô hình không gian vector, mô hình ngữ ngĩa LSI và mô hình kế hợp thuật toán K-means và mô hình ngữ nghĩa LSI.

Đánh giá hiệu quả thực thi của ba mô hình về các tiêu chí hiệu quả truy tìm, thời gian và dung lượng bộ nhớ cần thiết lưu trữ dữ liệu số hoá cho mỗi mô hình. Từ đó, thấy được hiệu quả của mô hình kết hợp thuật toán K-means và mô hình ngữ nghĩa LSI cao hơn so với mô hình không gian vector và mô hình ngữ nghĩa LSI.

Từ kết quả này, hỗ trợ cho việc xây dựng các hệ IR thực tế có hiệu quả truy tìm (*HQTT*) cao, phục vụ trong các lĩnh vực giáo dục như các hệ đào tạo từ xa dựa trên mạng Internet, các hệ thống E_Learning và cả trong các lĩnh vực thương mại, công nghiệp.

Luận văn đã chọn hệ số *k* trong mô hình LSI sử dụng định lý sai số xấp xỉ ma trận có hạng thấp dựa trên phân tích SVD ma trận từ chỉ mục (terms – document) để chọn hệ số *k* sao cho hệ thống hoạt động hiệu quả tốt nhất có thể. Và luận văn cũng đã sử dụng lại hệ số *k* này trong thuật toán gom cụm k-means để chọn ra *k* cụm đã cải thiện được việc chọn số cụm ban đầu cho thuật toán k-means.

Trong một thời gian không nhiều, những kết quả còn giới hạn, chưa sử dụng được một số bài toán gom cụm khác để so sánh kết quả truy tìm. Tuy nhiên luận

văn cũng đã đạt được những yêu cầu đề ra. Những kết quả đạt được làm cơ sở lý thuyết và thực nghiệm cho việc xây dựng các hệ IR thực tế hoạt động hiệu quả về sau.

Hướng phát triển của luận văn:

Luận văn đã đạt được một số kết quả nhất định, nhưng cũng còn một số vấn đề chưa đạt được và cũng là hướng phát triển trong tương lai.

Đối với mô hình LSI hiệu quả truy tìm của hệ thống cũng như hiệu quả về dung lượng lưu trữ và thời gian tìm kiếm phụ thuộc vào việc chọn hệ số *k*. Bài toán này hiện nay vẫn đang là bài toán mở chưa có lời giải tổng quát, chỉ giải quyết bằng thực nghiệm trên tập dữ liệu cụ thể. Hướng phát triển tương lai là sử dụng các công cụ toán học về tối ưu hoá để giải quyết bài toán chọn hệ số *k* sao cho hệ thống hoạt động tối ưu trong mô hình LSI này.

Đối với mô hình cải tiến bằng cách sử dụng thuật toán gom cụm K-means kết hợp với mô hình LSI thì hiệu quả truy tìm nhanh hơn mô hình LSI nhưng về mặt lưu trữ lại kém hơn LSI do phải lưu thêm ma trận trọng tâm của các cụm. Cách chọn số sụm trong thuật toán k-means cũng phụ thuộc vào hệ số k trong mô hình LSI. Do sự liên quan về ngữ nghĩa của câu truy vấn và tập văn bản trả về nên chưa đưa ra được sự thống kê hiệu quả giữa các mô hình một cách tự động mà chỉ dựa vào nội dung của tập văn bản trả về.

Trong luận văn chỉ tập trung truy vấn trên tài liệu bằng tiếng anh và phân tích trên từ của văn bản. Hướng phát triển tiếp theo của luận văn là làm cách nào truy vấn trên văn bản tiếng việt, từ tượng hình, truy vấn dựa trên mẫu câu. Và đây là một hướng phát triển trong tương lai.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Đỗ Trung Hiếu (2005), *Số hóa văn bản theo mô hình không gian vector và ứng dụng,* luận văn thạc sĩ, Trường Đại Học Khoa Học Tự Nhiên.

Tiếng Anh

[2] April Kontostathis (2007), "Essential Dimensions of latent sematic indexing", Department of Mathematics and Computer Science Ursinus College, Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[3] Cherukuri Aswani Kumar, Suripeddi Srinivas (2006) , "Latent Semantic Indexing Using Eigenvalue Analysis for Efficient Information Retrieval", Int. J. Appl. Math. Comput. Sci., 2006, Vol. 16, No. 4, pp. 551–558

[4] David A.Hull (1994), *Information retrieval Using Statistical Classification*, Doctor of Philosophy Degree, The University of Stanford.

[5] Gabriel Oksa, Martin Becka and Marian Vajtersic (2002)," Parallel SVD Computation in Updating Problems of Latent Semantic Indexing", *Proceeding ALGORITMY 2002 Conference on Scientific Computing*, pp. 113 – 120.

[6] Katarina Blom, (1999), *Information Retrieval Using the Singular Value Decomposition and Krylov Subspace*, Department of Mathematics Chalmers University of Technology S-412 Goteborg, Sewden

[7] Kevin Erich Heinrich (2007), *Automated Gene Classification using Nonnegative Matrix Factorization on Biomedical Literature*, Doctor of Philosophy Degree, The University of Tennessee, Knoxville.

[8] Miles Efron (2003). Eigenvalue – Based Estimators for Optimal Dimentionality Reduction in Information Retrieval. *ProQuest Information and Learning Company.*

[9] Michael W. Berry, Zlatko Drmac, Elizabeth R. Jessup (1999), "Matrix, Vector Space, and Information Retrieval", *SIAM REVIEW* Vol 41, No. 2, pp. 335 – 352.

[10] Nordianah Ab Samat, Masrah Azrifah Azmi Murad, Muhamad Taufik Abdullah, Rodziah Atan (2008), "Term Weighting Schemes Experiment Based on SVD for Malay Text Retrieval", Faculty of Computer Science and Information Technology University Putra Malaysia, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.10, October 2008.