

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**LÊ VĂN HÀO**

**NGHIÊN CỨU XÂY DỰNG HỆ THỐNG  
TÌM KIẾM VIDEO DỰA TRÊN NỘI DUNG**

**LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN**

**Hà Nội - 2016**

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**LÊ VĂN HÀO**

**NGHIÊN CỨU XÂY DỰNG HỆ THỐNG  
TÌM KIẾM VIDEO DỰA TRÊN NỘI DUNG**

Ngành: Công nghệ thông tin

Chuyên ngành: Hệ thống thông tin

Mã số: 60.48.01.04

**LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN**

**NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS – Nguyễn Trí Thành**

**Hà Nội - 2016**

## **LỜI CAM ĐOAN**

Tôi xin cam đoan kết quả đạt được trong Luận văn là sản phẩm của riêng cá nhân tôi, không sao chép lại của người khác. Những điều được trình bày trong nội dung Luận văn, hoặc là của cá nhân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn đúng quy cách. Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

*Hà Nội, 05/2016*

**Lê Văn Hào**

## MỤC LỤC

LỜI CAM ĐOAN.....	1
MỤC LỤC.....	2
BẢNG CHỮ CÁI VIẾT TẮT .....	4
DANH MỤC CÁC BẢNG BIỂU .....	5
DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ.....	6
MỞ ĐẦU .....	8
CHƯƠNG 1: GIỚI THIỆU .....	10
1.1. Giới thiệu về công cụ tìm kiếm .....	10
1.2. Lịch sử phát triển của công cụ tìm kiếm .....	10
1.3. Kiến trúc của công cụ tìm kiếm.....	11
1.3.1. Quá trình đánh chỉ mục.....	11
1.3.2. Quá trình truy vấn.....	13
1.4. Công cụ tìm kiếm video trên mạng internet.....	13
1.5. Tổng quan của đề tài và các vấn đề cần giải quyết .....	14
1.5.1. Tổng quan đề tài .....	14
1.5.2. Các vấn đề cần giải quyết.....	14
1.6. Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu.....	14
1.6.1. Ý nghĩa khoa học .....	14
1.6.2. Ý nghĩa thực tiễn.....	15
1.7. Kết luận.....	15
CHƯƠNG 2: BÀI TOÁN TÌM KIẾM VIDEO BÀI GIẢNG .....	16
DỰA TRÊN NỘI DUNG.....	16
2.1. Phát biểu bài toán .....	16
2.2. Các nghiên cứu về tìm kiếm video dựa trên nội dung.....	17
2.3. Hướng nghiên cứu của tác giả.....	18
2.4. Bài toán phân đoạn video thành ảnh .....	19
2.4.1. Khái niệm.....	19
2.4.2. Phương pháp tiếp cận.....	19
2.5. Bài toán trích xuất văn bản.....	20
2.5.1. Bài toán nhận dạng kí tự quang học .....	20
2.5.2. Bài toán xử lý trùng lặp văn bản.....	22
2.5.3. Bài toán sửa lỗi chính tả văn bản.....	26
2.6. Bài toán đánh chỉ mục và tìm kiếm.....	29
2.6.1. Khái niệm.....	29
2.6.2. Phương pháp tiếp cận.....	29
2.6.3. Kiến trúc của Elasticsearch.....	30

2.7. Kết luận.....	32
<b>CHƯƠNG 3: KỸ THUẬT ĐỀ GIẢI QUYẾT CÁC BÀI TOÁN TRONG KHUÔN KHỔ LUẬN VĂN .....</b>	<b>33</b>
3.1. Bài toán phân đoạn video thành định dạng ảnh .....	33
3.1.1. Phát biểu bài toán.....	33
3.1.2. Giải pháp thực hiện.....	33
3.2. Bài toán trích xuất văn bản.....	34
3.2.1. Bài toán nhận dạng kí tự quang học bằng công cụ Tesseract-OCR..	34
3.2.2. Bài toán xử lý trùng lặp văn bản bằng kĩ thuật Shingling .....	37
3.2.3. Bài toán sửa lỗi chính tả văn bản tiếng Việt.....	40
3.3. Bài toán đánh chỉ mục và tìm kiếm.....	45
3.3.1. Phát biểu bài toán.....	45
3.3.2. Lập chỉ mục và tìm kiếm bằng Elasticsearch .....	46
<b>CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM, ĐÁNH GIÁ VÀ KẾT LUẬN .....</b>	<b>50</b>
4.1. Công cụ, môi trường thực nghiệm.....	50
4.2. Kết quả thực nghiệm, đánh giá.....	51
4.3. Kết luận.....	54
4.3.1. Kết quả đạt được .....	54
4.3.2. Định hướng phát triển.....	55
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>56</b>

**BẢNG CHỮ CÁI VIẾT TẮT**

<b>STT</b>	<b>Từ viết tắt</b>	<b>Ý nghĩa</b>
1	ASR	Automatic Speech Recognition – Nhận dạng tiếng nói tự động
2	FPS	Frame Per Second – Số khung hình trên một giây
3	FTP	File Transfer Protocol – Giao thức truyền tệp tin
4	GNU	General Public License – Giấy phép công cộng
5	OCR	Optical Character Recognition – Nhận dạng kí tự quang học
6	PDF	Portable Document Format – Định dạng tài liệu di động.
7	NDD	Near Duplicate Detection – Phát hiện gần trùng lặp
8	TIFF	Tagged Image File Format – Định dạng tệp tin trên máy tính để lưu trữ các hình ảnh.
9	UTF-8	Unicode Transformation Format - Định dạng chuyển đổi Unicode.

**DANH MỤC CÁC BẢNG BIỂU**

Bảng 3.1. Kết quả Bigram tập dữ liệu .....	44
Bảng 4.1. Thông số phân cứng.....	50
Bảng 4.2. Danh sách công cụ phần mềm .....	50
Bảng 4.3. Kết quả thực hiện trích xuất khung hình từ video .....	51
Bảng 4.4. Kết quả thực hiện Tesseract-OCR đối với tập khung hình thu được.....	52
Bảng 4.5. Kết quả thực hiện NDD với kỹ thuật Shingling .....	52
Bảng 4.6. Kết quả quá trình phát hiện lỗi chính tả dùng Aspell kết hợp Bi-gram...	53
Bảng 4.7. Kết quả quá trình sửa lỗi chính tả .....	54

## DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

Hình 1.1. Quá trình đánh chỉ mục .....	12
Hình 2.1. Kiến trúc tổng quan hệ thống tìm kiếm video dựa trên nội dung .....	17
Hình 2.2. Kiến trúc hệ thống tìm kiếm video tác giả đề xuất .....	18
Hình 2.3. Sử dụng FFMpeg để chuyển đổi video thành ảnh .....	20
Hình 2.4. Kiến trúc của Tesseract – OCR.....	22
Hình 2.5. Văn bản gốc.....	23
Hình 2.6. Văn bản trùng lặp của văn bản trong hình 2.5 .....	24
Hình 2.7. Văn bản gần trùng lặp của văn bản trong hình 2.5. ....	24
Hình 2.8 <sup>[15]</sup> . Độ chính xác và độ hồi tưởng của độ đo tương tự cho phương pháp fuzzy-fingerprinting (FF), localitysensitive hashing (LSH), supershingling (SSh), shingling (Sh), and hashed breakpoint chunking (HBC).....	26
Hình 2.9. Kỹ thuật phát hiện lỗi chính tả dựa vào tra cứu từ điển.....	27
Hình 2.10. Kỹ thuật phát hiện lỗi chính tả dựa vào phân tích N-gram.....	28
Hình 2.11. Thứ hạng của 17 công cụ tìm kiếm. Nguồn <a href="http://db-engines.com">http://db-engines.com</a> ..	30
Hình 2.12. Kiến trúc cluster-node-shard của Elasticsearch .....	31
Hình 3.1. Mô tả quá trình biến đổi video nguồn thành dạng ảnh .....	33
Hình 3.2. Chuyển đổi ảnh màu thành ảnh đa cấp xám .....	34
Hình 3.3. Ảnh màu .....	35
Hình 3.4. Ảnh đa cấp xám.....	35
Hình 3.5. Quá trình OCR ảnh trong hình 3.4 bằng Tesseract-OCR .....	36
Hình 3.6. Kết quả sau khi hoàn thành OCR bằng Tesseract-OCR .....	36
Hình 3.7. Thực hiện OCR tất cả ảnh trong thư mục bằng Tesseract-OCR .....	36
Hình 3.8. Quá trình xử lý trùng lặp văn bản .....	37
Hình 3.9. Hệ số Jaccard của tài liệu $d_1$ và $d_2$ .....	38
Hình 3.10 <sup>[4]</sup> . Bốn quá trình tính toán shingle của hai tài liệu. ....	39
Hình 3.11. Sơ đồ khối quá trình trích xuất tập văn bản đại diện .....	40
Hình 3.12. Quá trình phát hiện và sửa lỗi chính tả văn bản.....	41
Hình 3.13. Sơ đồ khối sửa lỗi chính tả sử dụng từ điển Aspell .....	43
Hình 3.14. Sơ đồ khối sửa lỗi chính tả sử dụng Bigram.....	45
Hình 3.15. Mô tả quá trình lập chỉ mục tài liệu .....	46
Hình 3.16. Kiểm tra khởi động Elasticsearch .....	46
Hình 3.17. Danh sách các chỉ mục hiện có. Tên chỉ mục là lectures, số tài liệu docs.count hiện tại có giá trị bằng 0 (do chưa tạo tài liệu cho chỉ mục này).....	47
Hình 3.18. Tạo type và document cho chỉ mục. ....	47
Hình 3.19. Tạo type và document bằng lệnh POST. Id của document được Elasticsearch gán tự động.....	47



Hình 3.20. Cập nhật lại document cho chỉ mục với id đã tồn tại.....	48
Hình 3.21. Thực hiện cập nhật lại document bằng câu lệnh UPDATE.....	48
Hình 3.22. Tìm kiếm document trên chỉ mục .....	48

## MỞ ĐẦU

Cùng với sự phát triển của công nghệ thông tin, tốc độ internet đang cải thiện đáng kể. Số lượng video bài giảng, diễn thuyết... phục vụ học tập cho mọi lứa tuổi đang được tải lên và chia sẻ trên internet nhanh chóng. Mỗi ngày, hàng triệu video như vậy trên thế giới được đăng tải lên các ứng dụng internet như Youtube, Facebook, Yahoo. Đối với lượng video đang tăng trưởng từng ngày này, cơ chế tổ chức lưu trữ phục vụ cho việc tra cứu, tìm kiếm là một thách thức.

Giáo dục trực tuyến hay E-Learning không còn là khái niệm mới lạ và đang phát triển mạnh mẽ. Số lượng video bài giảng, diễn thuyết cũng vì thế ngày càng được tăng trưởng. Nhu cầu tìm kiếm của người học càng yêu cầu khắt khe hơn: cả về độ chính xác và thời gian tìm kiếm. Tuy nhiên, các chức năng tìm kiếm bài giảng cho của các hệ thống hiện tại thông thường chỉ cho phép người dùng tìm kiếm với tên bài giảng, tên học phần, hoặc tên giảng viên... Các chức năng này thường cho kết quả có độ chính xác không cao, và các kết quả trả về có nhiều nội dung không liên quan đến mục đích tìm kiếm thực sự của người dùng. Do đó, cần có một hệ thống mà có thể “hiểu” được nội dung của từng video bài giảng để phục vụ cho việc tìm kiếm của người dùng.

Những công cụ tìm kiếm phổ biến hiện nay - như Google, Yahoo, Bing..., là những hệ thống tìm kiếm dựa trên “từ khóa”, và tìm kiếm trên dữ liệu văn bản (text). Chính vì thế, nếu video không có bất kỳ siêu dữ liệu (metadata) ví dụ như ngày, tác giả, từ khóa, hoặc mô tả thì không thể tìm kiếm được bằng cách sử dụng các công cụ nêu trên. Siêu dữ liệu thường được thêm bằng tay, quá trình này sẽ rất tốn thời gian. Hơn nữa, ngay cả khi một đoạn video có thể được tìm thấy bằng siêu dữ liệu của nó, công cụ tìm kiếm thông thường không có khả năng tìm kiếm một đoạn bài giảng, slide cụ thể trong video mà người dùng quan tâm.

Mục tiêu chính của của Luận văn là tập trung  *nghiên cứu xây dựng một hệ thống tìm kiếm các bài giảng, thuyết trình, trình diễn bằng slide dưới dạng video*. Hệ thống sẽ cho phép người dùng chỉ cần nhập vào một phần nội dung của bài giảng, kết quả trả về sẽ là những video bài giảng có liên quan đến chuỗi truy vấn. Ngoài ra, với giải pháp này cũng cho phép các hệ thống tìm kiếm có thể truy vấn dữ liệu video mà không cần có siêu dữ liệu. Xuất phát từ quan điểm nêu trên, ngoài phần mở đầu và kết luận, luận văn được chia làm 4 chương được tóm tắt như sau:

- Chương 1: Giới thiệu về công cụ tìm kiếm trên mạng internet, các khái niệm và kiến trúc của công cụ tìm kiếm. Các vấn đề cần giải quyết trong luận văn và ý nghĩa khoa học, thực tiễn của luận văn.

- Chương 2: Trình bày về các bài toán cần giải quyết trong khuôn khổ tìm kiếm video bài giảng dạng slide. Một số khái niệm, mô hình các bài toán cần giải quyết. Các phương pháp tiếp cận để giải quyết vấn đề.

- Chương 3: Là chương quan trọng nhất của Luận văn. Nội dung chính của chương này là tập trung trình bày giải pháp thực hiện của tác giả, các kỹ thuật áp dụng để trích xuất văn bản, xử lý văn bản và đánh chỉ mục tìm kiếm cho video bài giảng.

- Chương 4: Là phần trình bày các kết quả thực nghiệm và đánh giá. Ở mỗi bài toán tác giả đều có những thực nghiệm để kiểm chứng và đánh giá về độ chính xác.

Tác giả xin bày tỏ lòng biết ơn chân thành tới PGS.TS. Nguyễn Trí Thành, thầy đã luôn ân cần, chỉ bảo, động viên, giúp đỡ tác giả trong suốt quá trình thực hiện Luận văn. Tác giả xin chân thành cảm ơn gia đình, bạn bè, đồng nghiệp đã luôn tin tưởng, động viên và giúp đỡ về nhiều mặt trong thời gian qua. Tác giả xin chân thành cảm ơn các thầy, cô giáo trong khoa Công nghệ Thông tin và Truyền thông, trường Đại học Hồng Đức đã động viên và tạo điều kiện giúp đỡ tác giả hoàn thành tốt nhất luận văn này.

## CHƯƠNG 1: GIỚI THIỆU

### 1.1. Giới thiệu về công cụ tìm kiếm

Nếu bạn đã từng truy cập địa chỉ [www.google.com.vn](http://www.google.com.vn), nhập nội dung cần tra cứu và bấm vào “tìm với google”. Một danh sách kết quả liên quan đến nội dung tìm kiếm được liệt kê trên màn hình cho phép người dùng lựa chọn các nội dung phù hợp với yêu cầu. Những công cụ cho phép người dùng tìm kiếm các thông tin trên mạng như Google, Bing, Yahoo... như vậy gọi là các công cụ tìm kiếm (web search engine).

Thuật ngữ “web search engine” được định nghĩa: “Một công cụ tìm kiếm là các ứng dụng thực tế của các kỹ thuật truy hồi thông tin trên miền dữ liệu văn bản qui mô lớn”<sup>[5]</sup>.

Để hiểu được lợi ích của các công cụ tìm kiếm chúng ta sẽ cần nắm rõ một số khái niệm liên quan:

- Thông tin (information): Là những hiểu biết được về một thực thể nào đó. Ví dụ như nội dung của luận này là thông tin.

- Dữ liệu (data): Là cái để biểu diễn thông tin dưới các dạng ký hiệu, chữ viết, chữ số, hình ảnh, âm thanh hoặc dạng tương tự. Ví dụ: quyển sách là dữ liệu.

- Truy hồi thông tin (information retrieval): Là các giải pháp để thu thập, mô hình hóa, biểu diễn, tổ chức, lưu trữ dữ liệu nhằm phục vụ quá trình tìm kiếm, truy cập thông tin mà người dùng quan tâm được thuận tiện, nhanh chóng và chính xác nhất có thể.<sup>[4]</sup>

### 1.2. Lịch sử phát triển của công cụ tìm kiếm

Năm 1990, Archie là công cụ tìm kiếm đầu tiên được phát triển bởi Alan Emtage, Bill Heelan and J. Peter Deutsch, hai sinh viên chuyên ngành khoa học máy tính của trường McGill University tại Montreal (Canada). Chương trình cho phép lập chỉ mục danh sách các tệp tin tải về qua FTP.

Năm 1991, một công cụ tương tự Archie là Gopher của tác giả Mark McCahill tại University of Minnesota, có chức năng tìm kiếm theo tên tệp tin và tiêu đề được lưu trữ trong hệ thống Gopher đã lập chỉ mục.

Năm 1993, đánh dấu những bước tiến mới về công cụ tìm kiếm như World Wide Web Wanderer bởi Matthew Gray, đây được xem là một web robot đầu tiên đo lường được dung lượng của trang web. Hay công cụ Aliweb cho phép người dùng cập nhật các trang web vào bộ chỉ mục (index).

Năm 1994, với sự ra đời của WebCrawler công cụ tìm kiếm đầu tiên chỉ mục toàn trang web và cho phép người dùng tìm kiếm và thu thập với bất kỳ từ nào một cách tự động.

Năm 1995, công cụ tìm kiếm yahoo được tạo bởi David Filo và Jerry Yang. Sử dụng danh bạ web thay vì đánh chỉ mục toàn văn bản.

Năm 1996-nay, với sự phát triển mạnh mẽ của internet các công cụ tìm kiếm phát triển mạnh mẽ hơn, tối ưu hơn nhiều so với các công cụ trước đây. Năm 1998, Google được phát triển bởi Larry và Sergey đưa ra khái niệm về PageRank (thứ hạng của một trang web), đánh dấu sự phát triển vượt bậc và hiện đang là công cụ tìm kiếm có thị phần lớn nhất hiện nay.

### **1.3. Kiến trúc của công cụ tìm kiếm**

Trong phần này tác giả sẽ mô tả kiến trúc cơ bản của một công cụ tìm kiếm. Các thành phần và các mối quan hệ giữa các thành phần có trong nó.

Trước tiên, thuật ngữ kiến trúc được hiểu là bản thiết kế để đảm bảo rằng hệ thống sẽ đáp ứng các yêu cầu hoặc mục tiêu của ứng dụng.

Hai mục tiêu chính của công cụ tìm kiếm đó là:

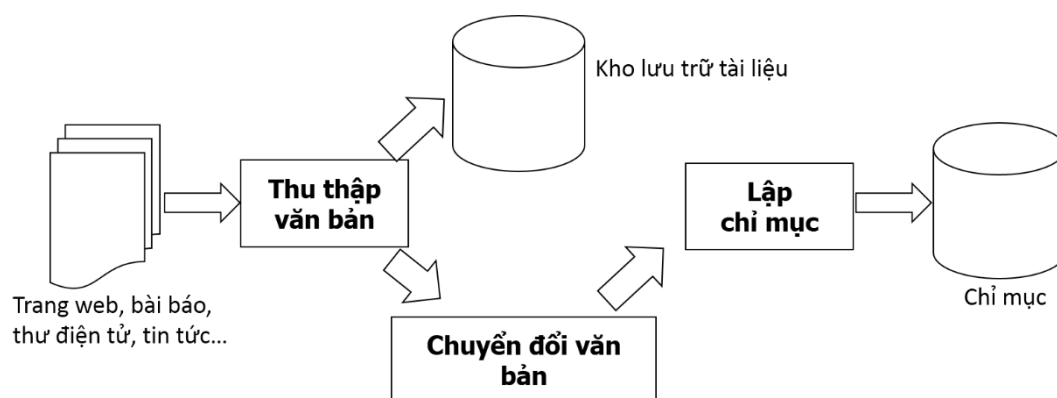
- Hiệu quả (chất lượng): chúng ta mong muốn lấy được thông tin liên quan nhất có thể khi truy vấn.
- Tốc độ: chúng ta mong muốn quá trình xử lý truy vấn từ người dùng càng nhanh càng tốt.

Kiến trúc thông thường của công cụ tìm kiếm hiện nay được xác định bởi hai yêu cầu nêu trên. Bởi vì chúng ta muốn một hệ thống có tốc độ cao, công cụ tìm kiếm sẽ sử dụng cấu trúc dữ liệu được tối ưu hóa để truy hồi thông tin nhanh chóng. Bởi vì chúng ta muốn kết quả có chất lượng cao, công cụ tìm kiếm sẽ xử lý văn bản và lưu trữ văn bản thống kê cẩn thận để giúp cải thiện sự liên quan của kết quả.

Các thành phần của công cụ tìm kiếm hỗ trợ cho hai chức năng chính đó là quá trình đánh chỉ mục, và quá trình truy vấn.

#### **1.3.1. Quá trình đánh chỉ mục**

Quá trình đánh chỉ mục tạo nên những cấu trúc mà cho phép việc tìm kiếm, và quá trình truy vấn sử dụng những cấu trúc này và truy vấn của người dùng để xây dựng một danh sách xếp hạng của tài liệu. Hình 1.1 thể hiện ở mức cao các thành phần của quá trình đánh chỉ mục.



**Hình 1.1.** Quá trình đánh chỉ mục

Các thành phần chính bao gồm việc thu thập văn bản, chuyển đổi văn bản và tạo chỉ mục.

Nhiệm vụ của việc thu thập văn bản là để xác định và làm tài liệu được sẵn sàng cho việc tìm kiếm sau này. Trong các công cụ tìm kiếm hiện nay, công việc này thông thường được thu thập bằng crawling hoặc quá trình quét tự động các trang web, hoặc các nguồn thông tin khác nhau. Ngoài ra, để có được tài liệu phục vụ cho quá trình tiếp theo là truy vấn thì quá trình thu thập văn bản sẽ tạo ra một kho lưu trữ tài liệu. Kho lưu trữ tài liệu bao gồm văn bản và siêu dữ liệu cho tất cả tài liệu. Siêu dữ liệu là thông tin về tài liệu mà không bao gồm phần nội dung của tài liệu. Ví dụ như kiểu của tài liệu (email, trang web, video....), cấu trúc của tài liệu, và các đặc điểm của tài liệu như (dung lượng, độ dài...).

Chuyển đổi văn bản là quá trình biến đổi tài liệu vào các chỉ mục thuật ngữ. Chỉ mục thuật ngữ là các phần của tài liệu mà được lưu trữ trong chỉ mục và được sử dụng trong việc tìm kiếm. Thuật ngữ chỉ đơn giản là một từ, nhưng không phải tất cả các từ có thể được sử dụng để tìm kiếm.

Thành phần tạo chỉ mục là kết quả của quá trình chuyển đổi văn bản và tạo ra các chỉ mục hoặc cấu trúc dữ liệu để cho phép việc tìm kiếm nhanh hơn. Với số lượng lớn các tài liệu trong nhiều ứng dụng tìm kiếm, tạo chỉ mục phải có hiệu quả cả về thời gian và không gian. Các chỉ mục này phải có khả năng được cập nhật một cách hiệu quả khi có các tài liệu mới. Thông thường có hai phương pháp để đánh chỉ mục là:

- Ánh xạ từ tài liệu đến thuật ngữ.
- Ánh xạ từ thuật ngữ đến tài liệu (chỉ mục ngược: inverted index).

Quá trình lập chỉ mục cho tài liệu là một trong những phần quan trọng nhất của công cụ tìm kiếm.

### 1.3.2. Quá trình truy vấn

Phần còn lại của công cụ tìm kiếm là quá trình truy vấn. Quá trình truy vấn thông thường bao gồm ba thành phần chính là tương tác người dùng, xếp hạng và đánh giá.

Thành phần thứ nhất, tương tác người dùng cung cấp các giao diện tương tác giữa người dùng và công cụ tìm kiếm. Nhiệm vụ của phần này gồm: thứ nhất là tiếp nhận câu truy vấn từ người dùng và chuyển đổi vào trong chỉ mục thuật ngữ, thứ hai là nhận danh sách thứ hạng đã được sắp xếp từ công cụ tìm kiếm và trình bày kết quả theo thứ tự cho người dùng.

Thành phần xếp thứ hạng là trung tâm của một công cụ tìm kiếm. Nó sẽ chuyển đổi truy vấn từ giao diện người dùng và tạo ra một danh sách tài liệu đã được sắp xếp bằng điểm số dựa vào một mô hình truy hồi thông tin. Việc xếp thứ hạng phải đảm bảo cả về thời gian và hiệu quả. Có nghĩa là, có thể nhiều truy vấn cần được xử lý trong thời gian ngắn và chất lượng của bảng xếp hạng sẽ quyết định đến kết quả của công cụ tìm kiếm đó có tốt hay không. Về tốc độ xử lý truy vấn phụ thuộc vào việc đánh chỉ mục, hiệu quả của xử lý truy vấn sẽ phụ thuộc và mô hình truy hồi thông tin.

Nhiệm vụ của thành phần đánh giá đó là đo và điều khiển hiệu quả và tốc độ của công cụ tìm kiếm. Một phần quan trọng của đánh giá đó là ghi lại và phân tích hành vi của người dùng bằng cách sử dụng dữ liệu nhật ký (log data). Kết quả của việc đánh giá đó là để điều chỉnh và cải thiện thành phần xếp hạng. Việc đánh giá chủ yếu là các hoạt động bên ngoài, không nằm trong công cụ tìm kiếm trực tuyến, nhưng nó là một phần quan trọng của bất kỳ ứng dụng tìm kiếm nào.

### 1.4. Công cụ tìm kiếm video trên mạng internet

Sự phát triển của hạ tầng công nghệ là nền tảng cho sự phát triển nhanh chóng của internet. Dữ liệu trên mạng truyền đi không còn gói gọn trong các văn bản (text) bình thường nữa mà rất đa dạng về chủng loại như âm thanh, hình ảnh, video... và đang ngày càng tăng trưởng với tốc độ lớn.

Công cụ tìm kiếm video cho phép người dùng thuận tiện trong quá trình tìm kiếm phục vụ nhiều mục đích khác nhau như giải trí, giáo dục và truyền thông. Nếu chia theo lĩnh vực thì video cũng rất đa dạng như video truyền hình, video quảng cáo, video bài giảng, học thuyết... Trong nội dung của luận văn này tác giả chỉ đề cập đến thể loại video bài giảng dạng slide, cách tiếp cận và hướng giải quyết để xây dựng một công cụ tìm kiếm video bài giảng dạng slide. Việc tìm kiếm yêu cầu bằng cách duyệt qua tập các video kết quả.

Video là một dạng băng từ dùng cho việc ghi lại các chuyển động hình ảnh và âm thanh. Video là phương tiện truyền liên tục (hoặc tuyến tính): nếu tạm dừng, chỉ có một khung hình duy nhất vẫn còn, âm thanh bị mất. Việc lưu trữ và chuyển đổi video là thách thức lớn hơn nhiều so với dữ liệu kiểu văn bản. Các đặc trưng của văn bản (kí tự, từ) thì có thể được xác định, mã hóa và giới hạn được. Nhưng đối với các đặc trưng của video (cạnh, màu, chuyển động, độ cao của âm thanh...) thì việc xác định, trích xuất và lấy mẫu khó hơn. Hơn nữa đối với văn bản thì người dùng có thể truy vấn một cách dễ dàng bằng cách gõ trực tiếp lên bàn phím, còn đối với tìm kiếm video thì truy vấn đầu vào là văn bản và kết quả ra lại là video.

## **1.5. Tổng quan của đề tài và các vấn đề cần giải quyết**

### **1.5.1. Tổng quan đề tài**

Trong đề tài này, tác giả hướng tới xây dựng một hệ thống tìm kiếm các video bài giảng, thuyết trình, trình diễn bằng silde dưới dạng video... Cho phép tìm thấy những video bằng văn bản xuất hiện trong đó. Với giải pháp này, đơn giản bằng cách nhập từ khóa tìm kiếm, người dùng có thể tìm kiếm các video bài giảng và những cảnh trong đó mà thuật ngữ xuất hiện. Giải pháp này cũng cho phép người dùng tìm kiếm các video không cần có siêu dữ liệu.

### **1.5.2. Các vấn đề cần giải quyết**

Vấn đề cần giải quyết ở trong đề tài này là giải pháp xử lý video đầu vào. Phân tích và đánh chỉ mục cho video. Đầu tiên, các đoạn video tĩnh trong một thời gian nhất định được xác định là các slide và trích xuất từ video. Tiếp theo, các dữ liệu văn bản chứa trong hình ảnh của slide được trích xuất bằng cách sử dụng kĩ thuật nhận dạng kí tự quang học. Các văn bản trích xuất sẽ được xử lý trùng lặp, sửa lỗi chính tả và được đánh chỉ mục tương ứng với video gốc lưu trữ trong cơ sở dữ liệu.

Sau đó người dùng có thể tìm kiếm các video bài giảng thông qua một giao diện trình duyệt web. Đầu vào, người dùng chỉ cần gõ bất kỳ từ khóa nào có liên quan đến nội dung video thì đầu ra sẽ là danh sách kết quả liên quan được liệt kê.

## **1.6. Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu**

### **1.6.1. Ý nghĩa khoa học**

- Đề tài đã tổng quát được các phương pháp khoa học để giải quyết vấn đề lập chỉ mục video bài giảng, phục vụ quá trình truy hồi thông tin.

- Đề tài cung cấp các cơ sở khoa học, định hướng cho các nghiên cứu về xử lý lập chỉ mục cho video bài giảng.



- Từ kết quả nghiên cứu của đề tài, góp phần làm cơ sở thực tiễn và lý luận để phát triển hệ thống tìm kiếm video dựa trên nội dung.

### **1.6.2. Ý nghĩa thực tiễn**

- Hướng tiếp cận mới cho các máy tìm kiếm, truy hồi thông tin video dựa trên nội dung.

- Góp phần nâng cao chất lượng của các máy tìm kiếm. Kết quả trả về có độ liên quan cao hơn so với phương pháp tìm kiếm dựa trên từ khóa hiện nay.

### **1.7. Kết luận**

Trong chương này, luận văn đã giới thiệu khái quát một số khái niệm, lịch sử và kiến trúc của một công cụ tìm kiếm nói chung. Ngoài ra, luận văn cũng giới thiệu tổng quan các vấn đề cần giải quyết của đề tài. Ý nghĩa khoa học và ý nghĩa thực tiễn của đề tài nghiên cứu xây dựng hệ thống tìm kiếm video dựa trên nội dung.

Chương tiếp theo, luận văn sẽ trình bày chi tiết các tiếp cận để giải quyết từng vấn đề trong bài toán tìm kiếm video bài giảng dạng slide dựa trên nội dung.

## CHƯƠNG 2: BÀI TOÁN TÌM KIẾM VIDEO BÀI GIẢNG DỰA TRÊN NỘI DUNG

### 2.1. Phát biểu bài toán

Trong khuôn khổ luận văn này, tác giả chỉ đề cập đến các video bài giảng, thuyết trình dưới dạng slide và bài toán liên quan đến quá trình xây dựng công cụ tìm kiếm những video dạng nói trên. Ngoài ra, còn rất nhiều chủng loại video khác nữa, và nội dung nghiên cứu các video khác là nằm ngoài khuôn khổ trong luận văn. Trọng tâm của luận văn là nghiên cứu cách thức xử lý và lập chỉ mục cho video đầu vào.

Tác giả sẽ xây dựng công cụ tìm kiếm cho phép nhận nội dung truy vấn là chuỗi văn bản và kết quả trả về là các video bài giảng mà nội dung có liên quan đến chuỗi văn bản người dùng truy vấn.

Như đã trình bày ở chương 1, công việc cần giải quyết đối với bài toán này gồm hai việc. Thứ nhất, trích xuất được nội dung từ video đầu vào để lập chỉ mục. Thứ hai, lập chỉ mục cho video và xử lý truy vấn tìm kiếm từ người dùng. Bài toán tìm kiếm video dựa trên nội dung được chia thành hai bài toán con được mô tả như sau:

**Bài toán 1:** Xử lý video đầu vào, trích xuất văn bản từ video.

Đầu vào:

- Tập videos bài giảng dạng slide.

Đầu ra:

- Văn bản trích xuất nội dung từ video đầu vào.

**Bài toán 2:** Lập chỉ mục và tìm kiếm video dựa trên nội dung bài giảng.

Đầu vào:

- Truy vấn từ người dùng.

Đầu ra:

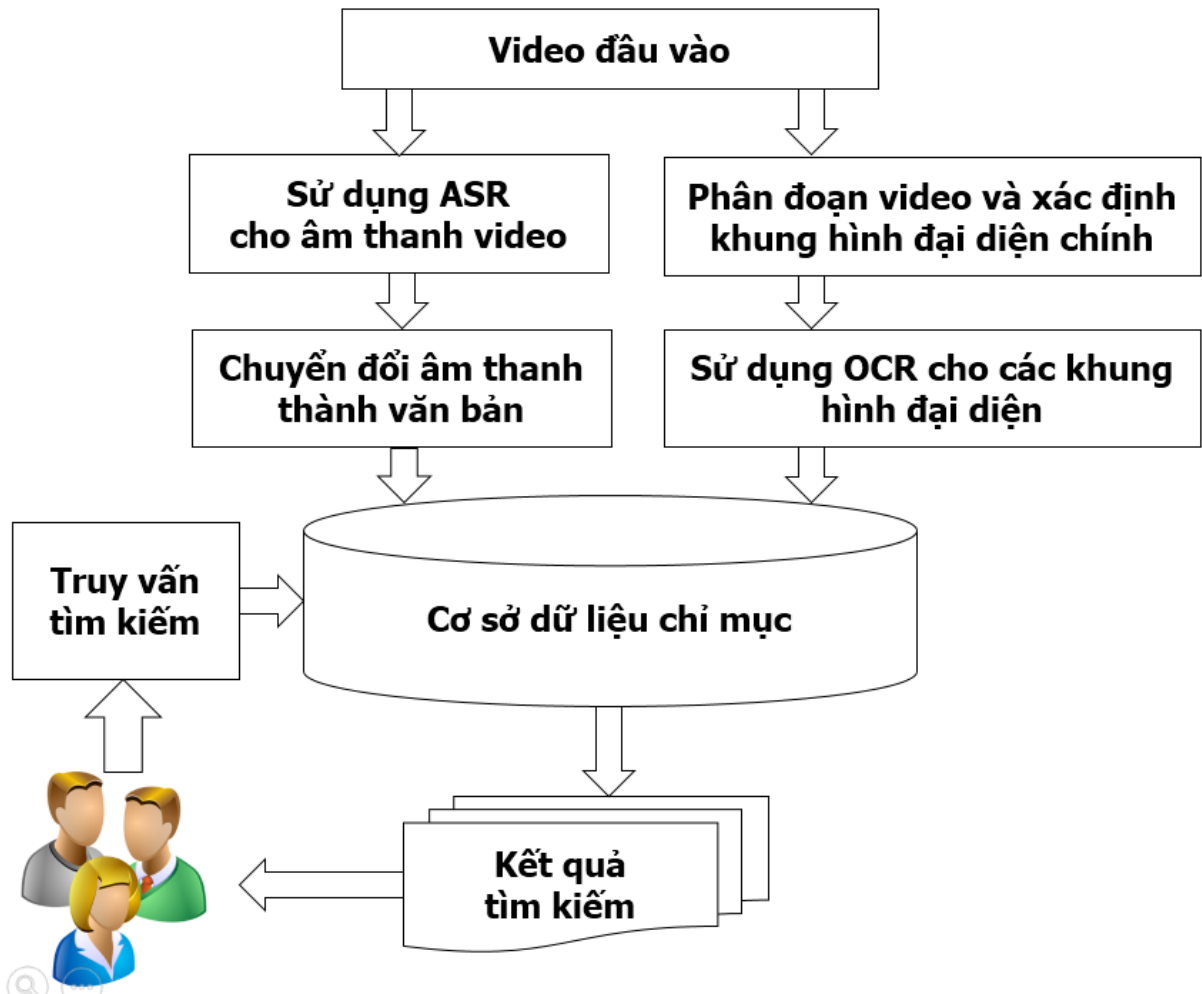
- Danh sách kết quả videos có nội dung liên quan đến truy vấn.

Trong bài toán thứ nhất, vì đặc tính của video là có cả hình ảnh và âm thanh nên sẽ có hai cách phương pháp tiếp cận chính để trích xuất văn bản từ video là:

- Phân tách video thành các khung hình để lựa chọn khung hình đại diện chính, sau đó sử dụng công nghệ OCR để trích xuất văn bản từ các khung hình đó.

- Sử dụng công nghệ nhận dạng giọng nói tự động, để chuyển đổi phân âm thanh của video thành văn bản.

Kiến trúc chung của một hệ thống tìm kiếm video dựa vào nội dung được miêu tả trong hình 2.1.



**Hình 2.1.** Kiến trúc tổng quan hệ thống tìm kiếm video dựa trên nội dung

Hình 2.1 cho thấy quá trình lập chỉ mục cho video được trải qua ba bước là phân đoạn video, trích xuất nội dung từ video và lập chỉ mục cho video.

## 2.2. Các nghiên cứu về tìm kiếm video dựa trên nội dung

Liška et al và cộng sự đã đề xuất một hệ thống tự động cho việc lập chỉ mục video bài giảng [8]. Họ sử dụng toàn bộ khung hình phân đoạn được từ video và sử dụng công cụ OCR để trích xuất văn bản trên tập khung hình đó. Văn bản sau khi thu thập được tiến hành lập chỉ mục và cho phép tìm kiếm. Giải pháp này hiệu quả kém do không loại bỏ các tệp văn bản trùng lặp. Thời gian xử lý video mất nhiều thời gian do số lượng lớn các khung hình.

Hunter et al đề xuất một hệ thống lập chỉ mục cho các bài thuyết trình đa phương tiện[7]. Đầu tiên, mọi người sẽ phải chuẩn bị một tệp tin thuyết trình định dạng PDF và gửi lại sau khi đã trình bày. Sau đó tệp tin sẽ được đồng bộ với video thuyết trình. Công việc OCR sẽ được thực hiện trên tệp tin PDF mà không cần quan tâm đến video thuyết trình.

Yang et al sử dụng công cụ nhận dạng giọng nói tự động ASR để trích xuất nội dung video thành văn bản[8]. Các kết quả cho thấy độ chính xác của nhận dạng giọng nói thấp hơn rất nhiều so với công nghệ OCR.

Lienhart et al đề xuất một phương pháp phát hiện văn bản trong video và hình ảnh[8]. Họ xây dựng một mạng nơ-ron nhiều tầng để huấn luyện phát hiện văn bản. Thuật toán của họ xử lý với tất cả các khung hình phân đoạn được và cách tiếp cận này kém hiệu quả về thời gian xử lý.

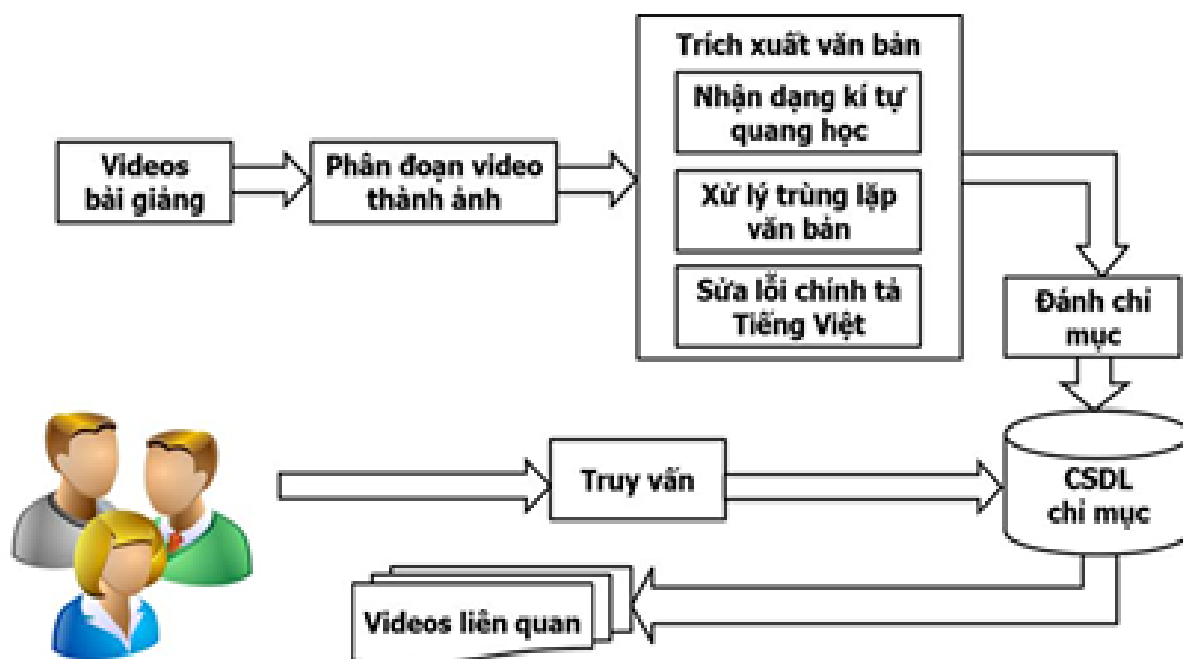
### 2.3. Hướng nghiên cứu của tác giả

Dựa vào các phương pháp tiếp cận nghiên cứu đã nêu trong phần 2.2, tác giả lựa chọn phương pháp tiếp cận để trích xuất văn bản từ video bằng công nghệ OCR thay vì sử dụng ASR.

Công cụ tìm kiếm video mà tác giả mong muốn xây dựng được hình thành từ cách giải quyết các bài toán cụ thể sau:

- Phân đoạn video.
- Trích xuất văn bản đại diện:
  - + Nhận dạng kí tự quang học.
  - + Xử lý trùng lặp văn bản.
  - + Sửa lỗi chính tả văn bản.
- Đánh chỉ mục và tìm kiếm.

Kiến trúc của công cụ tìm kiếm video dựa vào nội dung mà tác giả đề xuất được mô tả trong hình 2.2.



**Hình 2.2.** Kiến trúc hệ thống tìm kiếm video tác giả đề xuất

## **2.4. Bài toán phân đoạn video thành ảnh**

### **2.4.1. Khái niệm**

Về mặt bản chất thì video mà chúng ta thấy trên tivi, máy tính, điện thoại... được cấu thành từ những ảnh tĩnh. Những ảnh này sau đó được sắp xếp liên tiếp nhau và cùng trình diễn trong một đơn vị thời gian đủ nhỏ để làm cho mắt của chúng ta cảm nhận rằng các đối tượng này đang chuyển động. Các ảnh được trình chiếu nhanh hơn thì chúng ta cảm nhận được mượt mà và linh động hơn. Thông thường thì các video được quay ở khoảng 24-30 hình mỗi giây.

Mỗi hình này được gọi là một frame. Số frame trên một giây được đo bằng một số nguyên được kí hiệu FPS. Một video đơn giản được hiểu là tổng số khung hình được lưu trữ cùng nhau và trình chiếu theo một thứ tự, do vậy một video thông thường có khoảng vài trăm đến vài trăm nghìn khung hình.

### **2.4.2. Phương pháp tiếp cận**

Chúng ta có thể tìm kiếm được phần mềm, công cụ khác nhau để hỗ trợ việc chuyển đổi video thành các frames như phần mềm total video converter, video to picture converter... Nhưng tác giả quan tâm nhất là công cụ mã nguồn mở Ffmpeg bởi ba lý do chính:

- Hỗ trợ nhiều định dạng video khác nhau, ví dụ .mp4, avi, flv...
- Điều chỉnh được FPS.
- Mã nguồn mở.

FFMpeg là một thư viện có rất nhiều tiện ích cho việc xử lý video. Tính năng nổi bật nhất có lẽ là khả năng encode/decode nhiều video định dạng khác nhau, giúp chuyển đổi qua lại nhiều định dạng video. Ngoài ra, chúng ta cũng có thể dùng FFMpeg để chia cắt một đoạn video, chụp lại các frame và xuất ra dạng hình ảnh... Hình 2.3 mô tả câu lệnh mà FFMpeg thực hiện chuyển đổi video thành dạng ảnh.

```

mrhao@mrhao-Satellite-L310: ~/Videos
mrhao@mrhao-Satellite-L310:~/Videos$ ffmpeg -i lecture001.mp4 -r 5 %d.tif
ffmpeg version 2.7.2 Copyright (c) 2000-2015 the FFmpeg developers
  built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04)
  configuration: --disable-yasm
  libavutil      54. 27.100 / 54. 27.100
  libavcodec     56. 41.100 / 56. 41.100
  libavformat    56. 36.100 / 56. 36.100
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter    5. 16.101 / 5. 16.101
  libswscale     3.  1.101 / 3.  1.101
  libswresample  1.  2.100 / 1.  2.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'lecture001.mp4':
  Metadata:
    major_brand      : mp42
    minor_version    : 0
    compatible_brands: isommp42
    creation_time    : 2014-09-16 12:18:30
  Duration: 00:06:22.87, start: 0.000000, bitrate: 591 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1280x718
[SAR 1:1 DAR 640:359], 396 kb/s, 30 fps, 30 tbr, 30 tbn, 60 tbc (default)
  Metadata:
    handler_name     : VideoHandler
  Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, flt
p, 192 kb/s (default)
  Metadata:
    creation_time    : 2014-09-16 12:18:31
    handler_name     : IsoMedia File Produced by Google, 5-11-2011
Output #0, image2, to '%d.tif':
  Metadata:
    major_brand      : mp42
    minor_version    : 0
    compatible_brands: isommp42
    encoder          : Lavf56.36.100
  Stream #0:0(und): Video: tiff, yuv420p, 1280x718 [SAR 1:1 DAR 640:359], q=2-
31. 200 kb/s. 5 fps. 5 tbn. 5 tbc (default)

```

### Hình 2.3. Sử dụng FFMpeg để chuyển đổi video thành ảnh

## 2.5. Bài toán trích xuất văn bản

Trong bài toán trích xuất văn bản, để nâng cao hiệu quả và tránh các hạn chế của các nghiên cứu trước. Tác giả chia bài toán thành ba vấn đề nhỏ hơn đó là:

- Bài toán nhận dạng kí tự quang học để trích xuất văn bản từ video.
- Bài toán xử lý trùng lặp văn bản để thu được tệp văn bản đại diện cho video.
- Bài toán sửa lỗi chính tả Tiếng Việt. Lỗi chính tả phát sinh do quá trình nhận dạng OCR.

### 2.5.1. Bài toán nhận dạng kí tự quang học

#### 2.5.1.1. Khái niệm OCR

Sau khi thu được tập khung hình, tác giả sử dụng kĩ thuật nhận dạng kí tự quang học (Optical Character Recognition) để trích xuất văn bản cho từng

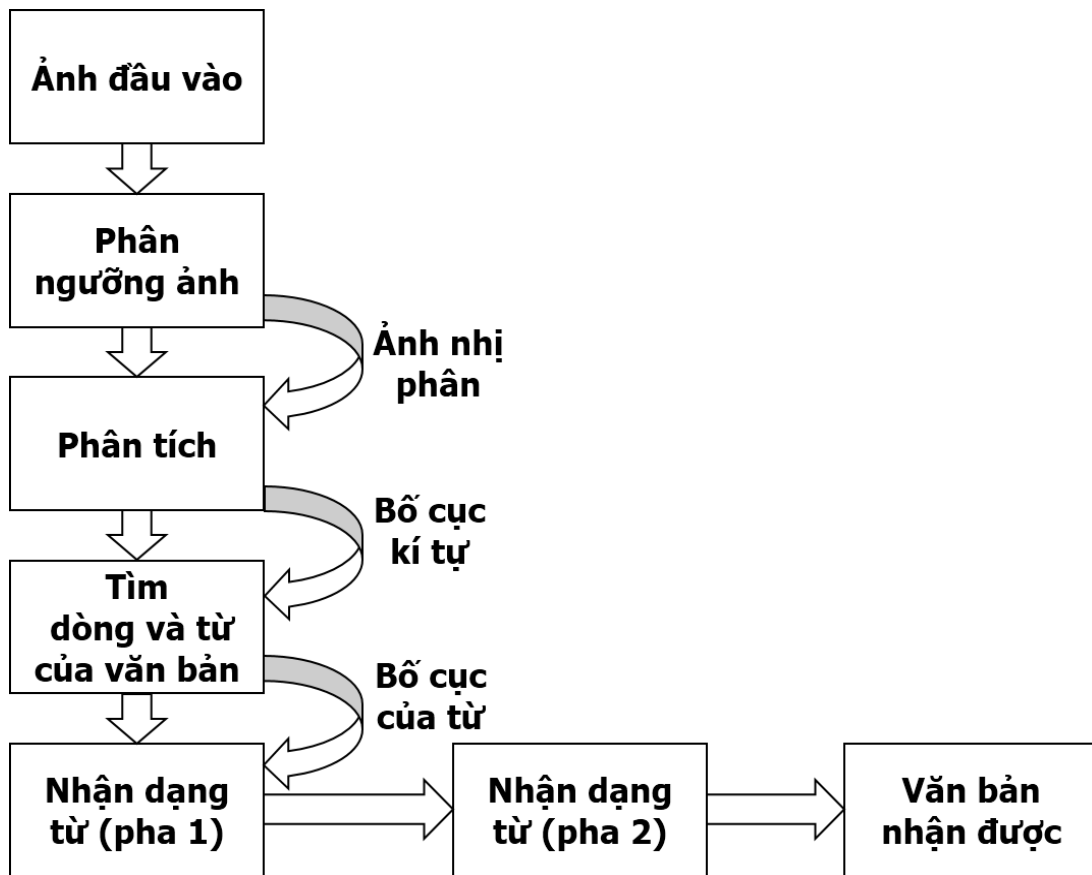
khung hình này. Kết thúc quá trình, kết quả thu được sẽ là một tập văn bản tương ứng với từng khung hình trích xuất được.

OCR là công nghệ cho phép chuyển đổi các loại tài liệu khác nhau, ví dụ như các tài liệu giấy, ảnh chụp hoặc các tập tin PDF bằng một máy ảnh kỹ thuật số thành dữ liệu văn bản có thể chỉnh sửa và tìm kiếm. Những hình ảnh này có thể là các chữ viết tay hoặc đánh máy. Đây là một kỹ thuật phổ biến của việc số hóa các văn bản in để có thể tìm kiếm bằng điện tử, lưu trữ gọn gàng, hiển thị trên mạng.

### **2.5.1.2. Phương pháp tiếp cận**

Tác giả sử dụng Tesseract- OCR để thực hiện trích xuất nội dung văn bản từ ảnh. Tesseract là một công cụ nhận diện kí tự quang học mã nguồn mở và hiện nay được phát triển bởi Google<sup>[8]</sup>. Có nhiều phiên bản, có tính phí, hoặc miễn phí trên mạng mà người dùng có thể tìm được. Nhưng trong phạm vi luận văn này tác giả sử dụng Tesseract-OCR bởi:

- Công cụ miễn phí.
  - Hỗ trợ nhiều hệ điều hành (Windows, Linux, Mac...)
  - Hỗ trợ trích xuất đồng loạt nhiều tập tin cùng lúc.
  - Được tài trợ phát triển bởi Google. Với hỗ trợ trên 100 ngôn ngữ khác nhau.
  - Một trong những công cụ mã nguồn mở OCR chính xác nhất hiện nay.<sup>[19]</sup>
- Hình 2.4 mô tả các bước mà công cụ Tesseract-OCR thực hiện.



**Hình 2.4.** Kiến trúc của Tesseract – OCR

Tesseract thực hiện từng bước như trong hình 2.4. Bước đầu tiên là phân ngưỡng ảnh để chuyển đổi ảnh thành ảnh nhị phân. Bước tiếp theo là quá trình kết nối tới bộ phân tích để trích xuất ra bộ cục các kí tự. Bộ cục này dễ dàng có được dựa trên nền đen và chữ trắng do quá trình chuyển đổi ảnh. Tiếp đến các kí tự sẽ được tổ chức trong những dòng văn bản. Những dòng văn bản này sẽ được phân tích riêng với từng vùng nhất định, hoặc theo từng dòng có kích thước tương đương. Quá trình nhận dạng các từ trong ảnh được thực hiện qua hai pha. Pha thứ nhất sẽ cố gắng nhận dạng từng từ một, với mỗi từ ở pha thứ nhất sẽ truyền sang pha thứ hai như là nơi đồng bộ phân lớp thích nghi. Tại đây dữ liệu sẽ được “học” nhằm cải thiện độ chính xác của quá trình nhận diện.

## **2.5.2. Bài toán xử lý trùng lặp văn bản**

### **2.5.2.1. Khái niệm**

Các khung hình liên tiếp về mặt thời gian tạo thành các đoạn cơ sở (shot). Một video bài giảng có thể gồm nhiều đoạn cơ sở ghép nối lại, chuyển từ đoạn này sang đoạn kia có thể là chuyển cảnh đột ngột hoặc chuyển cảnh dần dần bằng việc sử dụng một số hiệu ứng khi biên tập video. Việc chuyển cảnh trong trường hợp này xảy ra tương đương với việc thay đổi slide trong bài giảng. Vì vậy, các khung hình trong cùng một đoạn cơ sở sẽ có độ tương quan với nhau.



Những tệp văn bản thu được sau khi trích xuất của cùng một đoạn cơ sở là gần trùng nhau về nội dung. Do vậy, việc tóm tắt video có thể được thực hiện bằng cách biểu diễn mỗi đoạn cơ sở chỉ bằng một vài tệp văn bản đại diện.

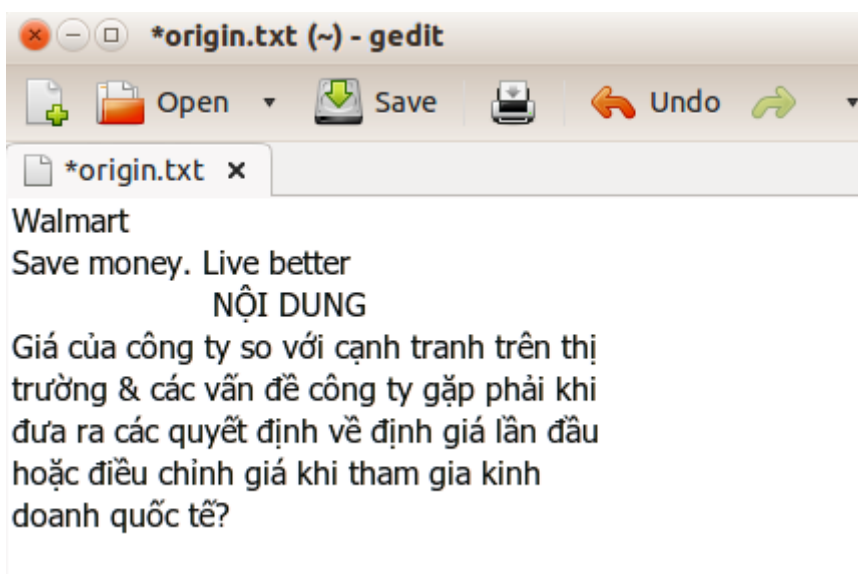
Khi hai văn bản mà nội dung đều giống hệt nhau thì chúng được coi là các văn bản trùng lặp hay gọi là bản sao của nhau. Trong nhiều trường hợp, hai tài liệu mà không phải giống nhau hoàn toàn vẫn có thể chứa cùng một nội dung thì được gọi là các văn bản gần trùng lặp. Một vài trường hợp được quy về văn bản gần trùng lặp:

- Các văn bản chỉ xáo trộn, thêm hoặc bớt vài từ ở nội dung. Dạng phổ biến của văn bản gần trùng lặp.

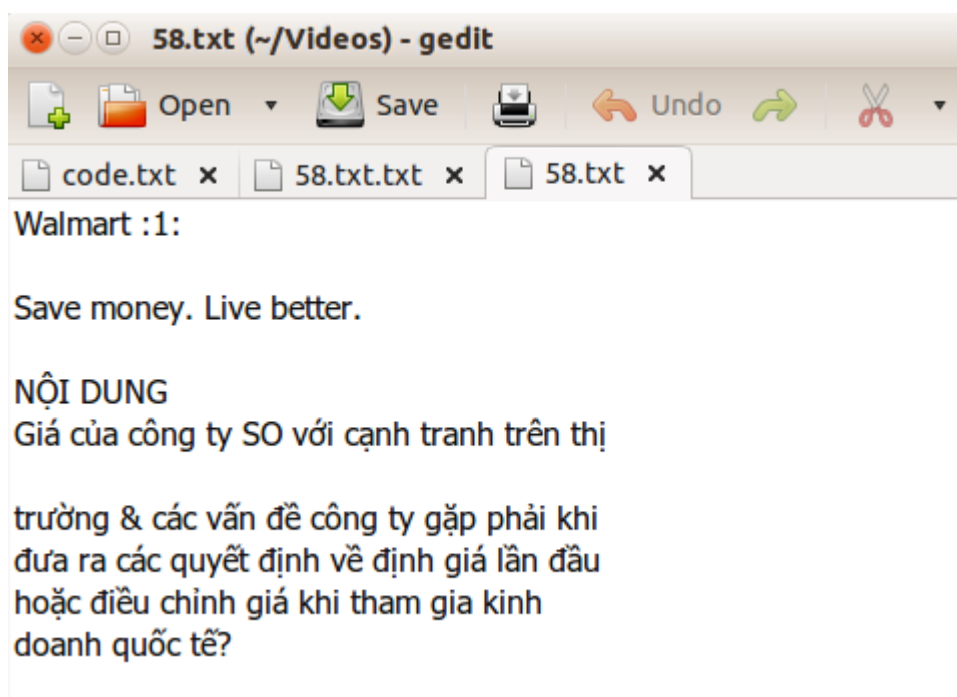
- Các văn bản cùng một nội dung nhưng cách định dạng, phong chữ, bố cục khác nhau.

- Các văn bản nội dung giống nhau, nhưng khác nhau về ngày tạo, ngày sửa chữa, định dạng tệp tin.

Với đặc thù là các văn bản được trích xuất từ các khung hình video bài giảng liên tiếp theo nhau thời gian. Chính vì thế tập hợp văn bản thu được tồn tại cả hai loại đó là trùng lặp và gần trùng lặp văn bản. Hình 2.6 là ví dụ về nội dung văn bản trùng lặp với hình 2.5, hình 2.7 là gần trùng lặp của hình 2.5.

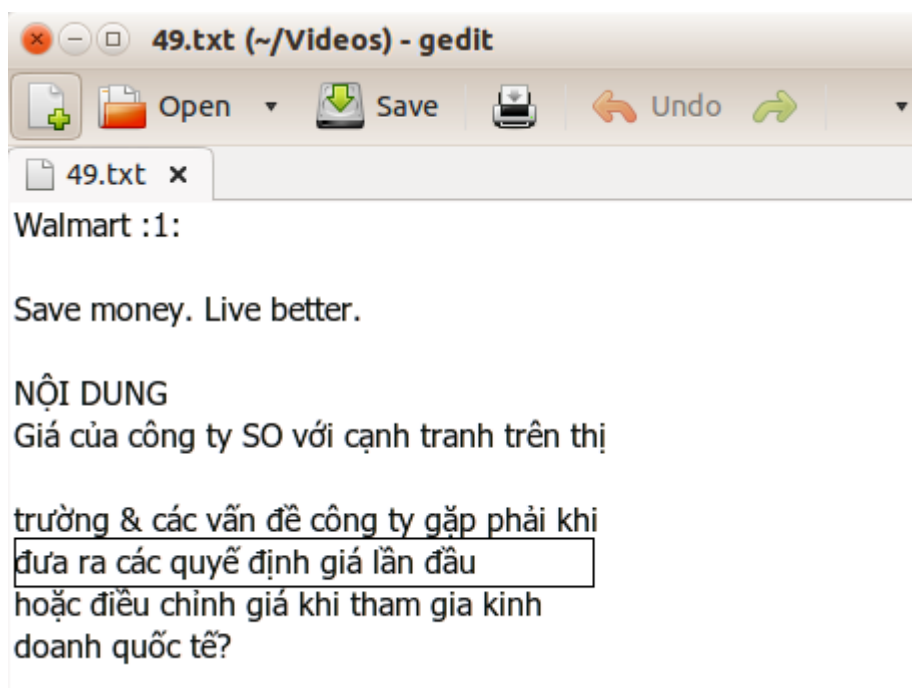


**Hình 2.5.** Văn bản gốc



**Hình 2.6.** Văn bản trùng lặp của văn bản trong hình 2.5

Hình ảnh thu được có nội dung hoàn toàn giống văn bản gốc. Đây là kết quả sau khi OCR bằng Tesseract-OCR với trường hợp mà hình ảnh trích xuất được không bị ảnh hưởng bởi các hiệu ứng trình chiếu làm thay đổi nội dung.



**Hình 2.7.** Văn bản gần trùng lặp của văn bản trong hình 2.5.

Nội dung của văn bản bị thiếu một vài từ so với văn bản gốc. Đây là kết quả khi nhận diện OCR bằng Tesseract-OCR từ ảnh bị hiệu ứng trình chiếu làm ảnh hưởng đến nội dung.

### 2.5.2.2. Phương pháp tiếp cận

Nhiệm vụ chính của bài toán này là phải xác định được các văn bản đại diện cho video bài giảng. Nghĩa là đối với các văn bản trùng lặp hoặc gần trùng lặp (được trích xuất từ cùng một slide) cần được loại bỏ và giữ lại một văn bản làm đại diện.

Theo các nghiên cứu [2], [6], [9], [13],[15] có nhiều phương pháp tiếp cận để giải quyết vấn đề tìm các văn bản trùng lặp như:

- Bag of words: So sánh các từ và tần số của những từ đó trên một văn bản với những văn bản khác.

- Shingling: Cải thiện hơn so với Bag of words, phương pháp này sẽ tiếp cận bằng cách so sánh các cụm từ “shingle”. Phương pháp này quan tâm đến ngữ cảnh của các từ (thứ tự của các từ).

- Hashing: Các cụm từ sẽ được băm thành các con số và sau đó so sánh để tìm ra sự trùng lặp.

- MinHash, SimHash: Cải tiến của phương pháp Hashing, giúp sắp xếp hợp lý quá trình lưu trữ nội dung được băm.

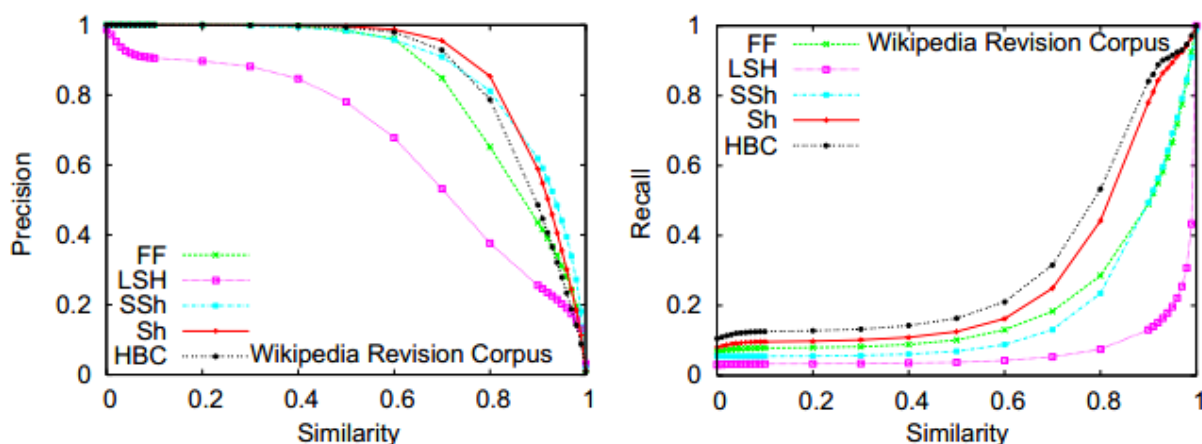
Trên cơ sở phân tích tập dữ liệu có các đặc điểm như:

- Văn bản trùng lặp hoàn toàn: Khi chưa có sự thay đổi slide trong bài giảng. Các văn bản thu được đều là kết quả OCR của cùng một slide bài giảng.

- Văn bản gần trùng lặp: Các văn bản thu được đều là kết quả OCR của một slide nhưng có sự sai khác một vài từ.

- Văn bản không trùng lặp: Các văn bản hoàn toàn khác nhau về nội dung. Khi các văn bản là kết quả OCR của những slide khác nhau trong video.

Dựa trên các kết quả nghiên cứu [2], [6], [9], [13],[15] thì phương pháp shingling cho kết quả độ chính xác cao và phù hợp với kiểu dữ liệu đầu vào như tập dữ liệu của tác giả. Chính vì thế, trong luận văn này, tác giả lựa chọn và cài đặt thuật toán phát hiện trùng lặp văn bản dựa vào kỹ thuật Shingling của Broder và cộng sự. Hình 3.12 bảng kết quả độ chính xác và độ hồi tưởng của các kỹ thuật tìm trùng lặp văn bản theo nghiên cứu [15].



**Hình 2.8** <sup>[15]</sup>. Độ chính xác và độ hồi tưởng của độ đo tương tự cho phương pháp fuzzy-fingerprinting (FF), locality-sensitive hashing (LSH), supershringling (SSh), shingling (Sh), and hashed breakpoint chunking (HBC).

### 2.5.3. Bài toán sửa lỗi chính tả văn bản

#### 2.5.3.1. Khái niệm

Chính tả là sự chuẩn hoá hình thức chữ viết của ngôn ngữ. Đó là một hệ thống các quy tắc về cách viết các âm tiết, từ, các dấu câu, tên riêng, từ nước ngoài...

Những lỗi chính tả phát sinh là do quá trình nhận dạng OCR phát sinh các lỗi chính tả cho từ nhận diện được. Bài toán này gồm ba bước chính là tiền xử lý tập văn bản đầu vào, phát hiện lỗi chính tả và sửa lỗi chính tả.

Lỗi chính tả được chia làm hai loại là non-word và real-word. Lỗi non-word được hiểu là những từ lỗi không tìm thấy trong từ điển. Ví dụ một câu trong tiếng Việt: “Khoa Công Nghệ Thông Tin”, chuỗi “Thông” là từ bị sai lỗi chính tả. Từ “Thông” này không hề xuất hiện trong từ điển Tiếng Việt, từ đúng phải là “Thông”.

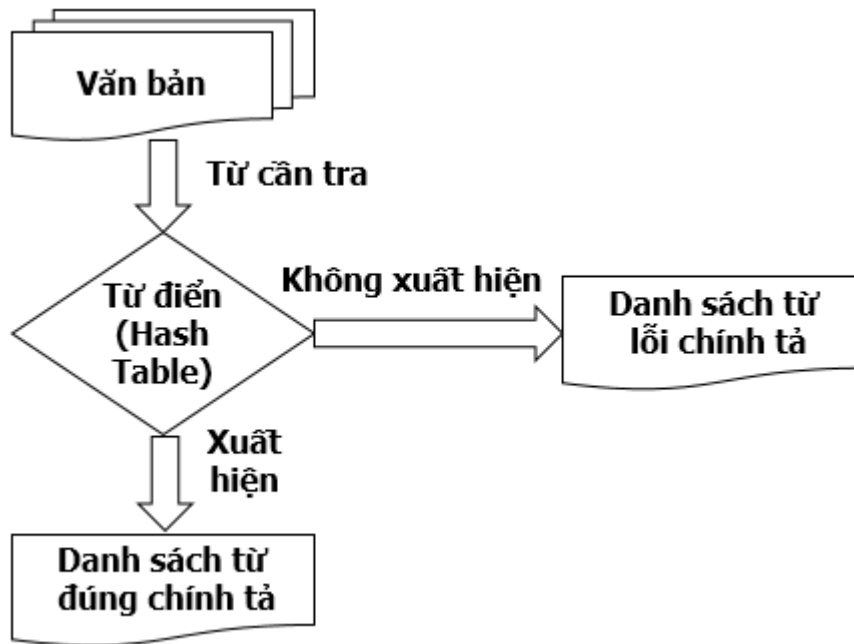
Lỗi real-word là những từ lỗi có trong từ điển nhưng không đúng trong ngữ cảnh của câu. Ví dụ một câu lỗi real-word: “Bài toán sửa lỗi chính tả”, từ “sửa” là từ sai lỗi chính tả, mặc dù từ “sửa” vẫn có trong từ điển. Từ đúng trong trường hợp này phải là “sửa”.

#### 2.5.3.2. Phương pháp tiếp cận

Đối với vấn đề phát hiện lỗi chính tả thì thường có hai phương pháp tiếp cận chính <sup>[17]</sup>.

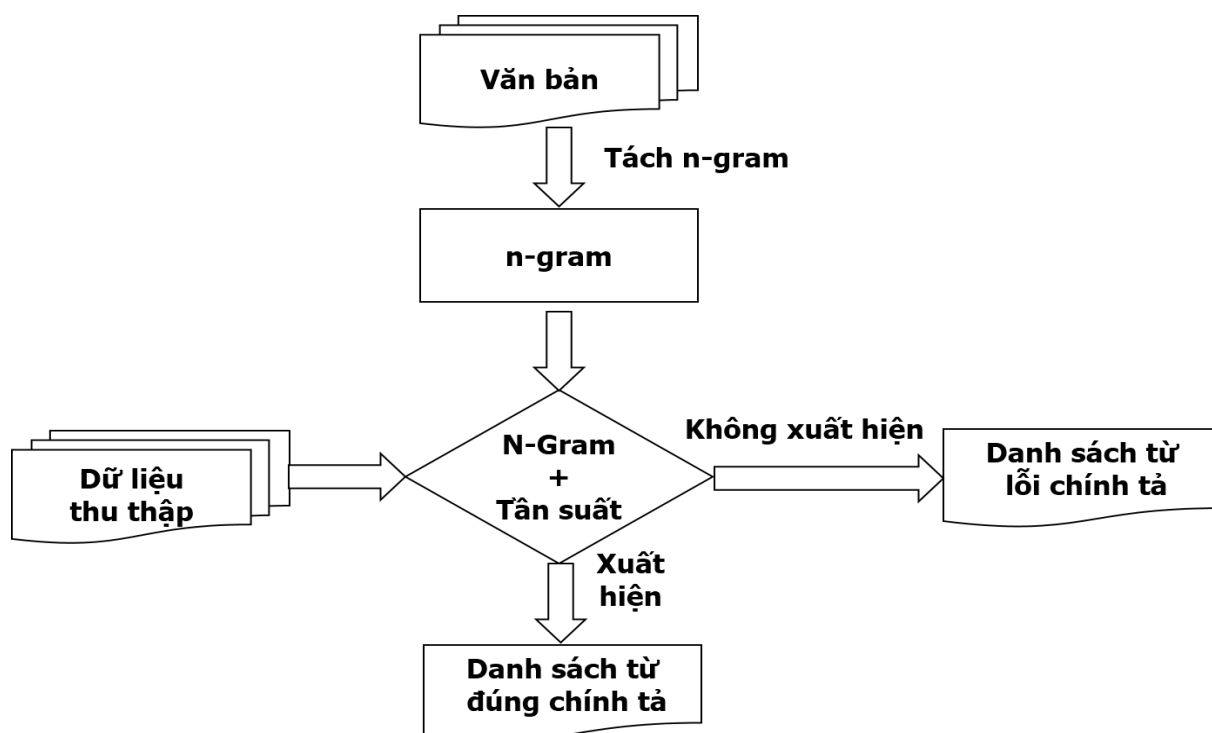
Kỹ thuật tra cứu dùng từ điển: Kỹ thuật đơn giản là kiểm tra sự hiện diện từng từ của văn bản đầu vào. Nếu từ đó có trong từ điển thì từ đó được coi là từ đúng chính tả, ngược lại thì từ đó được coi là lỗi chính tả. Kỹ thuật phổ biến nhất và nhanh chóng để phát hiện từ bị lỗi chính tả. Từ điển được xây dựng bằng

cách sử dụng bảng băm để cải thiện tốc độ tra cứu. Hình 2.9 mô tả quá trình kiểm tra lỗi chính tả bằng kỹ thuật dùng từ điển.



**Hình 2.9.** Kỹ thuật phát hiện lỗi chính tả dựa vào tra cứu từ điển

Kỹ thuật phân tích N-gram: N-gram là một chuỗi con gồm n từ, thường thì là hai, ba hoặc năm từ. Kỹ thuật này thực hiện bằng cách chia văn bản đầu vào thành n-gram tương ứng, đối với mỗi n-gram đầu vào, tìm kiếm trong bảng thống kê n-gram tính trước. Kết hợp thêm tần suất xuất hiện của n-gram trong bảng thống kê để kiểm tra sự tồn tại hoặc mức độ phổ biến của n-gram đầu vào nhằm xác định lỗi chính tả. Hình 2.10 mô tả quá trình kiểm tra lỗi chính tả bằng kỹ thuật sử dụng N-gram.



**Hình 2.10.** Kỹ thuật phát hiện lỗi chính tả dựa vào phân tích N-gram

Sau khi chuỗi kí tự được phát hiện là lỗi chính tả, tác giả cần tìm những từ gợi ý thích hợp trong tập từ đề cử để thay thế cho từ bị lỗi. Dựa vào các nghiên cứu [10], [20], [21] thuật toán tìm ứng viên thay thế phổ biến nhất là dựa vào tính toán khoảng cách chỉnh sửa nhỏ nhất (minimum edit distance).

Khoảng cách chỉnh sửa này lần đầu tiên được nhà khoa học người Nga, tên là Levenshtein đưa ra khái niệm vào năm 1965<sup>[17][20]</sup>.

Khoảng cách chỉnh sửa nhỏ nhất là số lượng tối thiểu của các hoạt động chỉnh sửa (chèn, xóa và thay thế) cần thiết để chuyển một chuỗi thành chuỗi khác. Ví dụ khoảng cách chỉnh sửa nhỏ nhất giữa hai chuỗi “chính” và “chén” là 2, vì phải dùng ít nhất 2 lần biến đổi.

1. chính -> chín (xóa kí tự “h”)
2. chín -> chén (thay kí tự “i” bằng “é”)

Ngoài ra khoảng cách chỉnh sửa nhỏ nhất còn sử dụng để xếp hạng các ứng viên thay thế cho từ bị lỗi. Mục tiêu của việc xếp hạng các ứng viên là đưa ra các từ có khả năng thay thế tốt nhất lên đầu danh sách từ gợi ý thay thế.

Bảng xếp hạng từ thay thế này có thể sử dụng khoảng cách chỉnh sửa nhỏ nhất để đẩy các từ có khoảng cách chỉnh sửa nhỏ nhất lên đầu danh sách gợi ý. Hoặc có thể sử dụng tần suất của từ, những từ có tần suất sử dụng phổ biến hơn (được sử dụng nhiều) sẽ được ưu tiên để thay thế cho từ bị lỗi. Tùy vào mục đích mà có thể áp dụng đồng thời cả hai kỹ thuật sử dụng khoảng cách chỉnh sửa nhỏ nhất và tần suất của từ. Ví dụ trong danh sách các từ gợi ý cùng khoảng

cách chỉnh sửa nhỏ nhất, từ nào có tần suất sử dụng cao hơn sẽ được ưu tiên để thay thế cho từ bị lỗi.

## **2.6. Bài toán đánh chỉ mục và tìm kiếm**

### **2.6.1. Khái niệm**

Kết thúc quá trình xử lý video nguồn, kết quả thu được là một tệp văn bản tương ứng đối với nội dung của video bài giảng đã được trích xuất. Các văn bản ở dạng thô cần được chuyển sang một dạng biểu diễn nào đó để xử lý. Quá trình đó là lập chỉ mục cho tệp văn bản để hỗ trợ việc tìm kiếm thông tin của người dùng.

Lập chỉ mục tài liệu là công việc sắp xếp tài liệu nhằm đáp ứng nhanh chóng yêu cầu tìm kiếm thông tin của người sử dụng. Quá trình lập chỉ mục được hiểu là giai đoạn phân tích tệp văn bản đã xử lý và thu được để xác định các chỉ mục biểu diễn nội dung của tệp văn bản này. Hệ thống chỉ mục thu được là danh sách các từ khóa, chỉ rõ các từ khóa nào xuất hiện ở video nào, địa chỉ nào. Các phương pháp lập chỉ mục đóng vai trò quan trọng trong việc xây dựng một hệ thống tìm kiếm thông tin hiệu quả.

### **2.6.2. Phương pháp tiếp cận**

Có nhiều công cụ để thực hiện lập chỉ mục cho tài liệu như Apache Solr, Lucence, Sphinx. Nhưng đối với bài toán lập chỉ mục tài liệu tác giả sử dụng công cụ Elasticsearch.

Elasticsearch là một máy chủ tìm kiếm dựa trên Lucence. Nó là công cụ mã nguồn mở cho phép tìm kiếm toàn văn (full-text search). Phiên bản đầu tiên của Elasticsearch được Shay Banon phát hành vào tháng 2 năm 2010. Hiện nay, theo đánh giá của DB-Engines thì Elasticsearch là công cụ tìm kiếm doanh nghiệp phổ biến nhất, tiếp theo là Apache Solr, cũng dựa trên Lucene.

17 systems in ranking, April 2016

Rank			DBMS	Database Model	Score		
Apr 2016	Mar 2016	Apr 2015			Apr 2016	Mar 2016	Apr 2015
1.	1.	↑ 2.	Elasticsearch	Search engine	82.58	+2.41	+17.92
2.	2.	↓ 1.	Solr	Search engine	66.02	-3.35	-15.98
3.	3.	3.	Splunk	Search engine	42.35	-1.38	+4.32
4.	4.	4.	MarkLogic	Multi-model	9.12	-0.25	-1.09
5.	5.	5.	Sphinx	Search engine	8.48	-0.24	-1.10
6.	6.	↑ 7.	Google Search Appliance	Search engine	3.57	-0.21	-0.12
7.	7.	↑ 8.	Amazon CloudSearch	Search engine	2.36	-0.03	+0.58
8.	8.	↑ 9.	Microsoft Azure Search	Search engine	1.25	-0.03	+0.53
9.	9.	↑ 10.	Xapian	Search engine	0.48	-0.02	-0.07
10.	10.	↑ 11.	Indica	Search engine	0.32	-0.02	+0.03
11.	↑ 12.	↑ 13.	Crate.IO	Multi-model	0.16	+0.02	+0.01
12.	↓ 11.	↑ 14.	SearchBlox	Search engine	0.16	-0.01	+0.07
13.	13.	↑ 16.	Srch <sup>2</sup>	Search engine	0.05	+0.02	+0.05
14.	14.	↑ 15.	DBSight	Search engine	0.02	-0.00	-0.03
15.	15.	↑ 16.	Exorbyte	Search engine	0.01	+0.01	+0.01
16.	↓ 15.	↓ 12.	Compass	Search engine	0.00	±0.00	-0.21
16.	↓ 15.		searchxml	Multi-model	0.00	±0.00	

**Hình 2.11.** Thứ hạng của 17 công cụ tìm kiếm. Nguồn <http://db-engines.com>

Theo nghiên cứu [18] thì Elasticsearch là công cụ có nhiều ưu điểm vượt trội hơn các công cụ tìm kiếm khác như:

- Không cần cấu hình phức tạp Elasticsearch sẽ tự động phát hiện các kiểu dữ liệu cơ bản mà ta đưa vào. Do đó ta chỉ cần tiến hành index tài liệu ngay sau khi cài đặt xong.

- Elasticsearch hỗ trợ thêm, xoá, sửa chỉ mục thông qua các phương thức HTTP như GET, POST, DELETE và PUT, hỗ trợ tham số dưới dạng JSON thay vì chỉ là GET params.

- Cài đặt và sử dụng dễ dàng mà không cần cài thêm bất cứ ứng dụng nào khác.

- Tìm kiếm gần như thời gian thực (real-time).

### 2.6.3. Kiến trúc của Elasticsearch

- Cluster: Một cluster là một tập hợp của một hoặc nhiều nodes (servers) mà cùng nhau nắm giữ toàn bộ dữ liệu và cung cấp các chỉ mục và khả năng tìm kiếm qua tất cả các nodes. Một cluster được định danh bởi một tên duy nhất, mặc định là “elasticsearch”.



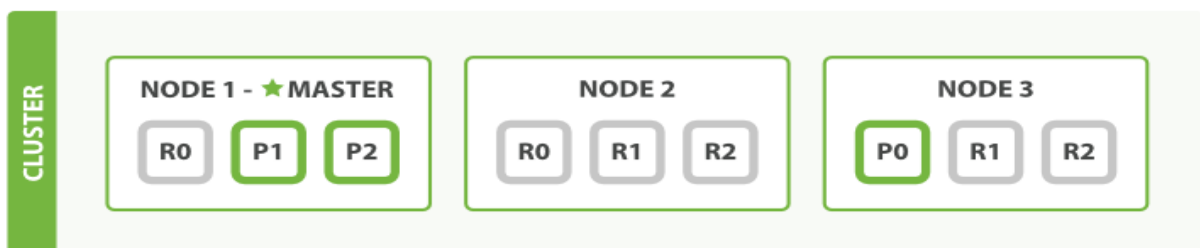
- Node: Một node là một server riêng mà là một phần trong cluster. Lưu trữ dữ liệu và tham gia vào việc lập chỉ mục và tìm kiếm. Một node cũng được định danh bởi một tên riêng biệt, mặc định được khởi tạo cho node khi khởi động. Mặc định mỗi node được thiết lập để liên kết với một cluster nhất định bởi cluster name. Trong một cluster thì không hạn chế số lượng các node.

- Index: Một index là một tập hợp các documents có đặc điểm chung. Ví dụ, ta có một index cho dữ liệu bài giảng, một index khác cho sản phẩm, hoặc một index khác để sắp xếp dữ liệu. Một index được định danh bởi tên riêng, tên riêng viết thường, không viết hoa. Tên này dùng để liên hệ đến index khi thực hiện tạo lập chỉ mục, tìm kiếm, cập nhật, xóa document trong đó. Trong một cluster có thể chứa nhiều indexes.

- Type: Trong một index bạn có thể định nghĩa một hoặc nhiều types. Một type là một mục/phân vùng có nghĩa trong index của bạn. Một type được định nghĩa cho document bao gồm một số trường.

- Document: Một document là một đơn vị thông tin cơ bản để lập được index. Ví dụ có một document cho một khách hàng, document khác cho một sản phẩm hay cho một đơn đặt hàng khác. Document này được định dạng theo một type nào đó. Trong mỗi index/type, có thể lưu trữ nhiều documents. Chú ý một document cần được gán vào một type bên trong index để có thể được lập index.

- Shard & Replicas: Mỗi index có thể được chia thành nhiều shards. Mỗi index cũng có thể được sao lưu nhiều lần. Mỗi khi được nhân bản, mỗi index sẽ có những shards chính và những shards nhân bản (sao chép từ shards chính). Số lượng shards và replicas có thể được khai báo khi tạo index. Sau khi index được tạo, có thể thay đổi số lượng bản sao bất cứ lúc nào nhưng không thể thay đổi số shards.



**Hình 2.12.** Kiến trúc cluster-node-shard của Elasticsearch

Hình 2.12 biểu diễn một mô hình đơn giản của Elasticsearch. Dữ liệu được lưu trữ ở cluster với ba nodes trong đó Node 1 là master. Có ba primary shards, hai trong số đó được đặt ở Node 1, còn lại ở Node 3. Mỗi primary shard (P0, P1, P2) có 2 replica shard (R1, R2) (ví dụ primary shard P0 ở Node 3 thì có replica shard R0 ở Node 1 và một shard nữa ở Node 2). Việc sắp đặt vị trí primary

shard là ngẫu nhiên, còn các replica shard luôn được đảm bảo là không nằm cùng node với primary shard.

## **2.7. Kết luận**

Kết thúc chương này, tác giả đã trình bày khái quát các bài toán cần giải quyết trong nội dung luận văn này. Các phương pháp tiếp cận để giải quyết vấn đề. Tiếp theo, chương ba tác giả xin trình bày chi tiết về các giải pháp kỹ thuật tiến hành của tác giả để thực hiện các bài toán đã nêu trong chương hai.

## CHƯƠNG 3: KỸ THUẬT ĐỂ GIẢI QUYẾT CÁC BÀI TOÁN TRONG KHUÔN KHỔ LUẬN VĂN

Nội dung chương này sẽ nghiên cứu các cách giải pháp kỹ thuật tiến hành cài đặt các thuật toán để giải quyết các bài toán đã nêu trong mục 2.

### 3.1. Bài toán phân đoạn video thành định dạng ảnh

#### 3.1.1. Phát biểu bài toán

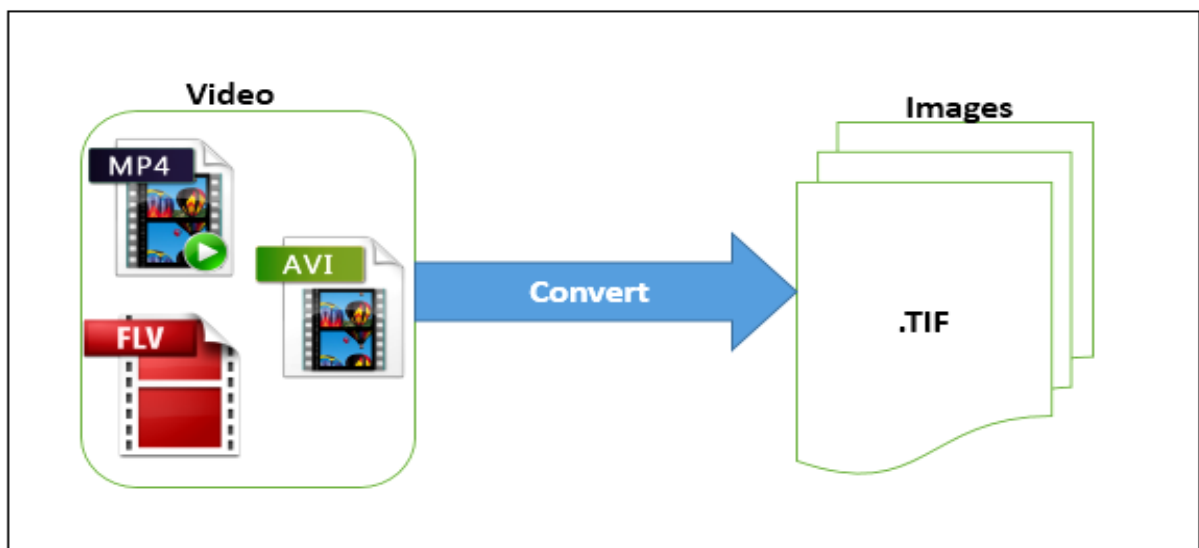
Bài toán đầu tiên được đề cập trong luận văn này là chuyển đổi video bài giảng đầu vào thành tập các khung hình rời rạc liên tiếp nhau theo giờ gian. Hình 3.1 mô tả quá trình biến đổi video bài giảng thành tập ảnh. Bài toán được phát biểu như sau:

Đầu vào:

- Video bài giảng nguồn.

Đầu ra:

- Tập các khung hình được trích xuất từ video nguồn.



**Hình 3.1.** Mô tả quá trình biến đổi video nguồn thành dạng ảnh

#### 3.1.2. Giải pháp thực hiện

Sau khi cài đặt phần mềm Ffmpeg, sử dụng dòng lệnh “*ffmpeg -i lecture001.mp4 -r 1 %d.tif*” trong đó:

- *i* là video đầu vào với đường dẫn của tệp tin video. Trong ví dụ này video được định dạng là .mp4 với tên tệp tin là lecture001.

- *r* là số khung hình trên giây.

- *%d.tif* là định dạng tên tệp tin hình ảnh để lưu với tên là số nguyên và định dạng là .tif. Ví dụ 1.tif, 2.tif, 3.tif...

Trong quá trình thực nghiệm để có kết quả tốt nhất phục vụ cho những quá trình xử lý tiếp theo tác giả đề xuất:

- Định dạng của ảnh thu được là .TIFF - Tagged Image Format File. Định dạng ảnh chuẩn dành cho in ấn, và dù có bị nén hay không thì ảnh không bị mất bất kì dữ liệu ảnh nào. Chất lượng ảnh thu được rất tốt và được khuyến nghị để áp dụng cho các máy nhận diện kí tự. Một quá trình tiếp theo được trình bày ở mục 3.2.

- Sử dụng số FPS là 1 (một khung hình một giây).

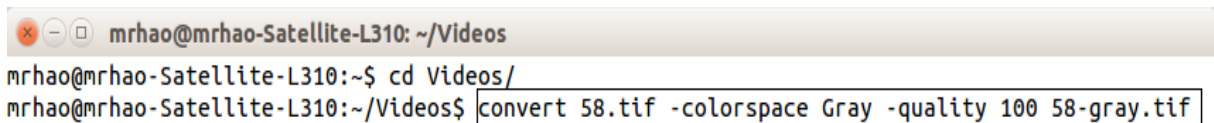
### 3.2. Bài toán trích xuất văn bản

Như đã trình bày ở chương 2, trong bài toán trích xuất văn bản để lấy nội dung của video bài giảng. Bài toán này được chia nhỏ thành ba bài toán con:

- Bài toán nhận dạng kí tự quang học bằng công cụ Tesseract-OCR.
- Bài toán xử lý trùng lặp văn bản bằng kĩ thuật Shingling.
- Bài toán sửa lỗi chính tả Tiếng Việt.

#### 3.2.1. Bài toán nhận dạng kí tự quang học bằng công cụ Tesseract-OCR

Theo nghiên cứu [19] thì ảnh màu thường cho kết quả nhận diện chính xác kém hơn so với ảnh đa cấp xám. Chính vì thế trước khi nhận dạng OCR thì ảnh màu sẽ được tác giả chuyển đổi thành ảnh đa cấp xám. Hình 3.2 mô tả câu lệnh chuyển đổi ảnh màu thành ảnh đa cấp xám tương ứng.

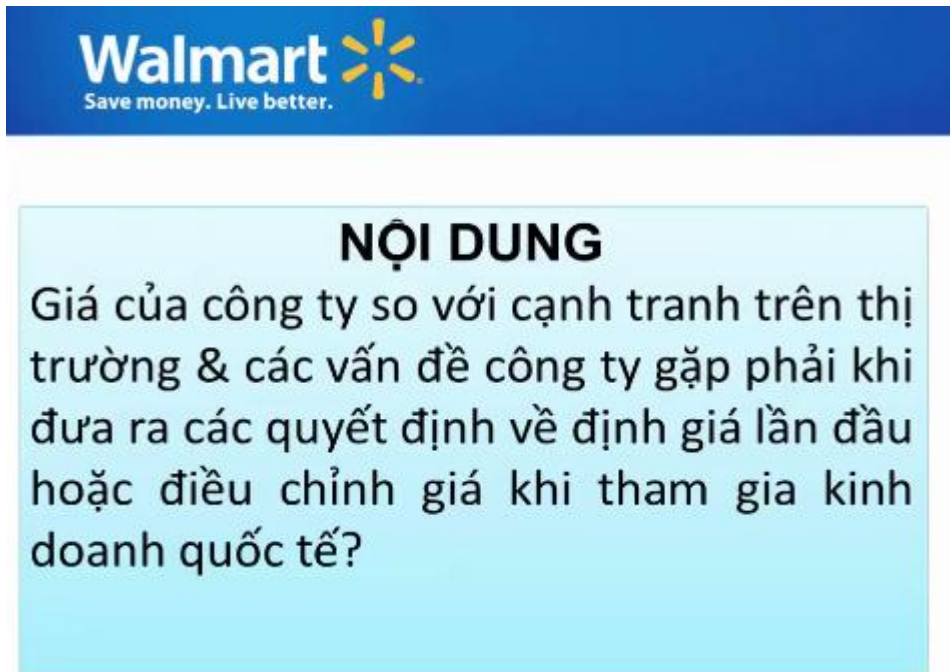


```

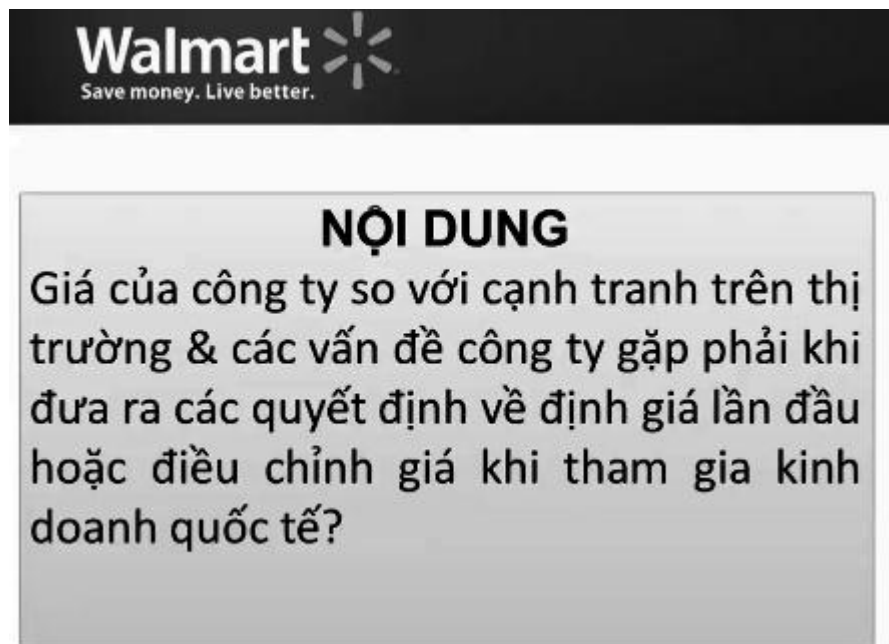
mrhao@mrhao-Satellite-L310: ~/Videos
mrhao@mrhao-Satellite-L310:~$ cd Videos/
mrhao@mrhao-Satellite-L310:~/Videos$ convert 58.tif -colorspace Gray -quality 100 58-gray.tif
  
```

---

**Hình 3.2.** Chuyển đổi ảnh màu thành ảnh đa cấp xám



**Hình 3.3.** Ảnh màu



**Hình 3.4.** Ảnh đa cấp xám

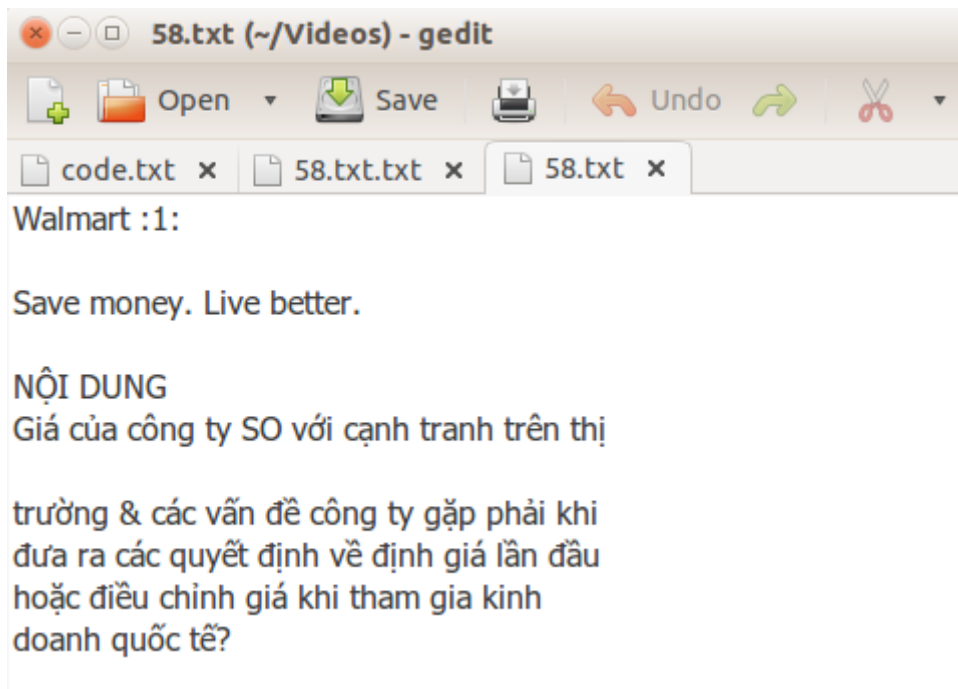
Bây giờ tác giả sẽ thực hiện OCR trên ảnh đa cấp xám ở Hình 3.4. Câu lệnh thực hiện quá trình OCR được mô tả trong hình 3.5. Câu lệnh Tesseract gồm ba tham số: Tham số thứ nhất là tên tệp tin hình ảnh, tham số thứ hai là tên tệp tin kết quả trích xuất văn bản từ ảnh đầu vào và tham số thứ ba là ngôn ngữ dùng để trích xuất ở đây là Tiếng Việt. Tên tệp tin kết quả trích xuất được lưu trữ với phần mở rộng là .txt. Kết thúc quá trình thu được tệp tin có nội dung ở hình 3.7.

```

mrhao@mrhao-Satellite-L310: ~/Videos
mrhao@mrhao-Satellite-L310:~$ cd Videos/
mrhao@mrhao-Satellite-L310:~/Videos$ tesseract 58-gray.tif 58 -l vie
Tesseract Open Source OCR Engine v3.03 with Leptonica
mrhao@mrhao-Satellite-L310:~/Videos$

```

**Hình 3.5.** Quá trình OCR ảnh trong hình 3.4 bằng Tesseract-OCR



```

58.txt (~/.Videos) - gedit
Open Save Undo
code.txt x 58.txt.txt x 58.txt x
Walmart :1:
Save money. Live better.
NỘI DUNG
Giá của công ty SO với cạnh tranh trên thị
trường & các vấn đề công ty gặp phải khi
đưa ra các quyết định về định giá lần đầu
hoặc điều chỉnh giá khi tham gia kinh
doanh quốc tế?

```

**Hình 3.6.** Kết quả sau khi hoàn thành OCR bằng Tesseract-OCR

Như đã trình bày ở mục trước, số lượng ảnh thu được trong quá trình chuyển đổi video thành dạng ảnh là lớn. Để thực hiện tự động và đồng loạt OCR tất cả tệp tin ảnh thì câu lệnh OCR được thay đổi với nội dung được nêu ở hình 3.7.

```

mrhao@mrhao-Satellite-L310: ~/Videos
mrhao@mrhao-Satellite-L310:~$ cd Videos/
mrhao@mrhao-Satellite-L310:~/Videos$ for i in *.tif ;
do tesseract $i $i -l vie; done;
Tesseract Open Source OCR Engine v3.03 with Leptonica
Tesseract Open Source OCR Engine v3.03 with Leptonica
Tesseract Open Source OCR Engine v3.03 with Leptonica
Tesseract Open Source OCR Engine v3.03 with Leptonica
Tesseract Open Source OCR Engine v3.03 with Leptonica

```

**Hình 3.7.** Thực hiện OCR tất cả ảnh trong thư mục bằng Tesseract-OCR

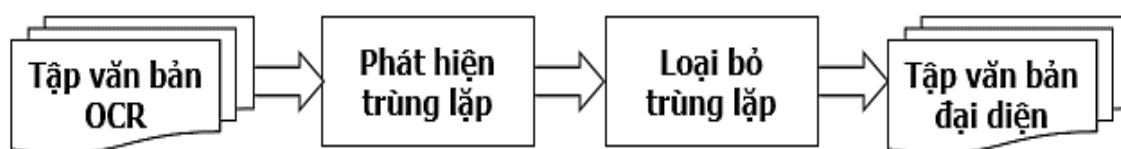
Theo các kết quả nghiên cứu [19] và kết quả tác giả thực hiện thì đối với ảnh màu thì độ chính xác trung bình khoảng 61%, và đối với ảnh đa cấp xám thì

là khoảng 70%. Điều đó chứng tỏ việc chuyển đổi ảnh màu thành ảnh đa cấp xám cho chất lượng nhận dạng tốt hơn và hữu ích hơn.

### 3.2.2. Bài toán xử lý trùng lặp văn bản bằng kỹ thuật Shingling

#### 3.2.2.1. Phát biểu bài toán

Mục tiêu của quá trình này sẽ là phát hiện và loại bỏ những tệp văn bản có nội dung gần trùng nhau (các tệp được trích xuất từ một slide). Quá trình này trải qua hai bước được trình bày trong hình



**Hình 3.8.** Quá trình xử lý trùng lặp văn bản

Bước 1- Phát hiện sự trùng lặp nội dung của hai tệp văn bản: Chính là quá trình tìm sự tương đồng giữa hai tệp văn bản dựa trên một giá trị ngưỡng cho trước.

Bước 2 - Loại bỏ một tệp văn bản có nội dung trùng lặp: Các văn bản được xác định là trùng lặp sẽ loại bỏ và giữ lại một văn bản là văn bản đại diện để tiếp tục quá trình lặp.

Trong bài toán này, tác giả cần giải quyết hai vấn đề là: phát hiện sự trùng lặp của hai văn bản và lấy văn bản đại diện cho tập văn bản trùng lặp.

Đầu vào:

- Tập các văn bản được trích xuất từ OCR.

Đầu ra:

- Phát hiện trùng lặp văn bản.
- Tập văn bản đại diện cho video bài giảng.

#### 3.2.2.2. Giải thuật Shingling

Phương pháp Shingling là một trong các phương pháp NDD sớm nhất, kỹ thuật này nhằm để ước lượng độ tương tự giữa các cặp tài liệu được trình bày năm 1997 bởi Broder và cộng sự. Thuật toán Shingling được trình bày như sau:

Cho một số nguyên dương  $k$  và một chuỗi liên tục các thuật ngữ trong tài liệu  $D$ .  $k$ -singles là tập  $k$  từ liên tục nhau trong  $D$ . Ví dụ: Văn bản có nội dung

“trường đại học công nghệ thuộc đại học quốc gia hà nội”. 5-shingles của tài liệu D vừa cho đó là “trường đại học công nghệ”, đại học công nghệ đại”, “học công nghệ đại học”, “công nghệ đại học quốc”, “nghệ đại học quốc gia”. Bằng trực giác chúng ta có thể nhận thấy là để kết luận hai văn bản gần trùng nhau nếu hai tập shingles được tạo từ chúng là gần như bằng nhau.

Gọi tập  $S(d_j)$  là tập shingles của tài liệu  $d_j$ . Sự tương đồng của hai tài liệu được đo bằng cách sử dụng hệ số Jaccard giữa các vector shingles. Giả sử với hai tập  $d_1$  và  $d_2$  thì hệ số Jaccard được tính theo công thức hình 3.9.

$$J(S(d_1), S(d_2)) = \frac{|S(d_1) \cap S(d_2)|}{|S(d_1) \cup S(d_2)|}$$

**Hình 3.9.** Hệ số Jaccard của tài liệu  $d_1$  và  $d_2$

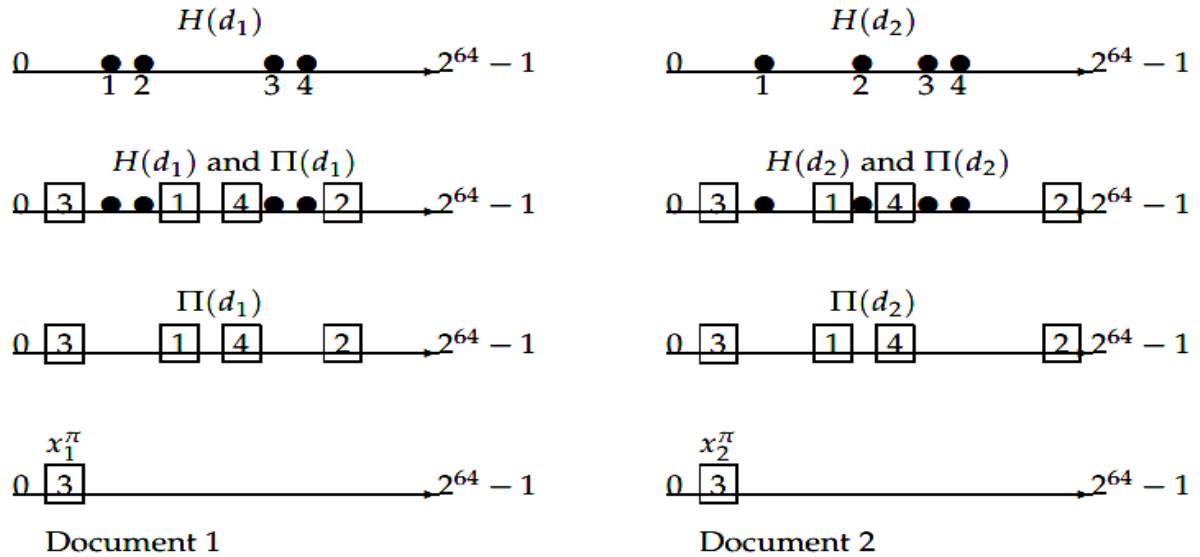
Hệ số Jaccard đưa ra kết quả là giá trị trong đoạn  $[0-1]$ , giá trị càng lớn thì có nghĩa độ tương đồng của hai văn bản càng cao. Trong thực nghiệm, tác giả sẽ sử dụng một giá trị ngưỡng để kết luận về độ tương đồng của văn bản.

Mặc dù độ đo Jaccard cho kết quả tốt nhưng với số lượng lớn các tài liệu thì việc tính toán từng cặp văn bản sẽ là một thách thức. Số lượng phép tính tăng tuyến tính theo số lượng shingle có trong tài liệu. Nếu số lượng shingle dài (hệ số  $k$  lớn) thì những thay đổi nhỏ trong các tài liệu sẽ ảnh hưởng đến độ Jaccard, ngược lại nếu kích thước shingle ngắn thì dẫn đến thời gian tính toán tăng lên đáng kể. Việc lựa chọn  $k$  cho kết quả tốt thường nằm trong khoảng  $[4-10]$ <sup>[8]</sup>.

Để tránh việc phải tính toán Jaccard với từng cặp tài liệu, người ta thường áp dụng thêm kỹ thuật băm. Đầu tiên, sẽ ánh xạ từng shingle vào một giá trị băm được lưu trữ bằng 64 bit. Hàm  $H(d_j)$  là giá trị băm tương ứng của tập  $S(d_j)$ . Gọi  $\Pi$  là tập các hoán vị của tập các giá trị băm  $H()$ . Kí hiệu là  $\Pi(d_j)$  là tập hợp các giá trị băm hoán vị trong  $H(d_j)$ ; do đó với mỗi  $h \in H(d_j)$  thì tồn tại một giá trị tương ứng  $\pi(h) \in \Pi(d_j)$ .

Người ta cũng chứng minh được rằng nếu gọi  $x_1^\pi, x_2^\pi$  là giá trị nhỏ nhất thì  $J(S(d_1), S(d_2)) = P(x_1^\pi = x_2^\pi)$ . Hình 3.9 mô tả cách biểu diễn shingle của hai tài liệu.





**Hình 3.10<sup>[4]</sup>.** Bốn quá trình tính toán shingle của hai tài liệu.

Bước đầu tiên (dòng trên cùng), chúng ta áp dụng hàm băm 64 bit cho mỗi shingle từ hai văn bản để tạo thành  $H(d_1)$  và  $H(d_2)$  (chấm hình tròn). Tiếp theo, chúng ta áp dụng hoán vị ngẫu nhiên  $\Pi$  của  $H(d_1)$  và  $H(d_2)$ , tạo thành  $\Pi(d_1)$  và  $\Pi(d_2)$  (hình vuông). Bước ba, chỉ hiện thị  $\Pi(d_1)$  and  $\Pi(d_2)$ , và dòng cuối cùng biểu thị giá trị nhỏ nhất  $x_1^\pi$  và  $x_2^\pi$  cho mỗi tài liệu.

Chúng ta cùng xét ví dụ dưới đây để hiểu rõ phương pháp này.

Tập A: Tôi thích màu xanh và đỏ.

Tập B: Tôi thích chúng, màu xanh và đỏ.

Tập 2-shingles

$S(A) = \{[Tôi thích] [thích màu] [màu xanh] [xanh và] [và đỏ]\}$

$S(B) = \{[Tôi thích] [thích chúng] [chúng màu] [màu xanh] [xanh và] [và đỏ]\}$

$S(A) \cap S(B) = \{[Tôi thích] [màu xanh] [xanh và] [và đỏ]\} = 4.$

$S(A) \cup S(B) = \{[Tôi thích] [thích màu] [màu xanh] [xanh và] [và đỏ] [thích chúng] [chúng màu]\} = 7.$

Hệ số Jaccard được tính theo công thức:

$$J(S(A), S(B)) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|} = \frac{4}{7} \approx 0,57$$

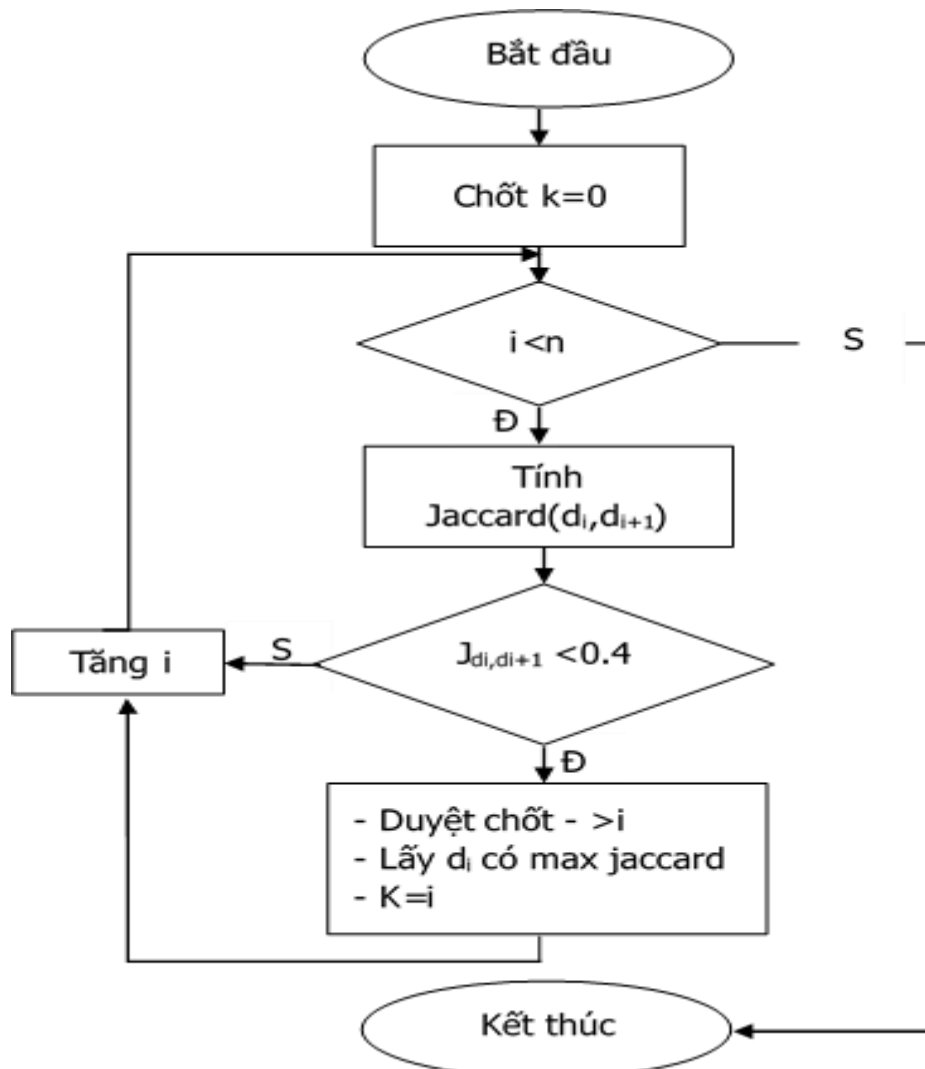
### 3.2.2.3. Kỹ thuật tiên hành

Dựa trên các cơ sở của phương pháp shingling, tác giả đã xác định và kết luận được hai tệp văn bản bất kỳ có phải là gần trùng lặp nhau hay không, căn cứ vào một giá trị ngưỡng của độ đo Jaccard trong hình 3.13. Bài toán tiếp theo trong nội dung này là xác định được tệp các văn bản đại diện cho video bài giảng.

Đầu vào: Cho tập  $D$  là tập tất cả văn bản được trích xuất OCR từ video, giá trị  $d_1, d_2, \dots, d_n$  là các văn bản được thuộc tập  $D$ .

Đầu ra: Tập  $D'$  là tập văn bản đại diện cho tập  $D$ .

Giải thuật



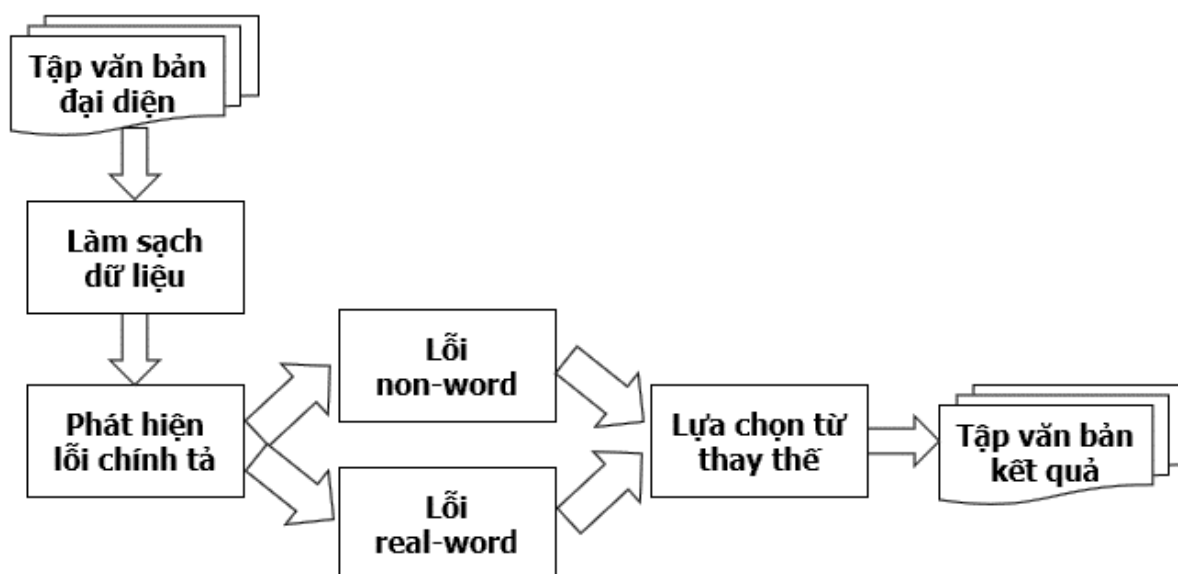
**Hình 3.11.** Sơ đồ khối quá trình trích xuất tập văn bản đại diện

Kết quả của quá trình này tác giả thu được tập các văn bản đại diện cho video bài giảng đầu vào. Đây là các văn bản được trích xuất từ nội dung các slide khác nhau trong video bài giảng đầu vào.

### 3.2.3. Bài toán sửa lỗi chính tả văn bản tiếng Việt

#### 3.2.3.1. Phát biểu bài toán

Tập văn bản đại diện cần được xử lý lỗi chính tả, lỗi này phát sinh do quá trình nhận dạng kí tự quang học. Chính vì thế quá trình này sẽ phát hiện và sửa các lỗi chính tả của tập văn bản đại diện. Hình 3.12 mô tả các bước để thực hiện phát hiện và sửa lỗi chính tả văn bản.



**Hình 3.12.** Quá trình phát hiện và sửa lỗi chính tả văn bản

- Bước 1: Đây là bước đầu tiên trong quá trình phát hiện và sửa lỗi chính tả. Dữ liệu đầu vào sau khi được nạp cần được loại bỏ một số kí tự dư thừa (không có ý nghĩa trong từ) như các khoảng trắng, các dấu chấm, hoặc các kí tự đặc biệt...

- Bước 2: Phát hiện lỗi chính tả: Để phát hiện lỗi chính tả chúng ta cần một số khái niệm về các loại lỗi chính tả. Có nhiều cách, tiêu chí để phân loại nhưng trong khuôn khổ chương trình phát hiện lỗi chính tả ở mức từ thì lỗi chính tả được chia làm hai loại là lỗi non-word và lỗi real-word:

+ Lỗi non-word: là lỗi tạo ra từ sai, từ đó hoàn toàn không có trong từ điển từ vựng hoặc một số từ điển tên riêng, từ điển viết tắt, từ điển vay mượn,... Đây là loại lỗi dễ phát hiện.

+ Lỗi real-word: là lỗi chính tả mà từ đó có trong từ điển nhưng sử dụng từ sai. Nếu không dựa vào ngữ cảnh xung quanh thì không thể xác định được đó có phải là lỗi chính tả hay không. Đây là loại lỗi khó phát hiện và xử lý.

- Bước 3: Dựa vào từng loại lỗi để lựa chọn từ thay thế cho từ bị lỗi.

### 3.2.3.2. Làm sạch dữ liệu trước khi sửa lỗi chính tả

Trước khi phát hiện và sửa lỗi chính tả, dữ liệu đầu vào cần được xử lý như xóa kí tự khoảng trắng thừa, loại bỏ các kí tự đặc biệt (dấu chấm, phẩy, chấm than, chấm hỏi...), và các chuỗi đặc biệt như địa chỉ trang web, email, các dữ liệu số, ngày tháng... Do những kí tự này không liên quan đến nghĩa của từ và hạn chế được các lỗi non-word. Tác giả thực hiện việc làm sạch dữ liệu ba bước:

- Bước 1: Loại bỏ các kí tự khoảng trắng thừa ở đầu, giữa, và cuối câu. Ví dụ “bài giảng ” sẽ được thay bằng “bài giảng”.

- Bước 2: Bỏ qua các chuỗi là địa chỉ email, địa chỉ website.
- Bước 3: Loại bỏ các kí tự đặc biệt, các dấu chấm, kí tự số, ngày tháng...

### 3.2.3.3. Kỹ thuật sửa lỗi chính tả dạng non-word

Đối với những lỗi chính tả dạng non-word thì đã được nghiên cứu rất rộng rãi, phổ biến. Các thuật toán để phát hiện và gợi ý từ chỉnh sửa cho các lỗi dạng non-word đã được đề xuất như trong các nghiên cứu [17] [11] [12]. Các thuật toán này được gọi là spell-checker và được tích hợp vào trong nhiều phần mềm xử lý văn bản khác nhau như Microsoft Word, LibreOffice Writer, Ispell, Aspell...

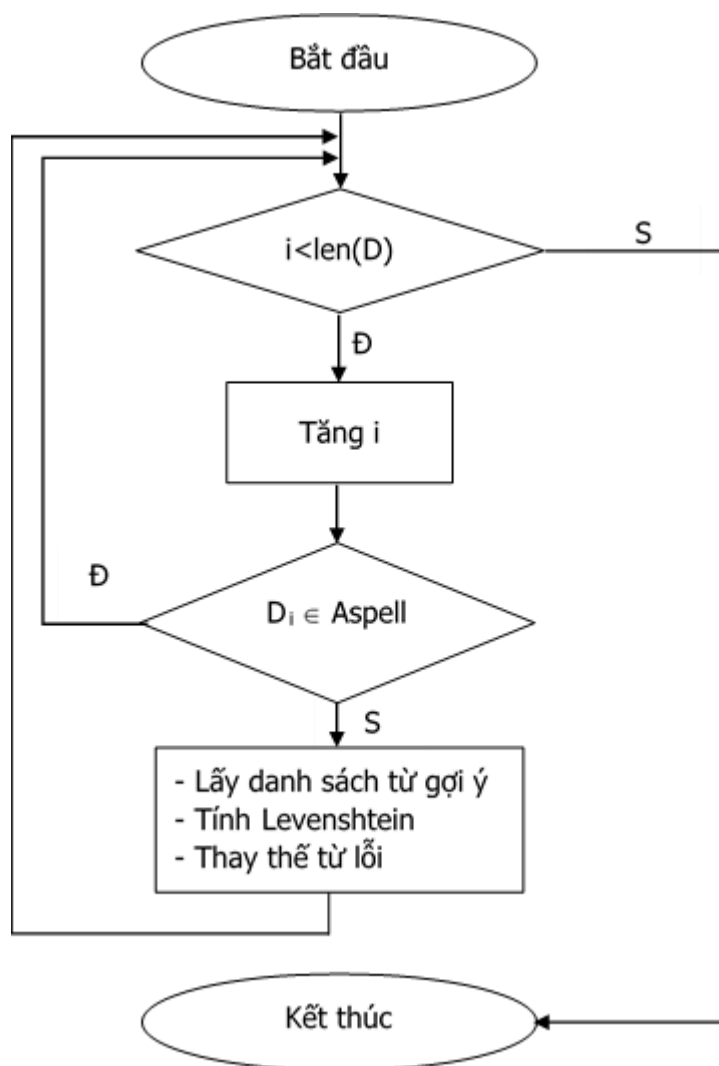
Trong luận văn này tác giả sẽ sử dụng công cụ mã nguồn mở Aspell để cài đặt chương trình sửa lỗi chính tả đối với dạng lỗi non-word.

GNU Aspell là một spell-checker mã nguồn mở và miễn phí. Aspell có thể sử dụng như một thư viện hoặc là một spell-checker độc lập. Các tính năng của GNU Aspell bao gồm:

- Hỗ trợ kiểm tra các văn bản định dạng UTF-8 mà không cần thêm từ điển đặc biệt nào khác.
- Hỗ trợ kiểm tra chính tả nhiều loại ngôn ngữ, trong đó ngôn ngữ Tiếng Việt. Tính năng này rất quan trọng bởi vì nội dung văn bản trong phạm vi luận văn này là văn bản Tiếng Việt.
- Có thể tùy chỉnh từ điển, bổ sung các ngoại lệ hoặc cập nhật thêm các từ mới cho từ điển.

GNU Aspell được phát triển đến năm 2004, đến thời điểm hiện tại thì Aspell không có bản nâng cấp nào thêm. Chính vì thế tác giả đã sử dụng một từ điển mới cho ngôn ngữ Tiếng Việt được cập nhật năm 2014.

Để giao tiếp và làm việc với GNU Aspell trong mục đích lập trình, tác giả sử dụng thêm một gói thư viện là Pspell. Pspell là công cụ cung cấp giao diện để làm việc với từ điển của Aspell qua một số hàm đã được khai báo. Thuật toán phát hiện lỗi và sửa lỗi chính tả dạng non-word được tác giả mô tả bởi sơ đồ khối như hình 3.13.



**Hình 3.13.** Sơ đồ khôi phục lỗi chính tả sử dụng từ điển Aspell

Kỹ thuật sửa lỗi chính tả dùng GNU Aspell chỉ áp dụng được với những lỗi dạng non-word. Vì kỹ thuật này sẽ kiểm tra từng từ và không quan tâm đến vị trí của từ đó so với các từ xung quanh. Vì vậy, để nâng cao hiệu quả của việc kiểm tra lỗi chính tả văn bản, tác giả áp dụng kỹ thuật N-gram.

#### 3.2.3.4. Kỹ thuật sửa lỗi chính tả dạng real-word

Lỗi chính tả dạng real-word thì phức tạp và khó hơn non-word, do những lỗi này thường làm nhập nhằng cú pháp và ý nghĩa của câu. Việc tự động phân tích cú pháp/ngữ nghĩa của một câu đúng là nhiệm vụ khó khăn và nhiệm vụ phân tích những câu sai gần như là không thể trong nhiều trường hợp. Ví dụ dưới đây cho thấy ngôn ngữ Tiếng Việt sự đa dạng và phong phú của ngữ pháp Tiếng Việt.

Câu được cho là: “Ông già đi nhanh quá”. Đây là một câu hoàn toàn đúng về ngữ pháp và các từ hoàn toàn có trong từ điển. Nhưng lại có sự nhập nhằng

giữa ý nghĩa câu trên. Câu trên có thể tách thành hai câu “/Ông/ già đi /nhanh / quá/” hoặc “Ông già/ đi / nhanh /quá”.

Các nghiên cứu [11] [12] [20] cũng đã chỉ ra rằng, các hệ thống phát hiện và sửa lỗi chính tả văn bản có độ chính xác xấp xỉ khoảng 50% cho tất cả các loại lỗi. Trong đó thì 25% - 40% trong tất cả loại lỗi này là lỗi real-word, chính vì thế việc nghiên cứu phát hiện và sửa loại lỗi này là hữu ích.

Do đặc trưng ngôn ngữ Tiếng Việt là gồm các từ đơn ghép lại với nhau. Vì vậy, đề xuất của tác giả là sử dụng kỹ thuật 2-gram để sửa các lỗi chính tả dạng real-word. Nghĩa là từ được kiểm tra sẽ xem xét kết hợp cả hai hàng xóm bên trái và bên phải của nó. Dưới đây là mô tả về kỹ thuật kiểm tra và sửa lỗi chính tả dùng bigram.

Tập đề cử cho từ được kiểm tra ( $W$ ) là tập các từ trong từ vựng mà có thể sinh ra  $W$  bằng cách một thao tác chỉnh sửa. Tập đề cử có thể được biểu diễn dạng

$$C(W^i) = \{W_1^i, W_2^i, \dots, W_j^i, \dots, W_{k_j}^i\}$$

Trong đó:  $W^i$  là từ thứ  $i$  trong câu cần kiểm tra và  $k_j$  là số phần tử trong  $C(W^i)$ . Bây giờ tập bigram trái và bigram phải của mỗi từ trong  $C(W^i)$  sẽ có dạng như sau:

Bigram trái:  $W^{i-1}W_j^i$

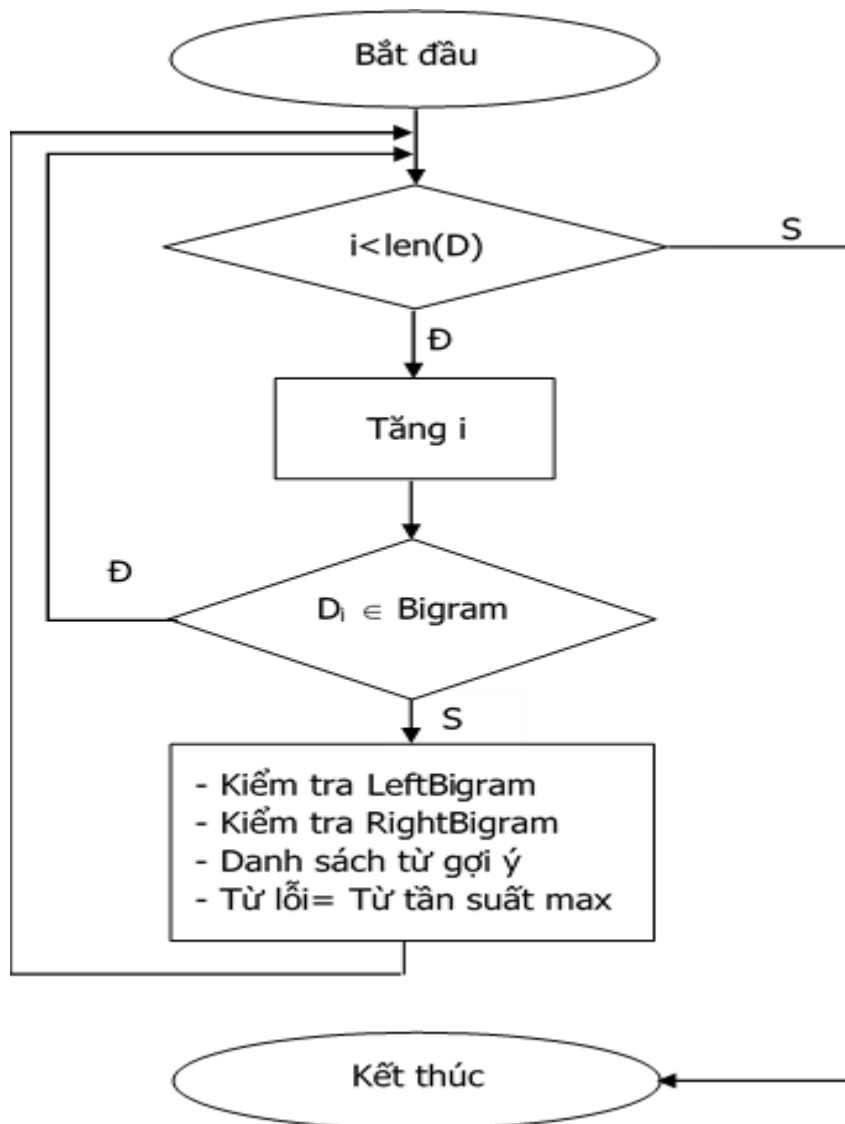
Bigram phải:  $W_j^iW^{i+1}$

Từ điển Bigram được tác giả xây dựng bằng cách thu thập dữ liệu từ nhiều nguồn trên mạng như vnexpress.net, dantri.com.vn, wikipedia.org. Dữ liệu bao gồm nhiều chủ đề như khoa học, xã hội, thể thao, giải trí... Kích thước của tập dữ liệu của tác giả khoảng 66 MB. Sau đó tác giả sẽ tính tần số của các bigram này. Kết quả được mô tả trong bảng 3.1.

**Bảng 3.1.** Kết quả Bigram tập dữ liệu

	<b>Kích thước tệp tin trước khi tách Bigram</b>	<b>Số Bigram tách được</b>	<b>Kích thước sau khi tách Bigram</b>
Bigram	66 MB	4.836.571	82 MB

Thuật toán phát hiện và sửa lỗi chính tả văn bản dựa vào kỹ thuật N-gram được tác giả cài đặt và mô tả như sau:



**Hình 3.14.** Sơ đồ khôi phục lỗi chính tả sử dụng Bigram

### 3.3. Bài toán đánh chỉ mục và tìm kiếm

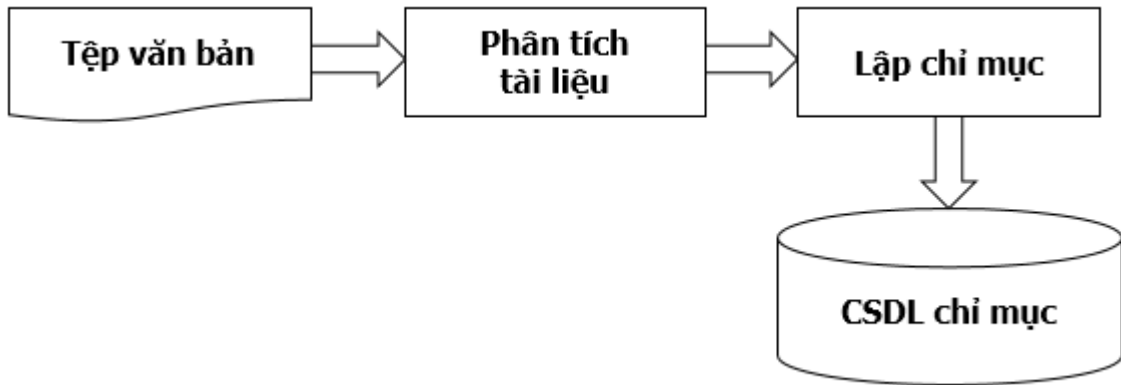
#### 3.3.1. Phát biểu bài toán

Bài toán lập chỉ mục cho tệp văn bản trải qua hai bước:

- Bước 1: Xác định các mục từ, khái niệm có khả năng đại diện cho văn bản sẽ được lưu trữ. Đây là quá trình phân tích tệp văn bản bao gồm các quá trình như tách từ, loại bỏ từ dừng...

- Bước 2: Xác định trọng số cho từng mục từ, trọng số này là giá trị phản ánh tầm quan trọng của mục từ đó trong văn bản.

Hình 3.15 mô tả các bước để lập chỉ mục tài liệu.

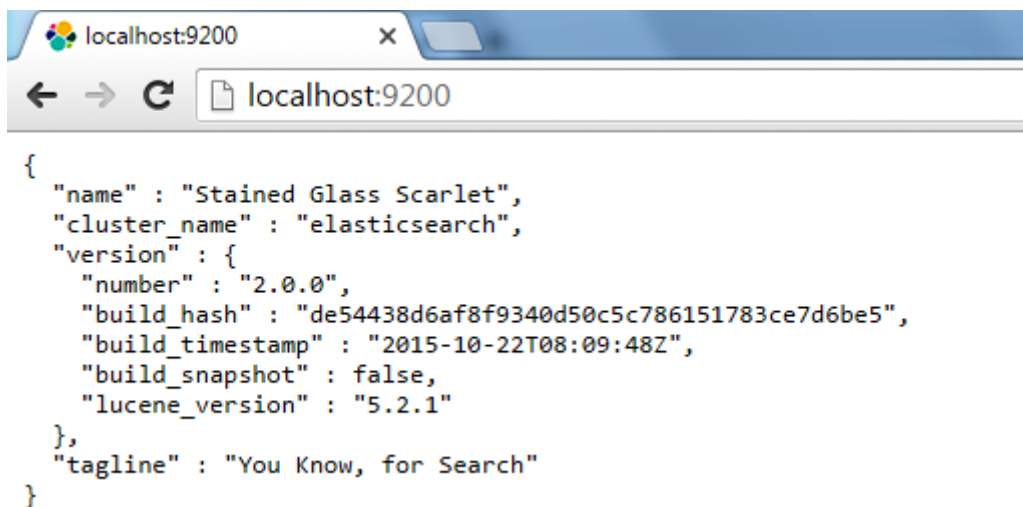


**Hình 3.15.** Mô tả quá trình lập chỉ mục tài liệu

### 3.3.2. Lập chỉ mục và tìm kiếm bằng Elasticsearch

Trước khi tiến hành lập chỉ mục bằng Elasticsearch, cần thực hiện khởi động Elasticsearch. Khởi động Elasticsearch bằng câu lệnh: “*sudo service elasticsearch start*”.

Để kiểm tra, trên thanh địa chỉ của trình duyệt web, truy cập vào địa chỉ <http://localhost:9200>. Nếu thành công thì kết quả sẽ có như mô tả của hình 3.16.



**Hình 3.16.** Kiểm tra khởi động Elasticsearch

Tạo index: Để tạo chỉ mục có tên là “lectures” thì sau khi khởi động elasticsearch. Sử dụng câu lệnh: *curl -XPUT 'localhost:9200/lectures'*.

Đưa ra danh sách tất cả các chỉ mục có trong Elasticsearch bằng câu lệnh: *curl 'localhost:9200/\_cat/indices?v'*. Kết quả được mô tả trong hình 3.17.



health	status	index	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	lectures	5	1	0	0	635b	635b

**Hình 3.17.** Danh sách các chỉ mục hiện có. Tên chỉ mục là lectures, số tài liệu docs.count hiện tại có giá trị bằng 0 (do chưa tạo tài liệu cho chỉ mục này).

Tạo type và document cho chỉ mục: Định dạng của một document sẽ có kiểu {"url": "đường dẫn đến tệp video bài giảng", "contents": "nội dung tệp tin văn bản nội dung đã được xử lý"}. Document ở đây thuộc type "external". Câu lệnh để tạo type và document như hình 3.18.

```

1 curl -XPUT 'localhost:9200/lectures/external/1?pretty' -d '
2 {
3   "url": "uploads/video001.mp4",
4   "content": "ngày nay khi công nghệ thông tin..."
5 }
```

**Hình 3.18.** Tạo type và document cho chỉ mục.

Chỉ mục được tạo có tên là lectures, type là external. Document có hai tham số là url và content. URL là đường dẫn đến tệp tin video, và content là nội dung của video bài giảng. Id của document ở đây được gán bằng 1.

Nếu thực hiện lệnh POST không gán id cho document thì Elasticsearch sẽ tạo một id tự động cho document.

```

1 curl -XPOST 'localhost:9200/lectures/external/?pretty' -d '
2 {
3   "url": "uploads/video001.mp4",
4   "content": "ngày nay khi công nghệ thông tin..."
5 }
```

**Hình 3.19.** Tạo type và document bằng lệnh POST. Id của document được Elasticsearch gán tự động.

Lấy document: Sử dụng câu lệnh GET để lấy ra document với id và chỉ mục tương ứng: `curl -XGET 'localhost:9200/lectures/external/1?pretty'`.

Cập nhật document: Thực hiện lệnh tạo document với id đã tồn tại thì thông tin của document cũng sẽ được cập nhật lại.

```

1 curl -XPUT 'localhost:9200/lectures/external/1?pretty' -d '
2 {
3   "url": "uploads/video001.mp4",
4   "content": "nội dung thay đổi..."
5 }'

```

**Hình 3.20.** Cập nhật lại document cho chỉ mục với id đã tồn tại.

Hoặc có thể sử dụng lệnh UPDATE trực tiếp được mô tả trong hình 3.21.

```

1 curl -XPOST 'localhost:9200/lectures/external/1/_update?pretty' -d '
2 {
3   "doc": {
4     "url": "uploads/video001.mp4",
5     "content": "nội dung thay đổi..."
6   }
7 }'

```

**Hình 3.21.** Thực hiện cập nhật lại document bằng câu lệnh UPDATE

Xóa chỉ mục: Để xóa chỉ mục đã tạo, sử dụng câu lệnh như sau:

*curl -XDELETE 'localhost:9200/lectures?pretty'.*

Xóa document: Câu lệnh để xóa một document đã tồn tại bằng cách:

*curl -XDELETE 'localhost:9200/lectures/external/1?pretty'.*

Tìm kiếm các document trên index:

The screenshot shows the Elasticsearch head interface. The search query is 'giáo án điện tử' and the results are displayed in a table. The table has columns for \_index, \_type, \_id, \_score, url, and content. There are 11 hits shown.

_index	_type	_id	_score	url	content
lectures	external	AVSi2CNEYDjXrjSOB0Y4	0.41683638	uploads/video05.mp4	phần lớn các giáo viên ng
lectures	external	AVSi2jxmYDjXrjSOB0Y7	0.23890013	uploads/video07.mp4	chính gì những khó khăn
lectures	external	AVSi2XNYDjXrjSOB0Y6	0.22431326	uploads/video06.mp4	khi sử dụng giáo án điện t
lectures	external	AVSi3gdLYDjXrjSOB0Y_	0.19815733	uploads/video10.mp4	Khi sử dụng giáo án điện t
lectures	external	AVSi3NxWYDjXrjSOB0Y-	0.12944615	uploads/video09.mp4	để tạo được một giáo án
lectures	external	AVSi1zHIYDjXrjSOB0Y3	0.014263429	uploads/video04.mp4	hiện nay bộ giáo dục và đ
lectures	external	AVSi3rrDYDjXrjSOB0ZB	0.0075643254	uploads/video11.mp4	Nếu chỉ trình bày suông,t
lectures	external	AVSi0TqPYDjXrjSOB0Y0	0.007408212	uploads/video02.mp4	công nghệ thông tin từ là
lectures	external	AVSi6-OLYDjXrjSOB0Yy	0.0060319444	uploads/video01.mp4	ngày nay khi công nghệ t
lectures	external	AVSi0vp5YDjXrjSOB0Y1	0.004688555	uploads/video03.mp4	cuốn sách cũng được hình

**Hình 3.22.** Tìm kiếm document trên chỉ mục

Thời gian tìm kiếm cho câu truy vấn “giáo án điện tử” là 0.030 giây. Hiện thị 10 kết quả đầu tiên có liên quan đến truy vấn. Kết quả được sắp xếp theo thứ tự giảm dần của score.

Kết thúc chương 3, tác giả đã trình bày chi tiết các giải pháp và các kỹ thuật cài đặt xây dựng hệ thống cho phép tìm kiếm các video bài giảng dựa vào chuỗi truy vấn nhập vào của người dùng. Chương tiếp theo, tác giả sẽ trình bày quá trình thực nghiệm và các đánh giá chương trình.

## CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM, ĐÁNH GIÁ VÀ KẾT LUẬN

### 4.1. Công cụ, môi trường thực nghiệm

Để phục vụ cho quá trình thực nghiệm, tác giả sử dụng cấu hình phần cứng và các công cụ phần mềm thể hiện trong hai bảng 4.1 và bảng 4.2 như sau:

**Bảng 4.1.** Thông số phần cứng

STT	Thành phần	Thông số kĩ thuật
1	CPU	Intel ® Pentium ® Dual core T3200 2.00GHz
2	RAM	DDR II - 3GB
3	Hệ điều hành	Ubuntu 14.04 LTS
4	Bộ nhớ ngoài	150 GB

**Bảng 4.2.** Danh sách công cụ phần mềm

STT	Tên công cụ	Chức năng	Nguồn tải
1	Sublime Text 3	Trình soạn thảo và bẫy lỗi chương trình.	<a href="https://www.sublimetext.com">https://www.sublimetext.com</a>
2	PHP 5.0	Ngôn ngữ lập trình dùng thực nghiệm.	<a href="http://php.net/downloads.php">http://php.net/downloads.php</a>
3	FFMpeg	Công cụ xử lý video.	<a href="https://ffmpeg.org/download.html">https://ffmpeg.org/download.html</a>
4	Imagemagick	Công cụ chuyển đổi ảnh màu thành ảnh đa cấp xám.	<a href="http://www.imagemagick.org/script/binary-releases.php">http://www.imagemagick.org/script/binary-releases.php</a>
5	Tesseract -OCR	Công cụ nhận dạng kí tự quang học.	<a href="https://github.com/tesseract-ocr">https://github.com/tesseract-ocr</a>
6	Aspell	Công cụ kiểm tra lỗi chính tả.	<a href="http://aspell.net/">http://aspell.net/</a>
7	Pspell	Thư viện lập trình sửa lỗi chính tả trên ngôn ngữ PHP.	<a href="http://php.net/manual/en/intro.pspell.php">http://php.net/manual/en/intro.pspell.php</a>
8	Vietnamese Dictionary	Từ điển từ vựng của Tiếng Việt.	<a href="https://github.com/lec5/hunspell-vi/tree/master/dictionaries">https://github.com/lec5/hunspell-vi/tree/master/dictionaries</a>
9	Teleport Pro	Công cụ hỗ trợ tải dữ liệu trên mạng.	<a href="http://www.tenmax.com/teleport/pro/download.htm">http://www.tenmax.com/teleport/pro/download.htm</a>
10	Elasticsearch	Công cụ hỗ trợ đánh chỉ mục và tìm kiếm tài liệu.	<a href="https://www.elastic.co/">https://www.elastic.co/</a>

## 4.2. Kết quả thực nghiệm, đánh giá

Trong phần thực nghiệm này, tác giả lấy ngẫu nhiên trên mạng năm video bài giảng. Tiến hành trích xuất các khung hình từ lần lượt cho các video này thu được bảng kết quả mô tả ở bảng 4.3.

**Bảng 4.3.** Kết quả thực hiện trích xuất khung hình từ video

STT	Định dạng	Kích thước (MB)	Thời gian (phút:giây)	Số khung hình thu được	Kích thước (MB)
1	mp4	23,8	6:22	382	404,6
2	mp4	48,1	6:38	398	450,7
3	mp4	32,1	3:07	187	174,8
4	mp4	137,6	28:27	1707	1740,8
5	mp4	19,6	2:35	155	139,4

Chúng ta có thể điều chỉnh tăng, giảm tần số FPS để nhằm thu được số lượng khung hình phù hợp. Qua quá trình thực nghiệm, để đảm bảo không bị thừa hoặc thiếu nội dung thì tần số FPS mà tác giả sử dụng trong luận văn này là 1 FPS.

Số lượng khung hình thu được của mỗi video tương ứng như trong bảng 4.3. Vì các khung hình hiện tại đang là ảnh màu, nhằm nâng cao chất lượng của quá trình OCR. Tác giả tiến hành chuyển đổi toàn bộ tập khung hình thu được thành ảnh đa cấp xám.

Bảng 4.4 mô tả kết quả nhận dạng kí tự quang học bằng công cụ Tesseract-OCR. Tập kết quả được lưu trữ với định dạng văn bản .txt.

Để đánh giá quá trình OCR bằng Tesseract-OCR, tác giả sử dụng độ chính xác - P, độ hồi tưởng - R, và độ đo F1.

Độ chính xác OCR của một video  $P = \frac{\sum_{i=1}^n P_i}{N}$ . Với N là tổng số tệp tin của video đó.

Độ chính xác  $P_i$  được tính theo công thức:

$$P_i = \frac{\sum \text{Từ nhận dạng được|đúng}}{\sum \text{Từ nhận dạng được}} * 100\%$$

Độ hồi tưởng OCR của một video  $R = \frac{\sum_{i=1}^n R_i}{N}$ . Với N là tổng số tệp tin của video đó.

Độ hồi tưởng  $R_i$  được tính theo công thức:

$$R_i = \frac{\sum \text{Từ nhận dạng được|đúng}}{\sum \text{Tổng số từ lỗi thực tế}} * 100\%$$

Độ đo F1 là sự kết hợp của hai độ đo chính xác và độ đo hồi tưởng. Độ đo F1 đối với một video được tính theo công thức.

$$F1 = 2 * \frac{\text{độ chính xác} * \text{độ hồi tưởng}}{\text{độ chính xác} + \text{độ hồi tưởng}}$$

**Bảng 4.4.** Kết quả thực hiện Tesseract-OCR đối với tập khung hình thu được

STT	Số lượng	Kích thước tập kết quả (KB)	Độ chính xác (%)	Độ hồi tưởng (%)	Độ F1 (%)
1	382	136,3	71,2	81,8	76,13
2	398	100,5	71,1	82,0	76,16
3	187	33,7	76,4	67,0	71,39
4	1707	529,1	66,4	76,2	70,96
5	155	45,0	77,5	66,3	71,46
<b>Trung bình</b>			<b>72,52</b>	<b>74,66</b>	<b>73,22</b>

Qua thực nghiệm tác giả nhận ra rằng, đối với các khung hình không bị ảnh hưởng bởi hiệu ứng trình chiếu thì kết quả nhận dạng bằng Tesseract-OCR cho kết quả với độ chính xác cao, xấp xỉ khoảng 96% đến 100%. Nhưng đối với các khung hình bị ảnh hưởng thì cho kết quả nhận dạng thấp, khoảng 56% - 64%. Vì vậy độ chính xác trung bình đối với một video bị giảm đáng kể, xấp xỉ 72,52%. Đây cũng là thách thức và hạn chế của tác giả trong luận văn này.

Tập kết quả sau quá trình OCR tiếp tục được xử lý trùng lặp bằng kỹ thuật Shingling. Kết quả thực hiện loại bỏ trùng lặp được mô tả trong hình 4.5.

**Bảng 4.5.** Kết quả thực hiện NDD với kỹ thuật Shingling

STT	Tập đầu vào	Số văn bản đại diện thu được	Số slide thực tế	Số văn bản đại diện đúng	Độ chính xác (%)	Độ hồi tưởng (%)	Độ F1 (%)
1	382	14	22	12	85,7	54,5	66,63
2	398	24	25	22	91,6	88,0	89,76
3	187	42	35	34	80,1	97,1	87,78
4	1707	14	18	13	92,8	72,2	81,21
5	155	21	24	18	85,7	75,0	79,99
<b>Trung bình</b>					<b>87,18</b>	<b>77,36</b>	<b>81,07</b>

Độ chính xác, độ hồi tưởng và độ đo F1 được dùng để đánh giá quá trình xử lý trùng lặp văn bản. Kết quả của quá trình này là tập văn bản đại diện cho video bài giảng đầu vào.

Độ chính xác P được tính bằng công thức:

$$P = \frac{\sum \text{Văn bản đại diện đúng}}{\sum \text{Văn bản đại diện thu được}} * 100\%$$

Độ hồi tưởng R được tính theo công thức:

$$R = \frac{\sum \text{Văn bản đại diện đúng}}{\sum \text{Văn bản đại diện thực tế}} * 100\%$$

$$\text{Độ đo F1 được tính là: } F1 = 2 * \frac{P * R}{P + R}$$

Sau khi xử lý trùng lặp văn bản, tập hợp các văn bản đại diện được gộp chung thành một văn bản duy nhất. Trước khi xử lý lỗi chính tả, tập văn bản cần được làm sạch như đã trình bày chi tiết trong mục 3.4.2.

Tập dữ liệu sau khi được làm sạch đều bao gồm cả hai loại lỗi non-word và real-word. Trong luận văn này, tác giả kết hợp cả thư viện Aspell để kiểm tra lỗi non-word và sử dụng Bi-gram để phát hiện lỗi real-word. Kết quả mô tả quá trình phát hiện lỗi chính tả được mô tả trong bảng 4.6.

Độ chính xác P được tính bằng công thức:

$$P = \frac{\sum \text{Số từ phát hiện được đúng}}{\sum \text{Số từ phát hiện được}} * 100\%$$

Độ hồi tưởng R được tính theo công thức:

$$R = \frac{\sum \text{Số từ phát hiện được đúng}}{\sum \text{Số từ lỗi thực tế}} * 100\%$$

$$\text{Độ đo F1 được tính là: } F1 = 2 * \frac{P * R}{P + R}$$

**Bảng 4.6.** Kết quả quá trình phát hiện lỗi chính tả dùng Aspell kết hợp Bi-gram

STT	Tập đầu vào (số từ)	Tổng số lỗi thực tế	Số lỗi phát hiện được	Số lỗi phát hiện đúng	Độ chính xác (%)	Độ hồi tưởng (%)	Độ F1 (%)
1	946	77	71	66	92,9	85,7	89,15
2	1365	121	112	96	85,7	79,3	82,38
3	2482	43	33	18	54,54	41,8	47,33
4	786	96	91	85	93,4	88,54	90,91
5	1520	31	26	22	84,6	70,9	77,15
<b>Trung bình</b>					<b>82,23</b>	<b>73,25</b>	<b>77,38</b>

Danh sách những từ gợi ý cho từ phát hiện lỗi, tác giả sử dụng từ điển kết hợp với khoảng cách chỉnh sửa nhỏ nhất và tần suất xuất hiện Bi-gram để lựa chọn từ thay thế phù hợp. Bảng kết quả sửa lỗi chính tả được mô tả bằng bảng 4.7.

Độ chính xác P được tính bằng công thức:

$$P = \frac{\sum \text{Số từ sửa được|đúng}}{\sum \text{Số từ sửa được}} * 100\%$$

Độ hồi tưởng R được tính theo công thức:

$$R = \frac{\sum \text{Số từ sửa được|đúng}}{\sum \text{Số từ lỗi thực tế}} * 100\%$$

Độ đo F1 được tính là:  $F1 = 2 * \frac{P * R}{P + R}$

**Bảng 4.7.** Kết quả quá trình sửa lỗi chính tả

STT	Số lỗi phát hiện	Số lỗi sửa	Số lỗi sửa đúng	Độ chính xác (%)	Độ hồi tưởng (%)	Độ F1 (%)
1	71	69	49	71,0	69,0	69,99
2	112	102	62	65,8	55,4	57,97
3	33	16	9	56,3	27,3	36,77
4	91	84	43	51,2	50,5	49,17
5	26	28	18	64,3	69,2	66,66
<b>Trung bình</b>				<b>60,72</b>	<b>53,64</b>	<b>56,11</b>

Như đã trình bày ở mục 3.4 về khó khăn khi sửa lỗi chính tả Tiếng Việt. Vì vậy trong luận văn này, tác giả đã cố gắng để nhằm cải thiện chất lượng của quá trình sửa lỗi. Độ chính xác trung bình xấp xỉ khoảng 60,72%.

### 4.3. Kết luận

#### 4.3.1. Kết quả đạt được

Trong luận văn này, tác giả hướng tới mục đích là tìm hiểu và nghiên cứu phương pháp để xây dựng một hệ thống tra cứu video dựa trên nội dung. Video tác giả quan tâm là các video bài giảng dạng silde. Nội dung của truy vấn sẽ là các từ hoặc các cụm từ có liên quan đến nội dung văn bản bên trong các video bài giảng.

Qua bốn chương, luận văn đã trình bày về các khái niệm liên quan đến công cụ tìm kiếm. Các phương pháp tiếp cận, kỹ thuật áp dụng để giải quyết các bài toán về xây dựng công cụ tìm kiếm video. Ứng dụng các phương pháp, kỹ thuật để thực nghiệm xây dựng một hệ thống tìm kiếm video bài giảng dựa trên nội dung.

Các đóng góp chính của luận văn:

- Hệ thống lại kiến thức, khái niệm liên quan và kiến trúc của công cụ tìm kiếm.



- Trình bày mô hình các bài toán cần xử lý trong quá trình xây dựng công cụ tìm kiếm video.
- Phân tích các phương pháp tiếp cận để giải quyết các bài toán và lựa chọn kỹ thuật để thực nghiệm.
- Xây dựng thử nghiệm ứng dụng tìm kiếm video bài giảng dạng slide dựa trên nội dung.

#### **4.3.2. Định hướng phát triển**

Với những kết quả đạt được trong luận văn này, tác giả hy vọng trong tương lai sẽ:

- Thử nghiệm với dữ liệu đa dạng hơn và lớn hơn. Thu thập và xử lý được với nhiều định dạng video.
- Nghiên cứu các phương pháp, kỹ thuật để nâng cao chất lượng chương trình sửa lỗi chính tả Tiếng Việt.
- Cải tiến và nghiên cứu để nâng cao chất lượng, giảm thời gian xử lý video đầu vào.

## TÀI LIỆU THAM KHẢO

1. Andrei Z. Broder. (2000), “Identifying and Filtering Near-Duplicate Documents”, *11<sup>th</sup> Annual Symposium on Combinatorial Pattern Matching*, Springer-Verlag London, pp.1-10.
2. Bassma S. Alsulami. (2012), “Near Duplicate Document Detection Survey”, *International Journal of Computer Science & Communication Networks*, pp. 147-151.
3. Chirag Patel, Atul Patel, Dharmendra Patel. (2012), “Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study”, *International Journal of Computer Applications*, Volume 55 –No.10, pp. 50-56.
4. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. (2009), *Introduction to Information Retrieval*, Cambridge University Press, Cambridge University.
5. David C. Gibbon. (2012), *Introduction to Video Search Engines*, Springer Verlag Berlin Heidelberg, Spinger.
6. Gurmeet Singh Manku, Arvind Jain, Anish Das Sarma. (2007), “Detecting Near Duplicates for Web Crawling”, *16<sup>th</sup> International Conference on World Wide Web*, pp. 141-150.
7. Haojin Yang, Maria Siebert, Patrick Lühne, Harald Sack, Christoph Meinel. (2011), “Automatic Lecture Video Indexing Using Video OCR Technology”, *2011 IEEE International Symposium on*, pp. 111 – 116.
8. Haojin Yang. (2011), “Lecture Video Indexing and Analysis Using Video OCR Technology”, *7<sup>th</sup> International Conference IEEE Dijon France*, pp. 54-61.
9. Hannaneh Hajishirzi, Wen-tau Yih, Aleksander Kolcz. (2010), “Adaptive Near-Duplicate Detection via Similarity Learning”, *ACM SIGIR conference on Research and development in information retrieval*, pp. 419-426.
10. Nguyen Thi Xuan Huong, Tran-Thai Dang, The-Tung Nguyen, Anh-Cuong Le. (2015), “Using Large N-gram for Vietnamese Spell Checking”, *Advances in Intelligent Systems and Computing*, pp. 617-627.
11. Kukich, Karen. (1992), “Techniques for Automatically Correcting Words in Text”, *24<sup>th</sup> ACM Computing Surveys*, pp. 377–439.
12. Kurt Hornik, Duncan Murdoch. (2011), “Watch Your Spelling”, *The R Journal Vol. 3*, pp. 22-28.

13. Kyle Williams, C. Lee Giles. (2013), "Near Duplicate Detection in an Academic Digital Library", *2013 ACM Symposium on Document Engineering*, pp. 91-94.
14. Martin Røst Halvorsen. (2007), *Content-based lecture video indexing*, Master's Thesis, Department of Computer Science and Media Technology Gjøvik University College.
15. Martin Potthast, Benno Stein. (2008), "New Issues in Near-duplicate Detection", *31<sup>th</sup> Conf. of the German Classification Society*, pp. 601-609.
16. Pratip Samanta, Bidyut B. Chaudhuri. (2013), "A simple real-word error detection and correction using local word bigram and trigram", *Association for Computational Linguistics and Chinese Language Processing*, pp. 211-220.
17. Ritika Mishra, Navjot Kaur. (2013), "A Survey of Spelling Error Detection and Correction Techniques", *International Journal of Computer Trends and Technology*, pp. 372-374.
18. Radu Gheorghe, Matthew Lee Hinman, Roy Russo. (2016), *Elasticsearch in Action*, Manning Publications Co, Shelter Island.
19. Smith, R. (2007), *An Overview of the Tesseract OCR Engine*, In proceedings of Document analysis and Recognition. IEEE Ninth International Conference.
20. Suzan Verberne. (2002), *Context-sensitive spellchecking based on word trigram probabilities*, Master thesis Taal, Spraak & Informatica University of Nijmegen.
21. Youssef Bassil, Mohammad Alwani. (2012), "Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information", *Computer and Information Science*, Vol. 5, No. 3, May 2012, pp. 37-48.