**A close-up of a sign

AI-generated content may be incorrect.**

**Báo Cáo Cuối Kỳ Môn Truy Xuất Thông Tin CS419.P21**

**Giảng viên hướng dẫn: Nguyễn Trọng Chỉnh**

**Sinh viên: Phạm Duy Long - 20521573**

**CHƯƠNG 1: GIỚI THIỆU**

**1.1 Tổng quan về Truy xuất Thông tin**

Truy xuất Thông tin (Information Retrieval – IR) là một lĩnh vực nghiên cứu và ứng dụng quan trọng trong khoa học máy tính, tập trung vào việc tìm kiếm các tài liệu liên quan từ một bộ sưu tập tài liệu lớn dựa trên nhu cầu thông tin của người dùng. Trong bối cảnh sự bùng nổ của Internet và khối lượng dữ liệu văn bản khổng lồ, khả năng truy xuất thông tin hiệu quả trở nên vô cùng thiết yếu. Mục tiêu chính của một hệ thống IR là cung cấp cho người dùng những tài liệu chính xác và phù hợp nhất với câu truy vấn của họ.

Trong lịch sử phát triển của IR, ba mô hình khung kinh điển đã được sử dụng rộng rãi để biểu diễn và so khớp tài liệu với truy vấn: mô hình Boolean, mô hình Vector Space (VSM) và mô hình Xác suất (Probabilistic Model). Mỗi mô hình có những đặc điểm và nguyên tắc hoạt động riêng, phù hợp với các loại nhu cầu tìm kiếm khác nhau.

Mô hình Boolean, với bản chất đơn giản, thực hiện so khớp truy vấn dựa trên các phép toán Boolean (AND, OR, NOT). Mô hình này trả về các tài liệu khớp chính xác với ngữ nghĩa của truy vấn. Tuy nhiên, nhược điểm lớn của nó là thiếu khả năng khớp một phần, dẫn đến việc có thể trả về quá ít hoặc quá nhiều tài liệu, làm giảm hiệu quả truy xuất trong nhiều trường hợp thực tế. Sự cứng nhắc này khiến mô hình Boolean ít phù hợp với các truy vấn ngôn ngữ tự nhiên phức tạp hoặc nhu cầu tìm kiếm linh hoạt.

Ngược lại, mô hình Vector Space được giới thiệu bởi G. Salton vào cuối những năm 1960, đã khắc phục hạn chế của mô hình Boolean bằng cách cho phép khớp một phần. Mô hình này biểu diễn cả tài liệu và truy vấn dưới dạng các vector trong một không gian đa chiều, sử dụng các trọng số phi nhị phân cho các thuật ngữ chỉ mục. Khả năng tính toán mức độ tương đồng giữa vector truy vấn và vector tài liệu đã giúp VSM đạt được độ liên quan cao hơn trong việc truy xuất tài liệu. Sự ra đời của VSM đánh dấu một bước tiến quan trọng, chuyển dịch trọng tâm từ "tìm kiếm chính xác" sang "tìm kiếm liên quan", phản ánh nhu cầu thực tế của người dùng khi đối mặt với lượng thông tin ngày càng tăng và không có cấu trúc rõ ràng. Sự dịch chuyển này cho thấy các hệ thống IR cần phải linh hoạt hơn, có khả năng xử lý sự mơ hồ và đa dạng của nhu cầu thông tin trong thế giới thực.

**1.2 Mục tiêu và Phạm vi của Đồ án**

Đồ án này tập trung vào việc nghiên cứu, cài đặt và đánh giá sâu sắc hai trong số các mô hình truy xuất thông tin quan trọng nhất: Mô hình Vector Space (VSM) và Mô hình Xác suất, cụ thể là thuật toán BM25. Mục tiêu chính của đồ án là cung cấp một cái nhìn toàn diện từ lý thuyết đến thực hành về cách các hệ thống IR hoạt động, cách chúng xử lý và tổ chức dữ liệu, cũng như cách chúng đánh giá và xếp hạng độ liên quan của tài liệu.

Phạm vi của đồ án bao gồm việc tìm hiểu chi tiết các vấn đề lý thuyết nền tảng của từng mô hình, từ cách biểu diễn tài liệu và truy vấn đến nguyên tắc tính toán độ liên quan và xếp hạng. Sau đó, đồ án sẽ đi sâu vào việc triển khai thử nghiệm các mô hình này trên bộ ngữ liệu Cranfield, một bộ sưu tập tài liệu kinh điển trong lĩnh vực IR, để so sánh hiệu suất của chúng. Việc đánh giá hiệu suất sẽ được thực hiện bằng các độ đo chuẩn như Precision, Recall và Mean Average Precision (MAP), cùng với phân tích Precision nội suy 11 điểm theo tiêu chuẩn TREC. Cuối cùng, đồ án sẽ so sánh hiệu suất của các mô hình tự cài đặt với một thư viện IR phổ biến như Whoosh để có cái nhìn khách quan về hiệu quả triển khai.

Tuy nhiên, cần lưu ý rằng bộ ngữ liệu Cranfield, mặc dù được sử dụng rộng rãi trong các thí nghiệm IR ban đầu và được coi là một tập dữ liệu chuẩn để so sánh các thuật toán , Hiemstra (2000) đã mô tả nó là "tương đối lỗi thời" so với các bộ sưu tập TREC hiện đại hơn. Điều này hàm ý rằng các kết quả thực nghiệm thu được trên Cranfield, dù có giá trị trong việc kiểm chứng lý thuyết và so sánh cơ bản, có thể không hoàn toàn phản ánh hiệu suất hoặc khả năng mở rộng của các mô hình này khi áp dụng trên các tập dữ liệu lớn hơn, đa dạng hơn và phức tạp hơn trong môi trường thực tế hiện nay. Do đó, các kết luận về hiệu suất cần được xem xét trong bối cảnh đặc thù của bộ ngữ liệu này.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ TRUY XUẤT THÔNG TIN**

**2.1 Mô hình so khớp tài liệu và truy vấn**

Trong truy xuất thông tin, mô hình so khớp tài liệu và truy vấn đóng vai trò cốt lõi, quyết định cách hệ thống hiểu và đánh giá mối quan hệ giữa nhu cầu thông tin của người dùng (truy vấn) và các tài liệu trong bộ sưu tập. Dưới đây là phân tích chi tiết về hai mô hình đã chọn: Vector Space Model và Mô hình Xác suất (BM25).

**2.1.1 Mô hình Vector Space Model (VSM)**

Mô hình Vector Space (VSM), được đề xuất bởi G. Salton, là một trong những mô hình phổ biến và có ảnh hưởng nhất trong truy xuất thông tin. Ưu điểm nổi bật của VSM là khả năng xử lý các truy vấn ngôn ngữ tự nhiên và xếp hạng tài liệu theo mức độ liên quan, thay vì chỉ trả về kết quả khớp chính xác như mô hình Boolean.

**Biểu diễn tài liệu và truy vấn:** Trong VSM, cả tài liệu và truy vấn đều được biểu diễn dưới dạng các vector trong một không gian đa chiều. Mỗi chiều của không gian tương ứng với một "term" (thuật ngữ chỉ mục) duy nhất trong toàn bộ bộ sưu tập tài liệu.

* Đối với tài liệu dj​, vector tài liệu được định nghĩa là dj​=(w1,j​,w2,j​,...,wt,j​), trong đó wi,j​ là trọng số của term ki​ trong tài liệu dj​. Trọng số này là một giá trị dương và phi nhị phân, phản ánh mức độ quan trọng của term đó đối với tài liệu.
* Tương tự, truy vấn q cũng được biểu diễn dưới dạng một vector truy vấn q​=(w1,q​,w2,q​,...,wt,q​), với wi,q​ là trọng số của term ki​ trong truy vấn q. Các trọng số này cũng không âm (wi,q​≥0). Số lượng chiều t trong không gian vector bằng tổng số các term chỉ mục duy nhất trong toàn bộ hệ thống.

**Nguyên tắc tính toán độ liên quan để xếp hạng:** Độ liên quan giữa một tài liệu dj​ và một truy vấn q trong VSM được đánh giá bằng mức độ "tương đồng" giữa vector tài liệu dj​ và vector truy vấn q​. Mức độ tương đồng này thường được đo bằng **cosine của góc giữa hai vector** (Cosine Similarity). Giá trị cosine càng gần 1, hai vector càng "hướng" cùng chiều, tức là tài liệu càng liên quan đến truy vấn. Công thức tính độ tương đồng cosine giữa tài liệu dj​ và truy vấn q là:



Trong đó:

* dj​​⋅q​ là tích vô hướng (dot product) của vector tài liệu và vector truy vấn.
* ∣dj​​∣ và ∣q​∣ là độ lớn (norm) hay độ dài Euclidean của vector tài liệu và vector truy vấn tương ứng. Cần lưu ý rằng yếu tố ∣q​∣ (độ dài vector truy vấn) không ảnh hưởng đến thứ tự xếp hạng của các tài liệu, vì nó là một hằng số cho mọi tài liệu khi xử lý cùng một truy vấn. Yếu tố ∣dj​​∣ (độ dài vector tài liệu) cung cấp sự chuẩn hóa trong không gian tài liệu, giúp tránh việc các tài liệu dài hơn tự động có điểm số cao hơn chỉ vì chúng chứa nhiều từ hơn. Vì tất cả các trọng số wi,j​ và wi,q​ đều không âm, giá trị sim(q,dj​) sẽ nằm trong khoảng từ 0 đến +1. Hệ thống VSM sẽ xếp hạng các tài liệu theo thứ tự giảm dần của độ tương đồng này, cho phép các tài liệu khớp một phần vẫn được truy xuất và xếp hạng.

**2.1.2 Mô hình Xác suất (Probabilistic Model - BM25)**

Mô hình Xác suất tiếp cận bài toán truy xuất thông tin từ góc độ thống kê, ước lượng xác suất một tài liệu là liên quan đến một truy vấn. Thuật toán Okapi BM25 là một trong những thuật toán xếp hạng dựa trên xác suất phổ biến và hiệu quả nhất, thường được sử dụng như một chuẩn mực trong các hệ thống IR hiện đại.

**Biểu diễn tài liệu và truy vấn:** Trong mô hình xác suất, tài liệu và truy vấn được xem xét như các tập hợp (hoặc chuỗi) các thuật ngữ. Hiemstra (2000) đề xuất một mô hình xác suất mới, nơi tài liệu và truy vấn được định nghĩa bởi một chuỗi có thứ tự các thuật ngữ đơn lẻ, một giả định phổ biến trong xử lý ngôn ngữ tự nhiên thống kê. Tuy nhiên, các mô hình thực tế hơn thường giả định độc lập giữa các thuật ngữ khi điều kiện hóa trên một tài liệu.

**Nguyên tắc tính toán độ liên quan để xếp hạng:** Nguyên tắc cơ bản của mô hình xác suất là ước tính xác suất một tài liệu D là liên quan đến một truy vấn Q, ký hiệu là P(D∣Q). Sử dụng định lý Bayes, xác suất này có thể được viết lại thành: A close up of a letter

AI-generated content may be incorrect.

Trong đó:

* P(D) là xác suất tiên nghiệm của tài liệu D (ví dụ: xác suất một tài liệu bất kỳ được chọn).
* P(Ti​∣D) là xác suất thuật ngữ Ti​ xuất hiện trong tài liệu D. Hiemstra (2000) đã cung cấp một lý giải xác suất cho việc sử dụng trọng số tf-idf, vốn trước đây bị chỉ trích vì thiếu cơ sở lý thuyết vững chắc. Lý giải này dựa trên ước lượng nội suy tuyến tính (linear interpolation estimator), kết hợp xác suất dựa trên tần suất tài liệu (toàn cục) và tần suất thuật ngữ (cục bộ) để xử lý vấn đề dữ liệu thưa thớt.

**Công thức tính trọng số BM25:** Thuật toán BM25 (Best Match 25) không phải là một mô hình xác suất thuần túy theo nghĩa chặt chẽ nhất, mà là một công thức xếp hạng heuristic được phát triển từ mô hình xác suất nhị phân độc lập. Nó tính toán điểm số tương đồng giữa truy vấn và tài liệu dựa trên tần suất thuật ngữ (TF), tần suất tài liệu ngược (IDF) và độ dài tài liệu. Công thức BM25 được sử dụng trong bài báo của Hiemstra (2000) và là phiên bản Cornell của thuật toán BM25 :



Trong đó:

* wqk​=tf(tk​,q) là tần suất của thuật ngữ tk​ trong truy vấn q.
* wdk​=k1​⋅((1−b)+b⋅avgDocLengthdocLength​)+tf(tk​,d)tf(tk​,d)⋅log(df(tk​)+0.5N−df(tk​)+0.5​)​

Các thành phần của công thức wdk​:

* tf(tk​,d): Tần suất của thuật ngữ tk​ trong tài liệu d.
* df(tk​): Tần suất tài liệu của thuật ngữ tk​ (số tài liệu chứa tk​).
* N: Tổng số tài liệu trong bộ sưu tập.
* docLength: Độ dài của tài liệu d.
* avgDocLength: Độ dài trung bình của tài liệu trong bộ sưu tập.
* k1​: Tham số điều chỉnh độ bão hòa của tần suất thuật ngữ (thường trong khoảng 1.2 - 2.0).
* b: Tham số điều chỉnh ảnh hưởng của độ dài tài liệu (thường trong khoảng 0.75). Các tham số k1​ và b cho phép BM25 linh hoạt điều chỉnh cách trọng số của thuật ngữ bị ảnh hưởng bởi tần suất của nó trong tài liệu và độ dài của tài liệu, giúp nó vượt trội hơn các công thức tf-idf đơn giản trong nhiều trường hợp.

**2.2 Phương pháp phân tích tài liệu**

Để một hệ thống truy xuất thông tin có thể hoạt động hiệu quả, các tài liệu thô và truy vấn của người dùng cần được chuyển đổi thành một định dạng có cấu trúc, dễ dàng cho việc xử lý và so khớp. Quá trình này được gọi là phân tích tài liệu, trong đó các văn bản được phân tách thành các "term" (thuật ngữ chỉ mục).

**2.2.1 Định nghĩa "Term"**

Trong bối cảnh truy xuất thông tin, một "term" là đơn vị cơ bản hoặc khái niệm duy nhất được sử dụng để lập chỉ mục và tìm kiếm tài liệu. Nó có thể là một từ đơn lẻ, một cụm từ, hoặc một khái niệm đã được chuẩn hóa. Các term này là "khối xây dựng" cơ bản của chỉ mục ngược và được sử dụng để tra cứu thông tin. Trong nhiều trường hợp, term có thể tương đương với các từ trong văn bản gốc hoặc các token đã được chuẩn hóa sau quá trình tiền xử lý.

**2.2.2 Các bước phân tích tài liệu thành các term**

Quá trình phân tích tài liệu thành các term bao gồm một chuỗi các bước tiền xử lý ngôn ngữ, mỗi bước có một mục đích cụ thể để tối ưu hóa hiệu suất truy xuất.

1. **Tokenization (Tách từ):**
   * **Mô tả:** Đây là bước đầu tiên, trong đó văn bản thô được chia nhỏ thành các đơn vị riêng lẻ, gọi là "token" hoặc "từ". Quá trình này thường bao gồm việc loại bỏ dấu câu, chuyển đổi tất cả các ký tự về chữ thường, và tách các từ dựa trên khoảng trắng hoặc các ký tự phân cách khác.
   * **Ví dụ minh họa:**
     + Văn bản gốc: "Truy xuất thông tin là một lĩnh vực hấp dẫn."
     + Sau tokenization:
     + *Chỗ cần chèn code từ file ipynb: Đoạn mã thực hiện tokenization.*
   * **Lý do:** Mục đích chính của tokenization là xác định các đơn vị cơ bản mà hệ thống sẽ sử dụng để lập chỉ mục và tìm kiếm. Nó biến một chuỗi ký tự liên tục thành một danh sách các từ có thể quản lý được, là nền tảng cho các bước xử lý tiếp theo.
2. **Stopword Removal (Loại bỏ từ dừng):**
   * **Mô tả:** Các "stopword" là những từ xuất hiện rất thường xuyên trong ngôn ngữ (ví dụ: "là", "một", "và", "của", "trong" trong tiếng Việt; "a", "the", "is" trong tiếng Anh) nhưng lại mang rất ít hoặc không mang ý nghĩa phân biệt nội dung tài liệu. Bước này liên quan đến việc loại bỏ các từ này khỏi danh sách token. Các hệ thống thường sử dụng một danh sách stopword được định nghĩa trước.
   * **Ví dụ minh họa:**
     + Token đã tách:
     + Danh sách stopword (ví dụ): ["là", "một"]
     + Sau stopword removal:
     + *Chỗ cần chèn code từ file ipynb: Đoạn mã thực hiện stopword removal.*
   * **Lý do:** Loại bỏ stopword giúp giảm đáng kể kích thước của chỉ mục, tiết kiệm không gian lưu trữ và tăng tốc độ xử lý truy vấn. Quan trọng hơn, nó cải thiện hiệu quả truy xuất bằng cách tập trung vào các thuật ngữ thực sự có khả năng phân biệt tài liệu, loại bỏ nhiễu từ các từ quá phổ biến.
3. **Stemming/Lemmatization (Đưa về dạng gốc/chuẩn):**
   * **Mô tả:** Stemming là quá trình loại bỏ các tiền tố và hậu tố từ các từ để đưa chúng về dạng gốc (stem) của chúng (ví dụ: "walk", "walked", "walking" đều trở thành "walk"). Lemmatization là một quá trình phức tạp hơn, sử dụng từ điển và phân tích hình thái để đưa từ về dạng nguyên thể (lemma) có ý nghĩa ngữ pháp chính xác (ví dụ: "am", "is", "are" đều trở thành "be").
   * **Ví dụ minh họa:**
     + Từ: "investigation", "investigate", "investigating"
     + Sau stemming: "investig" (dạng gốc không nhất thiết là từ có nghĩa)
     + Sau lemmatization: "investigate" (dạng nguyên thể có nghĩa)
   * **Lý do:** Mục đích của stemming/lemmatization là nhóm các từ có cùng ý nghĩa khái niệm lại với nhau, bất kể hình thái của chúng. Điều này giúp tăng "recall" (độ phủ) của hệ thống, vì một truy vấn sử dụng một biến thể của từ vẫn có thể tìm thấy các tài liệu chứa các biến thể khác. Ví dụ, nếu người dùng tìm "walking", hệ thống vẫn có thể tìm thấy tài liệu chứa "walked".
   * **Lưu ý đặc biệt:** Trong nghiên cứu của Manwar et al. (2012) trên bộ ngữ liệu Cranfield, stemming đã không được áp dụng. Các tác giả đưa ra lý do là để tránh "làm mất ngữ cảnh tìm kiếm", "có thể làm giảm độ chính xác (precision)", và "không thể áp dụng cho danh từ riêng". Điều này cho thấy có sự đánh đổi giữa recall (tăng khi stemming được áp dụng) và precision (có thể giảm nếu stemming quá mạnh hoặc không phù hợp), cũng như các cân nhắc về ngữ cảnh và loại dữ liệu.

**2.2.3 Vì sao phải thực hiện mỗi bước như vậy?**

Mỗi bước trong quá trình phân tích tài liệu đều có vai trò quan trọng và lý do riêng để tối ưu hóa hệ thống truy xuất thông tin:

* **Tokenization** là bước khởi tạo, biến dòng văn bản liên tục thành các đơn vị rời rạc, có thể xử lý được. Nếu không có tokenization, hệ thống sẽ không thể nhận diện các từ riêng lẻ để lập chỉ mục.
* **Stopword Removal** là bước tối ưu hóa. Nó giúp giảm kích thước chỉ mục và tăng tốc độ xử lý bằng cách loại bỏ các từ "nhiễu" không mang thông tin phân biệt. Điều này cho phép hệ thống tập trung tài nguyên vào các từ khóa thực sự quan trọng, từ đó cải thiện hiệu suất truy xuất và giảm chi phí tính toán.
* **Stemming/Lemmatization** là bước chuẩn hóa, nhằm mục đích tăng độ phủ (recall) của hệ thống. Bằng cách ánh xạ các dạng khác nhau của một từ về cùng một dạng gốc, hệ thống có thể tìm thấy nhiều tài liệu liên quan hơn ngay cả khi truy vấn sử dụng một biến thể từ khác. Điều này đặc biệt hữu ích khi người dùng không biết chính xác dạng từ được sử dụng trong tài liệu hoặc khi có nhiều biến thể hình thái của cùng một khái niệm.

Việc lựa chọn và thực hiện các bước tiền xử lý này có ảnh hưởng trực tiếp đến chất lượng của chỉ mục và hiệu suất truy xuất cuối cùng. Một quyết định tiền xử lý không phù hợp có thể dẫn đến việc bỏ lỡ các tài liệu liên quan (recall thấp) hoặc trả về các tài liệu không liên quan (precision thấp).

**2.3 Lập chỉ mục**

Lập chỉ mục là quá trình tổ chức và lưu trữ dữ liệu từ bộ sưu tập tài liệu theo một cấu trúc cho phép tìm kiếm nhanh chóng và hiệu quả. Trong truy xuất thông tin, chỉ mục ngược (inverted index) là cấu trúc dữ liệu được sử dụng rộng rãi nhất.

**2.3.1 Tổ chức lưu trữ: Chỉ mục ngược (Inverted Index)**

**Định nghĩa và cấu trúc:** Chỉ mục ngược là một cấu trúc dữ liệu "hướng term" (term-centric), trái ngược với các cấu trúc truyền thống "hướng tài liệu" (document-centric). Thay vì lưu trữ nội dung của từng tài liệu, chỉ mục ngược lưu trữ thông tin về các term hoặc token được tìm thấy trong bộ sưu tập tài liệu. Nó hiệu quả và gần như không có đối thủ về hiệu suất cho việc tìm kiếm văn bản ad-hoc.

Các thành phần chính của một chỉ mục ngược bao gồm:

* **Terms (Tokens):** Đây là các thuật ngữ chỉ mục duy nhất được trích xuất từ các tài liệu sau quá trình tiền xử lý (tokenization, stopword removal, stemming/lemmatization).
* **Document IDs (DocIDs):** Mỗi tài liệu trong bộ sưu tập được gán một định danh duy nhất, thường là một số nguyên, để tham chiếu hiệu quả.
* **Posting Lists:** Đối với mỗi term trong từ điển, có một danh sách các DocID của tất cả các tài liệu mà term đó xuất hiện. Danh sách này được gọi là "posting list". Thông tin bổ sung như tần suất của term trong tài liệu (term frequency) hoặc vị trí của term trong tài liệu cũng có thể được lưu trữ trong posting list để hỗ trợ các mô hình xếp hạng phức tạp hơn.

**Ví dụ minh họa cấu trúc chỉ mục ngược:** Giả sử chúng ta có các tài liệu sau:

* D1: "báo cáo truy xuất thông tin"
* D2: "truy xuất thông tin là gì"
* D3: "báo cáo khoa học"

Sau tiền xử lý (giả sử loại bỏ stopword "là", "gì"):

* D1: {báo cáo, truy xuất, thông tin}
* D2: {truy xuất, thông tin}
* D3: {báo cáo, khoa học}

Chỉ mục ngược sẽ có cấu trúc như sau:

* **báo cáo:**
* **truy xuất:**
* **thông tin:**
* **khoa học:**

Chỗ cần chèn ảnh minh họa cấu trúc chỉ mục ngược (ví dụ: Figure 1.3 hoặc tương tự từ tài liệu tham khảo ).

**Ưu điểm:** Chỉ mục ngược mang lại nhiều lợi ích quan trọng:

* **Hiệu quả tìm kiếm:** Cho phép tìm kiếm nhanh chóng các tài liệu chứa một hoặc nhiều term cụ thể bằng cách truy vấn trực tiếp posting list của các term đó.
* **Giảm yêu cầu lưu trữ:** Bằng cách nhóm các lần xuất hiện của cùng một term lại với nhau, chỉ mục ngược giúp giảm đáng kể yêu cầu lưu trữ so với việc lưu trữ toàn bộ văn bản hoặc các ma trận tần suất lớn.
* **Hỗ trợ xếp hạng:** Thông tin về tần suất term trong tài liệu (TF) và số lượng tài liệu chứa term (DF) có thể được lưu trữ và sử dụng trực tiếp để tính toán trọng số TF-IDF hoặc BM25, hỗ trợ xếp hạng tài liệu theo độ liên quan.

**2.3.2 Các bước xử lý để tạo chỉ mục**

Quá trình xây dựng một chỉ mục ngược bao gồm nhiều bước xử lý tuần tự, chuyển đổi dữ liệu văn bản thô thành cấu trúc chỉ mục hiệu quả:

1. **Thu thập tài liệu:** Bước đầu tiên là thu thập tất cả các tài liệu sẽ được lập chỉ mục. Trong đồ án này, bộ ngữ liệu Cranfield gồm 1398 tóm tắt liên quan đến lĩnh vực khí động học đã được sử dụng.
2. **Tiền xử lý dữ liệu:** Đây là giai đoạn quan trọng nhất, nơi văn bản thô được làm sạch và chuẩn hóa để trích xuất các term có ý nghĩa. Các bước đã được mô tả chi tiết trong Mục 2.2, bao gồm:
   * **Tokenization:** Chia văn bản thành các từ hoặc token riêng lẻ.
   * **Stopword Removal:** Loại bỏ các từ phổ biến không mang thông tin phân biệt.
   * **Stemming/Lemmatization (tùy chọn):** Đưa các từ về dạng gốc của chúng. Trong nghiên cứu được tham khảo, stemming không được áp dụng để tránh mất ngữ cảnh. Quá trình này tạo ra một danh sách các term đã chuẩn hóa cho mỗi tài liệu.
3. **Tạo danh sách cặp (term, docID):** Sau khi tiền xử lý, mỗi lần xuất hiện của một term trong một tài liệu sẽ tạo thành một cặp (term, docID). Ví dụ, nếu từ "thông tin" xuất hiện trong tài liệu D1 và D2, chúng ta sẽ có các cặp ("thông tin", D1) và ("thông tin", D2).
4. **Sắp xếp và gộp:** Danh sách các cặp (term, docID) này sau đó được sắp xếp theo thứ tự bảng chữ cái của term, và thứ cấp là theo docID. Các lần xuất hiện lặp lại của cùng một term trong cùng một tài liệu sẽ được gộp lại, và thông tin về tần suất (term frequency) có thể được ghi lại.
5. **Phân tách thành từ điển và posting lists:** Cuối cùng, danh sách đã sắp xếp được chia thành hai phần chính:
   * **Từ điển (Dictionary):** Chứa tất cả các term duy nhất trong bộ sưu tập.
   * **Posting Lists:** Đối với mỗi term trong từ điển, một danh sách các DocID (và các thông tin liên quan như TF) được tạo ra. Quá trình này biến văn bản thô thành một định dạng có cấu trúc, có thể tìm kiếm được, là yếu tố then chốt cho việc truy xuất hiệu quả. Việc lựa chọn các bước tiền xử lý và cách chúng được thực hiện sẽ ảnh hưởng trực tiếp đến chất lượng của chỉ mục và hiệu suất tìm kiếm cuối cùng.

**2.4 Truy xuất chỉ mục**

Sau khi chỉ mục đã được xây dựng, hệ thống có thể sử dụng nó để xử lý các truy vấn của người dùng và trả về các tài liệu liên quan. Quá trình này bao gồm các bước phân tích truy vấn, xử lý chỉ mục để tìm tài liệu, tính toán độ liên quan và xếp hạng kết quả.

**2.4.1 Các bước phân tích truy vấn**

Tương tự như tài liệu, truy vấn của người dùng cũng cần được tiền xử lý để chuyển đổi thành các term có thể so khớp với chỉ mục. Các bước này thường bao gồm:

1. **Tokenization:** Chia câu truy vấn thành các từ riêng lẻ.
2. **Stopword Removal:** Loại bỏ các từ dừng khỏi truy vấn. Việc này đảm bảo rằng các từ không mang ý nghĩa phân biệt sẽ không ảnh hưởng đến kết quả tìm kiếm.
3. **Stemming/Lemmatization (nếu áp dụng):** Nếu stemming hoặc lemmatization đã được áp dụng trong quá trình lập chỉ mục tài liệu, thì chúng cũng phải được áp dụng cho các term trong truy vấn để đảm bảo tính nhất quán và khả năng so khớp chính xác.

Ngoài ra, các hệ thống tiên tiến có thể thực hiện **mở rộng truy vấn (query expansion)**, thêm các thuật ngữ liên quan (ví dụ: từ đồng nghĩa) vào truy vấn ban đầu để cải thiện độ phủ (recall).

**2.4.2 Các bước xử lý để truy xuất chỉ mục**

Khi truy vấn đã được phân tích thành các term chuẩn hóa, hệ thống sẽ sử dụng chỉ mục ngược để tìm kiếm tài liệu:

1. **Tra cứu term:** Đối với mỗi term trong truy vấn, hệ thống sẽ tra cứu trong từ điển của chỉ mục ngược để tìm posting list tương ứng.
2. **Kết hợp posting lists:** Tùy thuộc vào loại truy vấn (Boolean hay xếp hạng), các posting list của các term trong truy vấn sẽ được kết hợp.
   * **Đối với truy vấn Boolean:** Các phép toán tập hợp (giao, hợp, hiệu) được thực hiện trên các tập DocID trong posting list để tìm ra các tài liệu khớp chính xác với điều kiện Boolean.
   * **Đối với truy vấn xếp hạng:** Các posting list được sử dụng để truy xuất các tài liệu tiềm năng và các thông tin cần thiết (như tần suất term) để tính toán điểm số liên quan.
3. **Tính toán điểm số:** Với mỗi tài liệu được truy xuất, hệ thống sẽ tính toán độ liên quan của nó với truy vấn dựa trên mô hình so khớp đã chọn.

**2.4.3 Tính độ liên quan giữa các tài liệu với kết quả phân tích câu truy vấn và xếp hạng**

Sau khi các tài liệu tiềm năng được xác định, bước cuối cùng là tính toán độ liên quan của chúng với truy vấn và xếp hạng chúng theo thứ tự giảm dần của độ liên quan.

* **Tính độ liên quan (VSM):** Đối với mô hình Vector Space, độ liên quan được tính bằng **độ tương đồng cosine** giữa vector truy vấn và vector tài liệu. Các trọng số TF-IDF đã được tính toán cho các term trong cả tài liệu và truy vấn sẽ được sử dụng để xây dựng các vector này, sau đó áp dụng công thức cosine similarity đã trình bày trong Mục 2.1.1.
* **Tính độ liên quan (BM25):** Đối với mô hình Xác suất, cụ thể là BM25, điểm số liên quan được tính toán dựa trên công thức BM25 đã trình bày trong Mục 2.1.2. Công thức này xem xét tần suất của term trong tài liệu, tần suất tài liệu của term và độ dài của tài liệu để đưa ra một điểm số phản ánh xác suất tài liệu liên quan đến truy vấn.
* **Xếp hạng:** Sau khi tính toán điểm số độ liên quan cho tất cả các tài liệu được truy xuất, hệ thống sẽ sắp xếp chúng theo thứ tự giảm dần của điểm số này. Danh sách tài liệu đã được xếp hạng sau đó sẽ được trả về cho người dùng. Các tài liệu có điểm số cao nhất được coi là liên quan nhất và được hiển thị ở đầu danh sách kết quả.

**Ví dụ minh họa quá trình truy xuất và xếp hạng:** Giả sử chúng ta có chỉ mục ngược đã xây dựng ở Mục 2.3.1 và truy vấn Q: "báo cáo thông tin".

1. **Phân tích truy vấn:** Q được phân tích thành các term: {báo cáo, thông tin}.
2. **Tra cứu và kết hợp posting lists:**
   * "báo cáo":
   * "thông tin":
3. **Tính toán độ liên quan (ví dụ với VSM đơn giản):**
   * Term "báo cáo" có trong D1, D3 và Q.
   * Term "thông tin" có trong D1, D2 và Q.
   * D1 chứa cả "báo cáo" và "thông tin".
   * D2 chỉ chứa "thông tin".
   * D3 chỉ chứa "báo cáo". Giả sử sử dụng trọng số đơn giản (ví dụ: 1 nếu có, 0 nếu không):
   * Điểm cho D1: 1 (cho "báo cáo") + 1 (cho "thông tin") = 2
   * Điểm cho D2: 0 (cho "báo cáo") + 1 (cho "thông tin") = 1
   * Điểm cho D3: 1 (cho "báo cáo") + 0 (cho "thông tin") = 1
4. **Xếp hạng:** D1 (điểm 2) sẽ được xếp hạng cao nhất, sau đó là D2 và D3 (cùng điểm 1, thứ tự có thể tùy thuộc vào cách phá vỡ hòa).

Quá trình này đảm bảo rằng các tài liệu phù hợp nhất với nhu cầu của người dùng sẽ được ưu tiên hiển thị, tối đa hóa khả năng người dùng tìm thấy thông tin hữu ích.

**CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM**

Phần này trình bày chi tiết về quá trình cài đặt thử nghiệm hệ thống truy xuất thông tin, bao gồm sơ đồ hệ thống, mô tả các thành phần, và các bước thực hiện thử nghiệm trên bộ ngữ liệu Cranfield cùng với kết quả đánh giá.

**3.1 Sơ đồ hệ thống và Mô tả các thành phần**

Hệ thống truy xuất văn bản được cài đặt theo mô hình chung của một hệ thống IR, bao gồm các giai đoạn tiền xử lý, lập chỉ mục, và truy xuất. Sơ đồ hệ thống dưới đây được lấy cảm hứng từ cấu trúc triển khai của Manwar et al. (2012).

*A screenshot of a computer

AI-generated content may be incorrect.*Sơ đồ hệ thống bao gồm các thành phần chính sau:

**1. Đầu vào (Input):**

* **Bộ ngữ liệu Cranfield:** Đây là tập hợp 1398 bản tóm tắt (abstracts) liên quan đến lĩnh vực khí động học. Dữ liệu này được cung cấp dưới dạng nén và đã được tiền xử lý để chuyển đổi thành 1398 tệp văn bản riêng lẻ bằng một script PHP.
* **Danh sách từ dừng (SMART Stop List):** Một tệp văn bản chứa danh sách các từ dừng (stopword) được sử dụng để loại bỏ các từ không mang ý nghĩa phân biệt khỏi tài liệu và truy vấn.
* **Tập truy vấn (Query Set):** Thay vì sử dụng tập truy vấn có sẵn của bộ Cranfield, nghiên cứu này đã tạo ra tập truy vấn riêng gồm 1398 truy vấn từ các tiêu đề của chính các tài liệu trong bộ sưu tập. Cách tiếp cận này được chọn để đánh giá độ liên quan một cách rõ ràng hơn, vì tài liệu có ID khớp với ID truy vấn được coi là tài liệu liên quan nhất.

**2. Các thành phần chức năng (Functional Components):**

* **Tiền xử lý dữ liệu (Preprocessing):**
  + **Loại bỏ Stopword:** Các từ trong danh sách từ dừng được loại bỏ khỏi cả tài liệu và truy vấn.
  + **Loại bỏ ký tự không nhúng & số:** Các ký tự đặc biệt không phải chữ cái và các chữ số cũng được loại bỏ để chỉ giữ lại các từ chữ cái.
  + **Lưu ý về Stemming:** Trong triển khai này, stemming không được áp dụng. Lý do được đưa ra là để tránh làm mất ngữ cảnh tìm kiếm, có thể làm giảm độ chính xác và không áp dụng được cho danh từ riêng.
* **Tạo từ điển các từ duy nhất (Generate Dictionary of Unique Words):**
  + Từ tất cả các từ còn lại sau tiền xử lý từ bộ sưu tập tài liệu, một từ điển chứa 7805 từ duy nhất được tạo ra. Đây là tập hợp các term sẽ được sử dụng để lập chỉ mục.
* **Tạo ma trận tần suất term (Generation of Term-Frequency Matrix - FM):**
  + Ma trận FM có kích thước M×N, trong đó M là số lượng term duy nhất trong từ điển (7805) và N là số lượng tài liệu (1398).
  + Mỗi phần tử Ai,j​ trong ma trận này biểu thị tần suất xuất hiện của term thứ i trong tài liệu thứ j.
* **Tạo ma trận truy vấn (Generation of Query Matrix - QM):**
  + Tương tự như ma trận tài liệu, ma trận QM được tạo ra cho các truy vấn. Với 1398 truy vấn (từ tiêu đề tài liệu), ma trận này có kích thước 1398×7805.
* **Tính toán trọng số term và kết quả (Term-weight Calculations and Result):**
  + **Tính trọng số TF-IDF:** Ma trận tần suất term (FM) được xử lý để tính toán trọng số term sử dụng lược đồ TF-IDF. Công thức trọng số được sử dụng là tw=(tf×idf)/EL.
    - tfi,j​: Tần suất của term i trong tài liệu j.
    - idf: Tần suất tài liệu ngược, tính bằng log(dj​D​), trong đó D là tổng số tài liệu và dj​ là số tài liệu chứa term j.
    - EL: Độ dài Euclidean, tính bằng căn bậc hai của tổng bình phương các trọng số term riêng lẻ trong mỗi tài liệu.
  + **Chuẩn hóa ma trận truy vấn:** Ma trận truy vấn cũng được chuẩn hóa bằng cách chia cho độ dài Euclidean tương ứng của chúng để có được trọng số chuẩn hóa.
  + **Tính toán độ tương đồng:** Đối với mỗi truy vấn q, ma trận truy vấn chuyển vị được nhân với ma trận trọng số term để tính toán độ tương đồng giữa truy vấn và tài liệu.
  + **Sắp xếp kết quả:** Kết quả cuối cùng là một ma trận điểm số tài liệu (1 x 1398) được sắp xếp theo thứ tự giảm dần của trọng số, hiển thị các tài liệu liên quan nhất ở đầu danh sách.

**3. Đầu ra (Output):**

* Danh sách tài liệu được xếp hạng theo độ liên quan với truy vấn, kèm theo điểm số tương đồng.

Việc sử dụng tiêu đề tài liệu làm truy vấn trong thiết lập thực nghiệm của Manwar et al. (2012) có thể dẫn đến một số ảnh hưởng đến kết quả đánh giá. Mặc dù cách tiếp cận này giúp đơn giản hóa việc xác định tài liệu liên quan (tài liệu có cùng ID với truy vấn tiêu đề), nó có thể không phản ánh đầy đủ sự phức tạp và đa dạng của các truy vấn trong thế giới thực. Các truy vấn được tạo từ tiêu đề thường ngắn gọn và chứa các từ khóa chính, điều này có thể làm tăng độ chính xác (precision) cho các tài liệu khớp chính xác, nhưng lại bỏ qua khả năng của hệ thống trong việc xử lý các truy vấn dài hơn, mơ hồ hơn hoặc các truy vấn ngôn ngữ tự nhiên phức tạp mà người dùng thường nhập vào các công cụ tìm kiếm.

**3.2 Thử nghiệm với bộ ngữ liệu Cranfield**

Thử nghiệm được thực hiện trên bộ ngữ liệu Cranfield nhằm đánh giá hiệu suất của Vector Space Model (VSM) và mô hình BM25, đồng thời so sánh chúng.

**3.2.1 Chuẩn bị dữ liệu**

* **Bộ ngữ liệu Cranfield:** Bộ sưu tập này bao gồm 1398 bản tóm tắt (abstracts) trong lĩnh vực khí động học, được lấy từ máy chủ FTP của Đại học Cornell. Đây là một bộ sưu tập nhỏ nhưng có số lượng truy vấn tương đối lớn (225 truy vấn gốc, nhưng trong nghiên cứu này sử dụng 1398 truy vấn được tạo từ tiêu đề).
* **Tiền xử lý:**
  + Các tệp văn bản thô đã được tiền xử lý để chuyển đổi thành các tệp riêng lẻ.
  + Một danh sách từ dừng SMART (SMART stop word text file) đã được sử dụng để loại bỏ các từ không cần thiết.
  + Các ký tự đặc biệt không phải chữ cái và các chữ số cũng được loại bỏ, chỉ giữ lại các từ chữ cái.
  + **Điểm quan trọng:** Stemming (đưa từ về dạng gốc) **không được áp dụng** trong quá trình tiền xử lý. Lý do được đưa ra là để tránh "làm mất ngữ cảnh tìm kiếm", "có thể làm giảm độ chính xác" và "không thể áp dụng cho danh từ riêng". Điều này cho thấy một sự đánh đổi có chủ đích giữa khả năng bao phủ (recall) và độ chính xác (precision) trong thiết kế hệ thống, ưu tiên việc duy trì ngữ nghĩa nguyên bản của từ.
* **Tập truy vấn:** Để đánh giá độ liên quan một cách rõ ràng hơn, thay vì sử dụng tập truy vấn gốc, 1398 truy vấn đã được tạo ra từ chính các tiêu đề của tài liệu trong bộ sưu tập Cranfield. Trong nghiên cứu này, tài liệu được coi là liên quan nhất nếu ID của nó khớp với ID của truy vấn.

**3.2.2 Cấu hình mô hình và tham số**

Để đạt được hiệu suất tối ưu, các mô hình VSM và BM25 đã được điều chỉnh siêu tham số (hyperparameter tuning) dựa trên dữ liệu thực nghiệm. Việc điều chỉnh siêu tham số là một bước quan trọng trong tối ưu hóa mô hình, vì các giá trị tham số khác nhau có thể ảnh hưởng đáng kể đến hiệu suất cuối cùng.

* **Mô hình Vector Space Model (VSM):**

A black text on a white background

AI-generated content may be incorrect.

* + Quá trình điều chỉnh siêu tham số cho TF-IDF Vectorizer đã hoàn tất.
  + Giá trị Mean Average Precision (MAP) tốt nhất cho VSM là **0.1290** với các tham số: min\_df=1 và max\_df=1.0.
  + Các tham số cuối cùng cho TF-IDF Vectorizer được đặt là: min\_df=1, max\_df=1.0.
  + Ma trận TF-IDF được tạo có kích thước: (1400, 4211).
* **Mô hình BM25:**

**A close-up of a white background

AI-generated content may be incorrect.**

* + Quá trình điều chỉnh siêu tham số cho BM25 cũng đã hoàn tất.
  + Giá trị MAP tốt nhất cho BM25 là **0.1358** với các tham số: k1=2.0 và b=0.75.
  + Các tham số cuối cùng cho mô hình BM25 được đặt là: k1=2.0, b=0.75.

Việc xác định các tham số tối ưu như min\_df, max\_df cho VSM và k1, b cho BM25 thông qua quá trình điều chỉnh siêu tham số là rất quan trọng. Điều này cho thấy tính chất thực nghiệm của việc tối ưu hóa các mô hình IR. Các giá trị này không chỉ ảnh hưởng đến cách trọng số của term được tính toán mà còn tác động trực tiếp đến khả năng phân biệt và xếp hạng tài liệu của mô hình, từ đó quyết định hiệu suất tổng thể của hệ thống.

**3.2.3 Mã nguồn thực nghiệm (Python)**

Phần này sẽ chỉ ra các đoạn mã nguồn chính từ file ipynb đã được cung cấp, minh họa các bước cài đặt thử nghiệm cho VSM và BM25. Các đoạn mã này là xương sống của hệ thống, thực hiện từ tiền xử lý dữ liệu đến tính toán độ tương đồng và đánh giá hiệu suất.

**1. Tiền xử lý dữ liệu và tạo ma trận TF-IDF (VSM):** Đoạn mã này sẽ bao gồm các bước đọc dữ liệu Cranfield, làm sạch văn bản (loại bỏ stopword, ký tự đặc biệt), và sau đó sử dụng TfidfVectorizer từ thư viện sklearn để tạo ma trận TF-IDF.

Python

A screenshot of a computer code

AI-generated content may be incorrect.**2. Cài đặt BM25:** BM25 không có sẵn trực tiếp trong sklearn như TfidfVectorizer, thường cần một triển khai tùy chỉnh hoặc sử dụng thư viện chuyên biệt. Đoạn mã dưới đây minh họa cấu trúc của một triển khai BM25, bao gồm tính toán IDF và sau đó tính điểm BM25 cho từng tài liệu.

Python

A screenshot of a computer program

AI-generated content may be incorrect.

**3. Tính toán độ tương đồng Cosine (VSM):** Sau khi có ma trận TF-IDF, độ tương đồng cosine có thể được tính toán giữa các vector truy vấn và tài liệu.

Python

A computer screen shot of a program code

AI-generated content may be incorrect.**4. Tính toán độ tương đồng BM25:** Sử dụng lớp BM25 đã định nghĩa để tính điểm số giữa truy vấn và tài liệu.

Python

A screenshot of a computer code

AI-generated content may be incorrect.

**5. Đánh giá hiệu suất (Precision, Recall, MAP):** Đoạn mã này sẽ minh họa cách tính toán các độ đo hiệu suất như Precision, Recall và MAP, dựa trên kết quả xếp hạng và ground truth (tài liệu liên quan).

Python

**VSM**

A screenshot of a computer program

AI-generated content may be incorrect.

**BM25**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**3.3 Kết quả đánh giá trên Cranfield**

A screenshot of a computer

AI-generated content may be incorrect.Sau khi thực hiện các thử nghiệm trên bộ ngữ liệu Cranfield, các kết quả đánh giá hiệu suất cho các mô hình Vector Space Model (VSM), BM25 và Semantic Search Model đã được tổng hợp. Các độ đo chính được sử dụng là Mean Precision, Mean Recall và Mean Average Precision (MAP), cùng với Precision nội suy 11 điểm. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.**A screenshot of a computer

AI-generated content may be incorrect.**

**Bảng 3.1: Kết quả đánh giá tổng hợp trên ngữ liệu Cranfield**

| Mô hình | Mean Precision | Mean Recall | Mean Average Precision (MAP) |
| --- | --- | --- | --- |
| Vector Space Model (VSM) | 0.0072 | 0.8426 | 0.1290 |
| BM25 Model | 0.0074 | 0.8261 | 0.1358 |
| Semantic Search Model | 0.0052 | 0.9840 | 0.1269 |

**Phân tích kết quả:**

Từ Bảng 3.1, có thể thấy một số điểm đáng chú ý:

* **So sánh VSM và BM25:** Mô hình BM25 cho thấy hiệu suất MAP tốt hơn một chút so với VSM (0.1358 so với 0.1290). Điều này cho thấy rằng BM25, với khả năng điều chỉnh độ bão hòa của tần suất từ và ảnh hưởng của độ dài tài liệu thông qua các tham số k1​ và b, có thể xử lý các từ hiếm và độ dài tài liệu hiệu quả hơn trong việc xếp hạng độ liên quan. Mặc dù sự khác biệt về MAP không quá lớn, nó vẫn khẳng định BM25 là một thuật toán mạnh mẽ và thường vượt trội hơn VSM trong các tác vụ IR truyền thống.
* **Precision và Recall của VSM và BM25:** Cả hai mô hình VSM và BM25 đều có Mean Precision rất thấp (0.0072 và 0.0074) nhưng lại có Mean Recall khá cao (0.8426 và 0.8261). Điều này có thể được giải thích bởi một số yếu tố. Thứ nhất, bản chất của bộ ngữ liệu Cranfield có thể có nhiều tài liệu liên quan cho mỗi truy vấn, và hệ thống có khả năng tìm thấy phần lớn chúng (recall cao). Tuy nhiên, việc Mean Precision thấp có thể chỉ ra rằng trong số các tài liệu được truy xuất, tỷ lệ tài liệu thực sự liên quan ở các vị trí đầu (top-k) là không cao. Điều này có thể do cách tính toán độ đo hoặc do hệ thống trả về một lượng lớn tài liệu, trong đó chỉ một phần nhỏ là thực sự liên quan ở các thứ hạng cao.
* **Mô hình Tìm kiếm Ngữ nghĩa:** Mô hình tìm kiếm ngữ nghĩa (sử dụng Sentence Embeddings) có MAP thấp nhất (0.1269) trong số ba mô hình được thử nghiệm. Tuy nhiên, nó lại đạt Mean Recall cao nhất một cách đáng kể (0.9840). Điều này cho thấy rằng mô hình ngữ nghĩa có khả năng bao phủ rất tốt, tức là nó có thể tìm thấy gần như tất cả các tài liệu liên quan trong bộ sưu tập. Tuy nhiên, Precision rất thấp (0.0052) và MAP thấp cho thấy mô hình này gặp khó khăn trong việc xếp hạng các tài liệu liên quan một cách chính xác ở các vị trí đầu của danh sách kết quả. Điều này có thể xảy ra khi độ tương đồng ngữ nghĩa rộng hơn so với độ liên quan từ khóa cụ thể mà người dùng mong đợi, dẫn đến việc trả về nhiều tài liệu "có liên quan ngữ nghĩa" nhưng không phải là "liên quan trực tiếp" theo tiêu chí đánh giá truyền thống.

**Precision nội suy 11 điểm (Recall Levels 0.0 to 1.0):**

| Mức Recall | Precision VSM | Precision BM25 | Precision Semantic |
| --- | --- | --- | --- |
| 0.0 | 0.3460 | 0.3627 | 0.3307 |
| 0.1 | 0.3250 | 0.3388 | 0.3168 |
| 0.2 | 0.2623 | 0.2814 | 0.2500 |
| 0.3 | 0.1963 | 0.2119 | 0.1930 |
| 0.4 | 0.1577 | 0.1679 | 0.1547 |
| 0.5 | 0.1325 | 0.1446 | 0.1280 |
| 0.6 | 0.0654 | 0.0688 | 0.0684 |
| 0.7 | 0.0427 | 0.0430 | 0.0468 |
| 0.8 | 0.0298 | 0.0301 | 0.0384 |
| 0.9 | 0.0123 | 0.0125 | 0.0188 |
| 1.0 | 0.0102 | 0.0099 | 0.0157 |

Biểu đồ Precision nội suy 11 điểm cung cấp cái nhìn chi tiết hơn về hiệu suất của các mô hình ở các mức độ recall khác nhau.

* Ở các mức recall thấp (0.0 đến 0.5), BM25 liên tục cho thấy Precision cao hơn VSM, củng cố nhận định về khả năng xếp hạng tốt hơn của nó ở các vị trí đầu.
* Mô hình ngữ nghĩa (Semantic Search) có Precision thấp hơn VSM và BM25 ở các mức recall thấp (0.0-0.5), nhưng lại có Precision cao hơn ở các mức recall rất cao (0.8-0.9), và thậm chí cao hơn VSM ở recall 1.0. Điều này một lần nữa khẳng định rằng mô hình ngữ nghĩa có khả năng tìm thấy nhiều tài liệu liên quan hơn, nhưng lại gặp khó khăn trong việc đưa chúng lên các thứ hạng cao.

Nhìn chung, kết quả thực nghiệm trên bộ ngữ liệu Cranfield cho thấy BM25 là mô hình mạnh mẽ hơn VSM một chút trong việc xếp hạng độ liên quan, trong khi mô hình ngữ nghĩa có tiềm năng về độ phủ nhưng cần cải thiện đáng kể về khả năng xếp hạng chính xác ở các vị trí đầu.

**CHƯƠNG 4: SO SÁNH VÀ ĐÁNH GIÁ**

Phần này sẽ so sánh chi tiết giữa các mô hình đã được thử nghiệm, phân tích ưu nhược điểm của chúng, và đưa ra đánh giá tổng thể về hiệu suất.

**4.1 So sánh VSM và BM25**

Dựa trên kết quả thực nghiệm và lý thuyết nền tảng, có thể phân tích ưu nhược điểm của VSM và BM25 như sau:

**Mô hình Vector Space Model (VSM):**

* **Ưu điểm:**
  + **Dễ hiểu và trực quan:** VSM biểu diễn tài liệu và truy vấn dưới dạng vector trong không gian Euclide, cho phép hình dung và tính toán độ tương đồng một cách trực quan thông qua góc cosine.
  + **Hiệu quả với TF-IDF:** Khi kết hợp với lược đồ trọng số TF-IDF, VSM đã chứng minh được hiệu quả trong nhiều tác vụ truy xuất thông tin, đặc biệt là với các tài liệu có độ dài tương đối đồng đều.
  + **Linh hoạt:** Có thể dễ dàng tích hợp các lược đồ trọng số khác nhau và các độ đo tương đồng khác ngoài cosine.
* **Nhược điểm:**
  + **Bỏ qua ngữ nghĩa và mối quan hệ giữa các từ:** VSM coi mỗi term là một chiều độc lập trong không gian vector, bỏ qua các mối quan hệ ngữ nghĩa (như từ đồng nghĩa, từ đa nghĩa) và thứ tự từ trong văn bản. Điều này có thể dẫn đến việc bỏ lỡ các tài liệu liên quan nếu chúng sử dụng từ đồng nghĩa hoặc các cấu trúc câu khác nhau.
  + **Giả định độc lập term:** Mô hình giả định rằng các term là độc lập với nhau, điều này hiếm khi đúng trong ngôn ngữ tự nhiên.
  + **Nhạy cảm với từ đồng nghĩa/đa nghĩa:** Nếu truy vấn sử dụng một từ đồng nghĩa với từ trong tài liệu, VSM có thể không tìm thấy tài liệu đó. Ngược lại, một từ đa nghĩa có thể dẫn đến việc truy xuất các tài liệu không liên quan.
  + **Vấn đề chiều dữ liệu:** Với bộ từ vựng lớn, không gian vector có thể trở nên rất thưa thớt và có chiều cao, gây ra các vấn đề về hiệu suất và tính toán.

**Mô hình Xác suất (BM25):**

* **Ưu điểm:**
  + **Thường vượt trội VSM trong các tác vụ IR truyền thống:** Như đã thấy trong kết quả thực nghiệm trên Cranfield, BM25 cho hiệu suất MAP nhỉnh hơn VSM. Điều này là do BM25 được thiết kế để xử lý tốt hơn các vấn đề như độ bão hòa của tần suất term (term frequency saturation) và ảnh hưởng của độ dài tài liệu (document length normalization).
  + **Tính đến tần suất term, tần suất tài liệu và độ dài tài liệu:** BM25 tích hợp một cách tinh vi các yếu tố này vào công thức tính điểm, giúp nó đưa ra các xếp hạng chính xác hơn. Các tham số k1​ và b cho phép điều chỉnh mức độ ảnh hưởng của tần suất term và độ dài tài liệu, giúp mô hình linh hoạt hơn và thích nghi tốt hơn với các bộ dữ liệu khác nhau.
  + **Cơ sở lý thuyết vững chắc hơn:** Mặc dù là heuristic, BM25 có nguồn gốc từ mô hình xác suất nhị phân độc lập, cung cấp một nền tảng lý thuyết cho các thành phần của nó.
* **Nhược điểm:**
  + **Phức tạp hơn VSM:** Công thức của BM25 phức tạp hơn và có các tham số cần điều chỉnh (k1​, b), đòi hỏi quá trình hiệu chỉnh để đạt được hiệu suất tối ưu.
  + **Vẫn bỏ qua ngữ nghĩa:** Tương tự VSM, BM25 vẫn là một mô hình dựa trên từ khóa, không trực tiếp xử lý ngữ nghĩa hoặc mối quan hệ phức tạp giữa các từ.
  + **Giả định độc lập term:** Mặc dù có những cải tiến, BM25 vẫn giữ giả định cơ bản về sự độc lập của các term.

**Phân tích sâu hơn về hiệu suất tương đối:** Mặc dù BM25 cho thấy hiệu suất MAP cao hơn VSM trên bộ ngữ liệu Cranfield, sự khác biệt này không quá lớn (0.1358 so với 0.1290). Điều này có thể được giải thích bằng một số yếu tố. Thứ nhất, bộ dữ liệu Cranfield có thể không đủ lớn hoặc đủ phức tạp để thể hiện rõ rệt toàn bộ ưu điểm của BM25 so với VSM. Trên các bộ dữ liệu lớn hơn và đa dạng hơn (như các bộ TREC hiện đại), BM25 thường thể hiện sự vượt trội rõ ràng hơn. Thứ hai, cả hai mô hình đều là các phương pháp dựa trên từ khóa, và trên một bộ dữ liệu có đặc điểm từ vựng nhất định, chúng có thể đạt đến một ngưỡng hiệu suất tương tự. Sự khác biệt nhỏ này cũng có thể cho thấy rằng các bước tiền xử lý (đặc biệt là việc không sử dụng stemming) có ảnh hưởng đáng kể đến hiệu suất cuối cùng của cả hai mô hình.

**4.2 So sánh với Whoosh**

Để có cái nhìn toàn diện hơn về hiệu suất và khả năng triển khai, việc so sánh các mô hình tự cài đặt với một thư viện truy xuất thông tin chuyên nghiệp như Whoosh là rất hữu ích.

**Giới thiệu Whoosh:** Whoosh là một thư viện lập chỉ mục và tìm kiếm toàn văn nhanh chóng, giàu tính năng, được triển khai hoàn toàn bằng Python. Nó cung cấp một API Pythonic, cho phép các lập trình viên dễ dàng thêm chức năng tìm kiếm vào ứng dụng và trang web của họ. Whoosh được thiết kế với tính mô đun cao, cho phép mở rộng hoặc thay thế mọi phần của cách nó hoạt động để phù hợp với nhu cầu cụ thể. Các tính năng nổi bật của Whoosh bao gồm:

* **Hoàn toàn bằng Python:** Không yêu cầu biên dịch hoặc các gói nhị phân, giảm thiểu các lỗi không mong muốn.
* **Lập chỉ mục và tìm kiếm theo trường (Fielded indexing and search):** Cho phép tìm kiếm trên các trường cụ thể của tài liệu (ví dụ: tiêu đề, nội dung).
* **Tốc độ nhanh:** Được đánh giá là nhanh hơn bất kỳ giải pháp tìm kiếm toàn văn, tính điểm, thuần Python nào khác.
* **Thuật toán tính điểm có thể cắm được:** Bao gồm cả thuật toán BM25F (một biến thể của BM25). Điều này rất quan trọng vì nó cho phép các nhà nghiên cứu thử nghiệm các công thức tính điểm khác nhau mà không cần xây dựng lại toàn bộ hệ thống.
* **Ngôn ngữ truy vấn mạnh mẽ:** Hỗ trợ các truy vấn phức tạp.
* **Kiểm tra chính tả thuần Python:** Một tính năng hữu ích cho trải nghiệm người dùng.

**So sánh hiệu suất với Whoosh:** Dữ liệu thực nghiệm được cung cấp không bao gồm kết quả đánh giá trực tiếp của Whoosh. Tuy nhiên, việc Maxwell et al. (2016) đề cập Whoosh là một công cụ IR được sử dụng trong framework SimIIR của họ ngụ ý rằng Whoosh là một nền tảng thực tế và đáng tin cậy để triển khai và thử nghiệm các mô hình IR.

* **Ưu điểm của Whoosh so với triển khai thủ công:**
  + **Tối ưu hóa hiệu suất:** Whoosh được phát triển và tối ưu hóa bởi các chuyên gia, có khả năng cung cấp hiệu suất cao hơn (đặc biệt về tốc độ lập chỉ mục và truy xuất) so với một triển khai thủ công, ngay cả khi sử dụng cùng thuật toán cơ bản như BM25. Điều này là do Whoosh có thể sử dụng các kỹ thuật tối ưu hóa cấp thấp, quản lý bộ nhớ hiệu quả và các cấu trúc dữ liệu được thiết kế đặc biệt.
  + **Tính năng phong phú và ổn định:** Whoosh cung cấp một bộ tính năng hoàn chỉnh (tiền xử lý, lập chỉ mục, truy vấn phức tạp, kiểm tra chính tả) và được bảo trì tích cực, giúp giảm đáng kể công sức phát triển và đảm bảo độ ổn định.
  + **Khả năng mở rộng:** Kiến trúc mô đun của Whoosh cho phép dễ dàng mở rộng và tùy chỉnh các thành phần, ví dụ như thay đổi thuật toán tính điểm hoặc phương pháp phân tích văn bản.
* **Điểm mạnh/yếu của các mô hình tự cài đặt so với Whoosh:**
  + **Điểm mạnh của triển khai tự cài đặt:** Việc tự cài đặt VSM và BM25 giúp hiểu sâu sắc hơn về cơ chế hoạt động bên trong của từng mô hình, từ các bước tiền xử lý, tính toán trọng số đến quá trình xếp hạng. Đây là một phần quan trọng của quá trình học tập và nghiên cứu.
  + **Điểm yếu của triển khai tự cài đặt:**
    - **Hiệu suất:** Triển khai thủ công có thể không đạt được hiệu suất tối ưu về tốc độ và khả năng mở rộng như một thư viện được tối ưu hóa như Whoosh, đặc biệt khi xử lý các bộ dữ liệu lớn.
    - **Tính năng:** Các tính năng bổ sung như xử lý truy vấn phức tạp, kiểm tra chính tả, hoặc các chiến lược lập chỉ mục nâng cao thường phải được xây dựng từ đầu.
    - **Độ ổn định và bảo trì:** Một hệ thống tự cài đặt có thể kém ổn định hơn và đòi hỏi nhiều công sức bảo trì hơn so với một thư viện đã được kiểm chứng và cộng đồng hỗ trợ.

Việc so sánh trực tiếp hiệu suất của các mô hình tự cài đặt với Whoosh sẽ cung cấp một cái nhìn thực tế về mức độ tối ưu hóa của cài đặt. Nếu một triển khai BM25 trong Whoosh cho hiệu suất vượt trội đáng kể so với triển khai thủ công, điều đó sẽ chỉ ra rằng có những tối ưu hóa về mặt kỹ thuật (ví dụ: cấu trúc dữ liệu, quản lý tài nguyên) mà triển khai thủ công chưa đạt được.

**4.3 Thảo luận về Mô hình Tìm kiếm Ngữ nghĩa**

Ngoài các mô hình dựa trên từ khóa truyền thống như VSM và BM25, dữ liệu thực nghiệm còn cung cấp kết quả cho một "Mô hình Tìm kiếm Ngữ nghĩa" sử dụng Sentence Embeddings (nhúng câu). Kết quả này mang lại một góc nhìn quan trọng về tiềm năng và thách thức của các phương pháp tìm kiếm dựa trên ngữ nghĩa.

**Kết quả đánh giá của Mô hình Tìm kiếm Ngữ nghĩa (Sentence Embeddings):**

* Mean Precision: 0.0052
* Mean Recall: 0.9840
* Mean Average Precision (MAP): 0.1269
* Precision nội suy 11 điểm:
  + Recall 0.0: 0.3307
  + Recall 0.1: 0.3168
  + ...
  + Recall 1.0: 0.0157

**Phân tích và thảo luận:** Kết quả của mô hình tìm kiếm ngữ nghĩa cho thấy một đặc điểm rất rõ ràng: nó có **Mean Recall cực kỳ cao (0.9840)**, cao hơn đáng kể so với cả VSM (0.8426) và BM25 (0.8261). Điều này chỉ ra rằng mô hình ngữ nghĩa có khả năng bao phủ rất tốt, tức là nó có thể tìm thấy hầu hết các tài liệu thực sự liên quan trong bộ sưu tập. Khả năng này xuất phát từ việc nhúng câu (sentence embeddings) cho phép mô hình hiểu được ý nghĩa ngữ cảnh của từ và cụm từ, vượt ra ngoài sự khớp từ khóa chính xác. Nhờ đó, nó có thể xác định các tài liệu liên quan ngay cả khi chúng không chứa các từ khóa chính xác của truy vấn mà thay vào đó sử dụng các từ đồng nghĩa hoặc cách diễn đạt khác có cùng ý nghĩa.

Tuy nhiên, đi kèm với recall cao là **Mean Precision rất thấp (0.0052)** và **MAP thấp nhất (0.1269)** trong số ba mô hình. Điều này có nghĩa là mặc dù mô hình ngữ nghĩa tìm thấy nhiều tài liệu liên quan, nó lại gặp khó khăn nghiêm trọng trong việc xếp hạng chúng một cách chính xác ở các vị trí đầu của danh sách kết quả. Nói cách khác, trong số các tài liệu mà nó trả về, rất ít tài liệu thực sự liên quan nằm ở các thứ hạng cao nhất mà người dùng thường xem. Điều này có thể xảy ra vì độ tương đồng ngữ nghĩa đôi khi quá "rộng", khiến mô hình coi nhiều tài liệu là "liên quan" theo một nghĩa nào đó, nhưng không phải là liên quan trực tiếp hoặc ưu tiên cao theo nhu cầu cụ thể của truy vấn. Các nhúng câu có thể nắm bắt các mối quan hệ ngữ nghĩa tổng quát, nhưng lại thiếu khả năng phân biệt các sắc thái nhỏ hoặc các từ khóa quan trọng mà người dùng mong đợi.

**Ý nghĩa và hướng đi tiềm năng:** Sự chênh lệch lớn giữa recall cao và precision/MAP thấp của mô hình ngữ nghĩa chỉ ra rằng việc chỉ dựa vào độ tương đồng ngữ nghĩa có thể không đủ để xác định độ liên quan chính xác trong một số ngữ cảnh, đặc biệt khi các từ khóa cụ thể hoặc cấu trúc thông tin là quan trọng. Điều này gợi mở một hướng đi đầy hứa hẹn trong nghiên cứu IR: **kết hợp các phương pháp tìm kiếm từ khóa truyền thống (như VSM/BM25) với tìm kiếm ngữ nghĩa**.

Một hệ thống lai (hybrid system) có thể tận dụng ưu điểm của cả hai:

* Sử dụng các mô hình từ khóa để đảm bảo độ chính xác cao cho các truy vấn có từ khóa rõ ràng và để xếp hạng ban đầu các tài liệu.
* Sử dụng các mô hình ngữ nghĩa để mở rộng độ phủ, tìm kiếm các tài liệu có liên quan ngữ nghĩa nhưng không khớp từ khóa trực tiếp, hoặc để tinh chỉnh thứ hạng của các tài liệu đã được tìm thấy bằng phương pháp từ khóa.

Việc kết hợp này có thể giúp cải thiện đồng thời cả Precision và Recall, mang lại trải nghiệm tìm kiếm toàn diện và chính xác hơn cho người dùng. Đây là một lĩnh vực nghiên cứu tích cực trong IR hiện đại, phản ánh sự phát triển từ các phương pháp dựa trên từ vựng sang các phương pháp hiểu biết ngữ nghĩa sâu sắc hơn.

**CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**5.1 Kết luận**

Nghiên cứu này đã đi sâu vào các vấn đề lý thuyết và thực nghiệm của truy xuất thông tin, tập trung vào hai mô hình kinh điển là Vector Space Model (VSM) và BM25, cùng với một cái nhìn sơ bộ về mô hình tìm kiếm ngữ nghĩa.

Các phân tích lý thuyết đã làm rõ cách tài liệu và truy vấn được biểu diễn, các bước tiền xử lý cần thiết để chuyển đổi văn bản thô thành các term có thể lập chỉ mục, cấu trúc của chỉ mục ngược và quy trình truy xuất thông tin. Việc hiểu rõ định nghĩa term, các bước tokenization, stopword removal và stemming (hoặc lý do không áp dụng) là nền tảng để xây dựng một hệ thống IR hiệu quả. Chỉ mục ngược được xác định là cấu trúc lưu trữ tối ưu, cho phép truy xuất nhanh chóng và linh hoạt.

Về mặt thực nghiệm, hệ thống truy xuất văn bản đã được cài đặt và thử nghiệm trên bộ ngữ liệu Cranfield. Kết quả cho thấy mô hình BM25 đạt Mean Average Precision (MAP) cao hơn một chút so với VSM (0.1358 so với 0.1290). Điều này khẳng định BM25, với khả năng điều chỉnh các tham số k1​ và b để xử lý hiệu quả tần suất term và độ dài tài liệu, thường cho hiệu suất xếp hạng tốt hơn trong các tác vụ IR truyền thống. Tuy nhiên, cả hai mô hình đều có Mean Precision rất thấp nhưng Mean Recall cao, cho thấy khả năng tìm thấy nhiều tài liệu liên quan nhưng gặp khó khăn trong việc đưa chúng lên các thứ hạng đầu.

Đặc biệt, mô hình tìm kiếm ngữ nghĩa (sử dụng Sentence Embeddings) đã thể hiện Mean Recall vượt trội (0.9840), chứng tỏ khả năng bao phủ rộng và tìm thấy hầu hết các tài liệu liên quan dựa trên ý nghĩa ngữ cảnh. Tuy nhiên, MAP và Mean Precision thấp của nó cho thấy mô hình này còn yếu trong việc xếp hạng chính xác các tài liệu liên quan ở các vị trí đầu. Điều này nhấn mạnh rằng việc chỉ dựa vào ngữ nghĩa có thể chưa đủ để đáp ứng nhu cầu xếp hạng chính xác trong IR.

Tổng kết lại, việc lựa chọn mô hình và các bước tiền xử lý dữ liệu đóng vai trò then chốt trong hiệu suất của hệ thống IR. BM25 cho thấy sự nhỉnh hơn về khả năng xếp hạng so với VSM trên Cranfield, nhưng cả hai đều có những điểm mạnh và hạn chế riêng. Sự xuất hiện của mô hình ngữ nghĩa với recall cao mở ra tiềm năng lớn nhưng cũng đặt ra thách thức về precision và xếp hạng.

**5.2 Hướng phát triển**

Để nâng cao hơn nữa hiệu suất và khả năng ứng dụng của hệ thống truy xuất thông tin, một số hướng phát triển tiềm năng có thể được xem xét:

1. **Thu thập nội dung trang web trên Internet để lập chỉ mục và truy xuất các trang web:** Mở rộng phạm vi của bộ sưu tập tài liệu từ một tập dữ liệu cố định như Cranfield sang dữ liệu web động và khổng lồ. Điều này sẽ đòi hỏi các kỹ thuật thu thập dữ liệu (web crawling), xử lý dữ liệu phi cấu trúc và bán cấu trúc, cũng như các chiến lược lập chỉ mục phân tán để xử lý quy mô lớn.
2. **Xây dựng website cho phép truy xuất văn bản:** Phát triển một giao diện người dùng dựa trên web sẽ biến hệ thống từ một công cụ nghiên cứu thành một ứng dụng thực tế, cho phép người dùng tương tác trực tiếp và trải nghiệm quá trình tìm kiếm. Điều này bao gồm thiết kế giao diện người dùng (UI), tích hợp backend với các mô hình IR đã cài đặt, và đảm bảo hiệu suất phản hồi nhanh chóng.
3. **Áp dụng phương pháp phân lớp và/hoặc phương pháp gom cụm để tăng độ phủ của kết quả truy xuất:**
   * **Phân lớp (Classification):** Phân loại tài liệu vào các chủ đề hoặc danh mục cụ thể. Khi người dùng đưa ra truy vấn, hệ thống có thể sử dụng thông tin phân lớp để giới hạn tìm kiếm trong các danh mục liên quan, hoặc để đề xuất các danh mục liên quan khác, từ đó tăng độ chính xác và độ phủ.
   * **Gom cụm (Clustering):** Nhóm các tài liệu tương tự lại với nhau. Kết quả truy xuất có thể được trình bày theo cụm, giúp người dùng dễ dàng khám phá các tài liệu liên quan theo các chủ đề phụ, ngay cả khi chúng không khớp chính xác với từ khóa ban đầu, từ đó tăng độ phủ và khả năng tìm kiếm thông tin mới.
4. **Sử dụng học máy để tính toán độ tương đồng giữa tài liệu và câu truy vấn:** Đây là một hướng đi rất hứa hẹn, đặc biệt sau khi quan sát kết quả của mô hình tìm kiếm ngữ nghĩa.
   * **Học để xếp hạng (Learning to Rank - LTR):** Thay vì sử dụng các công thức cố định như BM25 hay VSM, LTR sử dụng các thuật toán học máy (ví dụ: Gradient Boosting, Neural Networks) để học một hàm xếp hạng từ các cặp truy vấn-tài liệu và nhãn liên quan. Điều này cho phép hệ thống tự động tối ưu hóa các trọng số và kết hợp nhiều yếu tố (tần suất từ, ngữ nghĩa, độ dài tài liệu, v.v.) để đưa ra thứ hạng tốt nhất.
   * **Mô hình nhúng (Embeddings) tiên tiến:** Khám phá các mô hình nhúng từ và câu phức tạp hơn (ví dụ: BERT, RoBERTa, GPT embeddings) để nắm bắt ngữ nghĩa tài liệu và truy vấn một cách sâu sắc hơn. Việc này có thể giúp cải thiện đáng kể khả năng xếp hạng của các mô hình ngữ nghĩa, giải quyết vấn đề precision thấp đã quan sát.
   * **Kết hợp mô hình (Hybrid Approaches):** Phát triển các hệ thống lai kết hợp sức mạnh của tìm kiếm từ khóa truyền thống (như BM25) với khả năng hiểu ngữ nghĩa của học máy. Ví dụ, sử dụng BM25 để tạo ra một tập hợp tài liệu ứng viên ban đầu, sau đó sử dụng một mô hình học máy dựa trên ngữ nghĩa để tinh chỉnh thứ hạng của các tài liệu đó.

Các hướng phát triển này phản ánh xu hướng hiện đại trong lĩnh vực Truy xuất Thông tin, nơi các hệ thống đang dịch chuyển từ các phương pháp tĩnh, dựa trên từ khóa sang các hệ thống động, tương tác và thông minh hơn, tận dụng tối đa các kỹ thuật học máy tiên tiến để nâng cao hiệu suất và đáp ứng nhu cầu thông tin ngày càng phức tạp của người dùng.

**TÀI LIỆU THAM KHẢO**

[1] Manwar, A. B., Mahalle, H. S., Chinchkhede, K. D., & Chavan, V. (n.d.). A VECTOR SPACE MODEL FOR INFORMATION RETRIEVAL: A MATLAB APPROACH. *Indian Journal of Computer Science and Engineering (IJCSE)*. (Tài liệu này được tham khảo dưới dạng INDJCSE12-03-02-028.pdf)

[2] Hiemstra, D. (2000). A probabilistic justification for using tfxidf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2), 131-139. (Tài liệu này được tham khảo dưới dạng hiemstra2000.pdf)

[3] Maxwell, D., & Azzopardi, L. (n.d.). SimIIR: A Framework for the Simulation of Interaction. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. (Tài liệu này được tham khảo dưới dạng maxwell2016.pdf)