ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

ĐỒ ÁN CUỐI KỲ

Latent Semantic Analysis và Ứng dụng trong Information Retrieval

Môn học: Phương pháp số cho Khoa học dữ liệu
GVHD: TS. Nguyễn Thị Hoài Thương
Nhóm: 6

STT	Họ và tên	MSSV	Mức độ hoàn thành (%)
1	Đặng Minh Phúc	22280064	100
2	Nguyễn Minh Hùng	21110301	100
3	Trương Quốc Trung	21110427	100
4	Lê Hồng Cát	21110249	100
5	Trương Minh Hoàng	22280034	100
6	Trần Nguyễn Trung Tuấn	22280101	100
7	Nguyễn Thuận Phát	22280062	100

Thành phố Hồ Chí Minh, tháng 07/2025

Mục lục

1 Giới thiệu

Hình 1: Vấn đề đặt ra của bài toán

Truy xuất thông tin (Information Retrieval - IR) là một bài toán trung tâm trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) và khoa học máy tính. Mục tiêu chính của IR là tìm kiếm và truy xuất các tài liệu hoặc đoạn văn bản có liên quan đến một truy vấn đầu vào từ phía người dùng.

Trong bối cảnh lượng thông tin về phim ngày càng trở nên phong phú và đa dạng, bao gồm mô tả phim, đánh giá, thể loại, tóm tắt nội dung và nhận xét từ người dùng, nhu cầu tìm kiếm phim dựa trên những mô tả ngắn gọn, đôi khi không đầy đủ, ngày càng trở nên thiết thực. Ai trong chúng ta chắc hẳn cũng từng bắt gặp một đoạn trích hấp dẫn của một bộ phim nào đó nhưng lại không thể nhớ tên phim (Hình 1).

Từ thực tế này, nhóm chúng tôi đề xuất xây dựng một hệ thống hỗ trợ người dùng tìm kiếm tên phim dựa trên mô tả nội dung. Nhờ việc dữ liệu phim thường được lưu trữ dưới dạng văn bản, các kỹ thuật IR trở nên rất phù hợp để giải quyết bài toán này. Tuy nhiên, khác với IR truyền thống chỉ dựa trên từ khóa, hệ thống cần hiểu được ngữ nghĩa tiềm ẩn bên trong mô tả phim và truy vấn người dùng, điều này đòi hỏi những phương pháp xử lý ngôn ngữ hiệu quả hơn.

Một trong những phương pháp kinh điển nhưng vẫn mang lại hiệu quả nhất định trong việc khai thác ngữ nghĩa tiềm ẩn là Phân tích ngữ nghĩa tiềm ẩn (Latent Semantic Analysis - LSA), một kỹ thuật học không giám sát dựa trên nền tảng đại số tuyến tính.

Trong bài báo cáo này, chúng tôi sẽ trình bày chi tiết nguyên lý hoạt động của LSA, cách ứng dụng kỹ thuật này vào bài toán truy xuất phim từ mô tả, và đánh giá hiệu quả của mô hình trên tập dữ liệu thực tế.

2 Cơ sở lý thuyết

2.1 Ma trận term-document

Ma trận term-document là dạng biểu diễn ma trận của tập hợp tài liệu văn bản, trong đó mỗi hàng tương ứng với một từ (*term*), mỗi cột tương ứng với một tài liệu (*document*), và mỗi phần tử biểu thị trọng số của từ trong tài liệu tương ứng. Ký hiệu tổng quát của ma trận:

$$X \in \mathbb{R}^{n \times N}$$

- n: số lượng từ (kích thước từ vựng)
- N: số lượng tài liệu
- x_{ij}: là phần tử của X, độ đo liên quan giữa từ t_i và tài liệu d_j

Như vậy, mỗi hàng của ma trận term-document tương ứng với cách biểu diễn của mỗi từ, mỗi cột tương ứng với cách biểu diễn của mỗi tài liệu, và ma trận này có tính chất thưa. Một phiên bản chuyển vị của nó là ma trận document-term, với mỗi hàng tương ứng với số tài liệu, mỗi cột tương ứng với số từ vựng duy nhất.

2.2 Term Frequency - Inverse Document Frequence

Trong không gian vector ngữ liệu, mỗi tài liệu d_j được biểu diễn như một vector trong không gian $\mathbb{R}^n$, trong đó n là kích thước từ vựng duy nhất trong tập dữ liệu. Mỗi thành phần của vector này tương ứng với một từ t_i trong từ vựng, và trọng số của nó được tính bằng Term Frequency - Inverse Document Frequency (TF-IDF).

Cho tập văn bản $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ và từ vựng $V = \{t_1, t_2, \ldots, t_n\}$, ta định nghĩa:

- Hàm tần suất chuẩn hóa - term frequency (TF):

$$\mathrm{TF}(t_i, d_j) = \frac{f_{t_i,d_j}}{\sum_{k=1}^{n} f_{t_k,d_j}} \tag{1}$$

- Hàm nghịch đảo tần suất tài liệu - inverse document frequency (IDF):

$$\mathrm{IDF}(t_i) = \log\left(\frac{N}{df(t_i)}\right), \quad df(t_i) = |\{d_j \in \mathcal{D} : t_i \in d_j\}| \tag{2}$$

- Trọng số TF-IDF:

$$w_{ij} = \text{TF-IDF}(t_i, d_j) = \mathrm{TF}(t_i, d_j) \cdot \mathrm{IDF}(t_i) \tag{3}$$

Toàn bộ tập dữ liệu được biểu diễn thành ma trận $A \in \mathbb{R}^{n \times N}$, trong đó hàng i là từ t_i, cột j là tài liệu d_j, và phần tử $a_{ij} = w_{ij}$.

2.3 Phân rã giá trị kỳ dị

Cho một ma trận thực $A \in \mathbb{R}^{m \times n}$, tồn tại phép phân rã giá trị kỳ dị - Singular Value Decomposition (SVD) [1]:

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

- $U \in \mathbb{R}^{m \times m}$: ma trận trực chuẩn chứa các vector riêng trái (left singular vectors), là các vector riêng của $AA^\top$.
- $V \in \mathbb{R}^{n \times n}$: ma trận trực chuẩn chứa các vector riêng phải (right singular vectors), là các vector riêng của $A^\top A$.
- $\Sigma \in \mathbb{R}^{m \times n}$: ma trận đường chéo chứa các giá trị kỳ dị $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0, r = \min(m, n)$, là căn bậc hai của các trị riêng không âm của cả $AA^\top$ và $A^\top A$.

Tuy nhiên, với các bài toán thực tế, chỉ các giá trị kỳ dị đầu tiên chiếm phần lớn thông tin của ma trận. Khi này, ta có thể xấp xỉ ma trận A bằng cách giữ lại $k < r$ thành phần lớn nhất:

$$A \approx A_k = U_k \Sigma_k V_k^\top$$

với $U_k \in \mathbb{R}^{m \times k}, \Sigma_k \in \mathbb{R}^{k \times k}, V_k \in \mathbb{R}^{n \times k}$. Phương pháp này được gọi là Phân rã kì dị bị cắt ngắn - Truncated SVD hay xấp xỉ hạng thấp. Phần trăm lượng thông tin mà A_k giữ lại được tính như sau:

$$\frac{\|A_k\|_F^2}{\|A\|_F^2} = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{r} \lambda_i^2}$$

Giá trị k có thể được điều chỉnh để giữ lại lượng thông tin mong muốn. Nhờ vậy, Truncated SVD trở thành một công cụ mạnh mẽ để giảm chiều dữ liệu, nén thông tin, và khử nhiễu trong các hệ thống học máy và khai phá dữ liệu.

2.4 Ma trận đồng xuất hiện (Co-occurrence Matrix)

Ma trận đồng xuất hiện là một biểu diễn phân bố tần suất, trong đó mỗi phần tử $C_{i,j}$ biểu diễn số lần từ w_i và w_j cùng xuất hiện trong một ngữ cảnh nhất định (thường là cửa sổ trượt quanh từ mục tiêu).

Giả sử có một tập từ vựng $V = \{w_1, w_2, \ldots, w_n\}$, ta xây dựng một ma trận $C \in \mathbb{R}^{n \times n}$, trong đó:

$$C_{i,j} = \text{số lần từ } w_i \text{ và } w_j \text{ xuất hiện trong cùng một ngữ cảnh}$$

Ngữ cảnh có thể là một câu, một tài liệu, một cửa sổ có một cửa sổ kích thước k từ,. Ngoài ra, có thể áp dụng trọng số cho các đồng xuất hiện, ví dụ: càng gần nhau thì trọng số càng lớn ($\frac{1}{|i-j|^{0.75}}$), để phản ánh đúng mức độ gắn kết ngữ nghĩa.

2.5 Pointwise Mutual Information (PMI)

Pointwise Mutual Information đo mức độ bất ngờ của việc hai từ cùng xuất hiện, so với giả định chúng độc lập:

$$PMI(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)}$$

- $P(w_i, w_j) = \frac{C_{i,j}}{N}$: xác suất đồng xuất hiện.
- $P(w_i) = \frac{\sum_j C_{i,j}}{N}$, $P(w_j) = \frac{\sum_i C_{i,j}}{N}$: xác suất biên.
- $N = \sum_{i,j} C_{i,j}$: tổng số lần đồng xuất hiện.

Giá trị PMI dương phản ánh mối liên hệ ngữ nghĩa mạnh. Tuy nhiên, nếu từ rất hiếm, PMI có thể rất lớn không thực tế. Đồng thời, các giá trị âm hoặc gần 0 thường không hữu ích trong học máy. Vì thế, ta dùng phiên bản không âm của PMI chính là Positive PMI. Phiên bản này giúp loại bỏ các liên hệ yếu hoặc tiêu cực bằng cách lấy giá trị dương của PMI:

$$PPMI(w_i, w_j) = \max(0,\ PMI(w_i, w_j)) = \max\left(0,\ \log 2 \frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)}\right) \tag{4}$$

Mục tiêu là giữ lại các cặp từ có khả năng đồng xuất hiện cao hơn kỳ vọng ngẫu nhiên, phản ánh liên kết ngữ nghĩa tích cực.

2.6 Phân tích thành phần chính (Principal Component Analysis - PCA)

Giả sử ta có ma trận dữ liệu $X \in \mathbb{R}^{n \times d}$ đã được chuẩn hóa, trong đó n là số lượng điểm dữ liệu và d là số đặc trưng. Mỗi hàng của X là một vector dữ liệu d-chiều. Mục tiêu của PCA là tìm ra một tập hợp mới gồm k trục tọa độ trực giao, sao cho khi chiếu dữ liệu lên không gian mới này, ta giữ lại được càng nhiều phương sai của dữ liệu gốc càng tốt.

Trước hết, cần tìm ma trận hiệp phương sai thể hiện mức độ tương quan tuyến tính giữa các chiều dữ liệu, được tính bằng:

$$S = \frac{1}{n} X^\top \tilde{X} \tag{5}$$

Trong đó $S \in \mathbb{R}^{d \times d}$ là một ma trận đối xứng và bán xác định dương. Sau đó, thực hiện phân tích trị riêng trên ma trận S để tìm các vector riêng và trị riêng:

$$S v_i = \lambda_i v_i, \quad i = 1, \ldots, d$$

Trong đó λ_i là trị riêng tương ứng với vector riêng v_i. Các trị riêng biểu thị phương sai dữ liệu khi chiếu lên trục v_i. Sắp xếp các trị riêng theo thứ tự giảm dần, và chọn k trị riêng lớn nhất và lấy các vector riêng tương ứng để tạo thành ma trận chiếu $W_k \in \mathbb{R}^{d \times k}$. Số chiều k có thể được chọn dựa trên tỷ lệ phương sai mong muốn giữ lại:

$$\text{Explained Variance Ratio} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

Dữ liệu ban đầu sau khi được chiếu vào không gian k-chiều mới được tính bằng:

$$X_{\text{new}} = X W_k$$

Mỗi hàng của $X_{\text{new}} \in \mathbb{R}^{n \times k}$ là một biểu diễn rút gọn của điểm dữ liệu gốc trong không gian thành phần chính.

Một cách khác để thực hiện PCA hiệu quả hơn là sử dụng phân rã giá trị suy biến. Thay vì tính ma trận hiệp phương sai, ta áp dụng trực tiếp SVD lên ma trận dữ liệu X (sau khi đã chuẩn hóa), dữ liệu chiếu vào không gian mới có thể được tính bằng biểu thức

$$X_{\text{new}} = X V_k$$

Việc sử dụng SVD giúp PCA ổn định và chính xác hơn, đặc biệt khi số chiều đặc trưng d lớn hoặc dữ liệu phân tán. Đây cũng là phương pháp được sử dụng trong các thư viện học máy hiện đại.

2.7 Độ đo tương đồng Cosine

Độ đo tương đồng Cosine (Cosine Similarity) là một phương pháp phổ biến để đo lường mức độ tương đồng giữa hai vector trong không gian vector, thường được áp dụng trong các bài toán xử lý ngôn ngữ tự nhiên và truy xuất thông tin. Thay vì dựa vào độ lớn tuyệt đối, cosine tập trung vào góc giữa hai vector, phản ánh mức độ định hướng tương đối của chúng.

Công thức tính độ đo cosine giữa hai vector $\vec{A}$ và $\vec{B}$ được định nghĩa như sau:

$$CosineSim(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|} \tag{6}$$

Trong đó:

- $\vec{A} \cdot \vec{B}$ là tích vô hướng (dot product) giữa hai vector,
- $\|\vec{A}\|$ và $\|\vec{B}\|$ là chuẩn (norm) của từng vector, tính theo công thức $\|\vec{X}\| = \sqrt{\sum_i x_i^2}$.

Giá trị của cosine similarity nằm trong khoảng $[-1, 1]$. Tuy nhiên, trong các bài toán xử lý văn bản, nơi vector biểu diễn thường có giá trị không âm, độ đo này thường dao động trong khoảng $[0, 1]$:

- Giá trị gần 1: hai vector gần như cùng hướng, mức độ tương đồng cao.
- Giá trị gần 0: hai vector gần như vuông góc, ít tương đồng.
- Giá trị âm (hiếm gặp trong NLP): thể hiện hai vector có hướng ngược nhau.

3 Ý tưởng hình thành

3.1 Thách thức

Phim ảnh là một trong những loại hình giải trí phổ biến và có sức lan tỏa mạnh mẽ nhất trong đời sống hiện đại. Với sự phát triển của các nền tảng trực tuyến như Netflix, YouTube, Disney+, hay các trang xem phim miễn phí, người dùng ngày càng có nhiều lựa chọn hơn để tiếp cận các tác phẩm điện ảnh đến từ nhiều quốc gia, thể loại và thời kỳ khác nhau. Tuy nhiên, chính sự phong phú này cũng khiến việc tìm kiếm một bộ phim phù hợp trở nên ngày càng khó khăn, đặc biệt là trong những tình huống mà người dùng không nhớ rõ thông tin cụ thể về bộ phim.

Trong thực tế, nhu cầu tìm kiếm và khám phá phim thông qua tên phim, diễn viên hoặc thể loại là khá phổ biến và đã được nhiều hệ thống hỗ trợ tốt. Tuy nhiên, một tình huống rất thường gặp là người dùng chỉ nhớ mang máng nội dung hoặc diễn biến của bộ phim - ví dụ như một cảnh hành động đặc biệt, tình tiết bất ngờ, hoặc mối quan hệ giữa các nhân vật - mà hoàn toàn không nhớ tên phim hay thông tin cụ thể nào khác. Trong những trường hợp như vậy, việc sử dụng các phương pháp tìm kiếm truyền thống sẽ không còn hiệu quả, và người dùng thường gặp khó khăn trong việc diễn đạt ký ức của mình sao cho khớp với dữ liệu hệ thống có sẵn.

Thách thức đặt ra là làm thế nào để xây dựng một hệ thống truy xuất thông tin có khả năng tìm kiếm phim dựa trên mô tả ngôn ngữ tự nhiên, tức là cho phép người dùng nhập

vào một đoạn văn mô tả nội dung phim mà họ nhớ, và từ đó hệ thống sẽ đề xuất những phim có khả năng trùng khớp cao nhất. Điều này đòi hỏi hệ thống phải hiểu được ngữ nghĩa của văn bản đầu vào, xử lý được sự mơ hồ trong cách diễn đạt của người dùng, và ánh xạ chính xác đến các bộ phim tương ứng trong cơ sở dữ liệu.

Hơn nữa, một điểm cần đạt được là hệ thống cần được thiết kế sao cho gọn nhẹ, có thể triển khai trong môi trường hạn chế về tài nguyên tính toán - ví dụ như máy tính cá nhân hoặc server miễn phí - thay vì dựa vào các mô hình ngôn ngữ lớn (LLM) tiêu tốn nhiều chi phí hoặc các thuật toán học sâu phức tạp. Do đó, bài toán này không chỉ là một bài toán truy xuất thông tin thuần túy, mà còn là bài toán cân bằng giữa tính hiệu quả, chi phí và trải nghiệm người dùng trong điều kiện thực tế.

3.2 Giải pháp

Giải quyết bài toán tìm kiếm phim dựa trên mô tả ngôn ngữ tự nhiên không chỉ giúp cải thiện rõ rệt trải nghiệm người dùng mà còn mở ra cơ hội khai thác sâu hơn khả năng hiểu ngôn ngữ trong các ứng dụng thực tế. Đây là bước tiến cần thiết để hướng tới các hệ thống thông minh có khả năng giao tiếp tự nhiên hơn với con người. Đặc biệt, việc nghiên cứu các giải pháp đơn giản, hiệu quả, không phụ thuộc vào mô hình ngôn ngữ quy mô lớn còn mang ý nghĩa thực tiễn lớn, giúp hệ thống có thể dễ dàng triển khai trong các điều kiện hạn chế về tài nguyên hoặc phục vụ cho cộng đồng rộng lớn hơn với chi phí thấp. Vì vậy, nhóm tác giả đề xuất hệ thống truy xuất thông tin sử dụng thuật toán Latent Senmatic Analysis (LSA)

3.2.1 Vì sao là LSA?

Trước khi LSA được dùng rộng rãi [2], đã có nhiều kỹ thuật biểu diễn từ xuất hiện nhưng tồn tại một số điểm yếu cố hữu. Đầu tiên, các kỹ thuật biểu diễn ma trận từ như ma trận document-term [3] hoặc ma trận đồng xuất hiện [3] tạo ra một ma trận rất thưa do chỉ dựa trên tần suất xuất hiện. Điều này gây khó khăn cho các thuật toán học máy do số chiều quá lớn, dẫn đến hiện tượng curse of dimensionality (lời nguyền cao chiều) [4] và khiến việc học các mối quan hệ ngữ nghĩa trở nên kém hiệu quả. Bên cạnh đó, việc lưu trữ các ma trận thưa như vậy cũng tiêu tốn nhiều bộ nhớ

Tiếp theo, các phương pháp biểu diễn truyền thống không phản ánh chính xác ngữ nghĩa và mối quan hệ giữa các từ. Chúng không nhận diện được các từ đồng nghĩa như *car* và *automobile*, đồng thời không xử lý được hiện tượng từ đa nghĩa như *back*, dẫn đến khó khăn trong việc đo lường độ tương đồng giữa từ hoặc văn bản. Một hạn chế rõ rệt của biểu diễn dựa trên tần suất là các từ phổ biến như *the*, *is*, *and*, *of* thường có trọng số rất cao, dù đóng góp rất ít về mặt ngữ nghĩa. Điều này gây nhiễu cho vector đặc trưng, làm giảm hiệu quả trong việc đo độ tương đồng, phân loại văn bản hoặc trích xuất thông tin.

Cuối cùng, việc phát hiện các chủ đề tiềm ẩn trong một tập văn bản lớn là một thách thức đáng kể. Các phương pháp thời kỳ đầu thường dựa vào các luật thủ công, từ điển chủ đề, hoặc các quy tắc heuristic đơn giản. Tuy nhiên, cách tiếp cận này thiếu khả năng tổng quát hóa, khó áp dụng cho các tập dữ liệu lớn và đa dạng về nội dung.

Để khắc phục những hạn chế đó, LSA đã được đề xuất như một phương pháp có nền tảng toán học rõ ràng nhằm trích xuất và biểu diễn ý nghĩa ngữ cảnh của từ dựa trên các phép tính thống kê áp dụng trên tập văn bản lớn. Ý tưởng chính của LSA là việc một từ xuất hiện hay không xuất hiện trong các ngữ cảnh khác nhau sẽ tạo nên những ràng

buộc tương hỗ, từ đó phản ánh mức độ tương đồng về ngữ nghĩa giữa các từ và cụm từ. Nhiều nghiên cứu đã chứng minh tính hiệu quả của LSA trong việc mô phỏng tri thức ngôn ngữ của con người: từ việc đạt điểm số tương đương với con người trong các bài kiểm tra từ vựng và kiến thức chuyên môn, cho đến khả năng mô phỏng cách con người phân loại từ, đánh giá mức độ liên quan giữa các từ hoặc đoạn văn. Ngoài ra, LSA còn có thể ước lượng độ mạch lạc của đoạn văn, khả năng tiếp thu nội dung của người học, cũng như đánh giá chất lượng tri thức trong văn bản.

3.2.2 Hướng tiếp cận thay thế

LSA tuy có thể đáp ứng được các thách thức Mục 3.1, nhưng nó vẫn là một phương pháp đã lâu đời và thiếu hiệu quả trong các hệ thống hiện đại khi mà vấn đề nguồn tài nguyên được cải thiện. Sự phát triển mạnh mẽ của phần cứng hiện đại, đặc biệt là GPU, các mô hình học sâu trở nên khả thi và phổ biến hơn, cho phép xây dựng những biểu diễn ngôn ngữ giàu ngữ nghĩa hơn, học được trực tiếp từ dữ liệu mà không cần giả định tuyến tính hay ma trận thưa.

Vì vậy, để có cái nhìn toàn diện và đánh giá khách quan hơn giữa các thế hệ phương pháp, nhóm tác giả xin giới thiệu thêm một hướng tiếp cận hiện đại - đó là Word Embedding. Phương pháp này không chỉ cải thiện hiệu quả biểu diễn ngữ nghĩa của từ vựng mà còn cho phép mô hình học được mối quan hệ phức tạp giữa các từ trong ngữ cảnh, từ đó nâng cao độ chính xác trong các bài toán truy xuất thông tin phim. Việc đưa Word Embedding vào so sánh với LSA sẽ giúp làm rõ ưu điểm và hạn chế của từng phương pháp trong từng điều kiện cụ thể, từ đó đưa ra lựa chọn phù hợp hơn cho các hệ thống tìm kiếm thực tế.

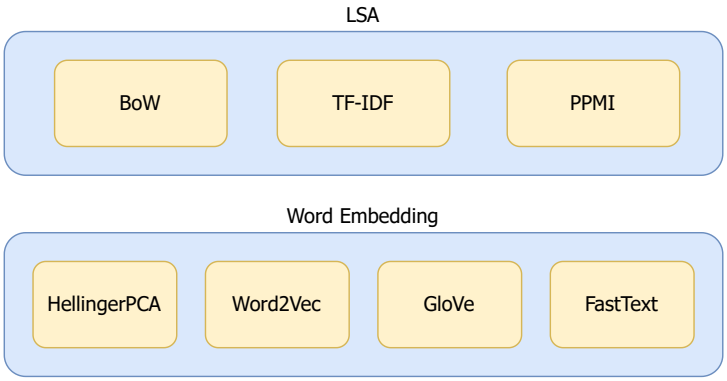
4 Phương pháp

Truy xuất thông tin là một bài toán lâu đời trong lĩnh vực xử lý ngôn ngữ tự nhiên, đóng vai trò quan trọng trong nhiều ứng dụng thực tế nhằm nâng cao trải nghiệm người dùng. Mục tiêu của hệ thống IR là trả về những kết quả có nội dung phù hợp nhất với nhu cầu tìm kiếm của người dùng.

Một hệ thống IR điển hình bao gồm các thành phần chính như: tập tài liệu (corpus), truy vấn (query), chỉ mục (index), và mô hình truy xuất (retrieval model). Quy trình hoạt động cơ bản bao gồm: xử lý câu truy vấn, ánh xạ truy vấn và tài liệu sang không gian nhúng, tính toán độ tương đồng (thường là Cosine), và xếp hạng kết quả. Các chỉ số đánh giá phổ biến của hệ IR bao gồm Precision, Recall, F1-score, MAP, MRR và nDCG, giúp đảm bảo kết quả truy xuất vừa chính xác vừa có thứ tự ưu tiên phù hợp.

Trong báo cáo này, nhóm tác giả triển khai và so sánh hai hướng tiếp cận để xây dựng mô hình truy xuất, được minh họa trong Hình 2:

- LSA: là hướng tiếp cận chính, được trình bày chi tiết trong Mục 4.1, bao gồm các phương pháp truyền thống như BoW, TF-IDF và PPMI, nhằm khai thác mối quan hệ ngữ nghĩa tiềm ẩn giữa các từ và tài liệu thông qua ma trận xuất hiện và giảm chiều.
- Word Embedding: là hướng tiếp cận có tiềm năng thay thế LSA, được trình bày trong Mục 4.2, gồm các kỹ thuật hiện đại hơn như HellingerPCA, Word2Vec, GloVe và FastText, tận dụng ngữ cảnh phân phối của từ trong tập liệu lớn để tạo ra biểu diễn vector giàu ngữ nghĩa.



Hình 2: Tổng quan phương pháp truy xuất thông tin

4.1 Latent Semantic Analysis

Nguyên lý của LSA là ánh xạ ma trận biểu diễn từ (document-term hoặc term-document) sang một không gian mới giàu ngữ nghĩa tiềm ẩn nhờ vào phép xoay trục. Cốt lõi của phương pháp là áp dụng Truncated Singular Value Decomposition, nhằm tách ma trận ban đầu thành ba thành phần chính phản ánh cấu trúc tiềm ẩn của ngôn ngữ.

Cho $X \in \mathbb{R}^{n \times N}$ là ma trận term-document, áp dụng Truncated SVD, ta có (7):

$$X \approx U_k \Sigma_K V_k^\top \tag{7}$$

- $U_k \in \mathbb{R}^{n \times k}$: ma trận biểu diễn các từ trong không gian $\mathcal{S}$, đã được lược bỏ các thành phần không quan trọng
- $V_k \in \mathbb{R}^{k \times N}$: ma trận biểu diễn các tài liệu trong không gian $\mathcal{S}$, đã được lược bỏ các thành phần không quan trọng
- $\Sigma_k \in \mathbb{R}^{k \times k}$: ma trận đường chéo chứa các giá trị suy biến, biểu diễn mức độ quan trọng của các thành phần trong không gian $\mathcal{S}$, và đã được lược bỏ các thành phần không quan trọng.

Sau khi tách thành ba ma trận, nếu muốn biểu diễn tài liệu hoặc từ bất kỳ dựa theo không gian ngữ nghĩa, thì chỉ cần thực hiện phép nhân ma trận.Có hai cách tiếp cận việc xoay trục này: xoay theo không gian từ vựng và xoay theo không gian văn bản.

Xoay theo không gian từ vựng tức là ánh xạ các từ sang một không gian ngữ nghĩa có chiều thấp, trong đó các từ có ngữ nghĩa tương tự sẽ có tọa độ gần nhau. Điều này giúp làm nổi bật mối liên hệ ngữ nghĩa giữa các từ, ngay cả khi chúng không cùng xuất hiện trong một ngữ cảnh cụ thể. Để ánh xạ b từ ngữ bất kỳ vào không gian ngữ nghĩa tiềm ẩn, ta biểu diễn chúng qua ma trận $\mathbf{t} \in \mathbb{R}^{b \times N}$. Khi đó, biểu diễn tương ứng là $\overline{\mathbf{t}}$ được xác định theo công thức:

$$\overline{\mathbf{t}} = \mathbf{t} V_k \tag{8}$$

Nếu ánh xạ văn bản gốc vào tập không gian từ, công thức (8) trở thành:

$$\overline{X} = X V_k = U_k \Sigma_k \tag{9}$$

Khi này, mỗi hàng của $\overline{X} \in \mathbb{R}^{n \times k}$ hoặc $\overline{\mathbf{t}} \in \mathbb{R}^{b \times k}$ tương ứng với mỗi biểu diễn từ trong không gian ngữ nghĩa. Với cách xoay theo không gian văn bản, các văn bản được biểu

diễn dưới dạng vector trong không gian ngữ nghĩa mới. Việc thay đổi về thấp chiều này giúp phát hiện ra các chủ đề tiềm ẩn và mô hình hóa nội dung văn bản hiệu quả hơn, nhất là khi so sánh mức độ tương đồng giữa các tài liệu. Để ánh xạ a tài liệu bất kỳ vào không gian $\mathcal{S}$, trước tiên cần biểu diễn chúng dưới dạng ma trận term-document $\mathbf{d} \in \mathbb{R}^{n \times a}$. Sau đó, dùng công thức:

$$\tilde{\mathbf{d}} = U_k^\top \mathbf{d} \tag{10}$$

Nếu ánh xạ văn bản gốc vào tập không gian tài liệu, công thức (10) trở thành:

$$\tilde{X} = U_k^\top X = \Sigma_k V_k^\top \tag{11}$$

Khi này, mỗi cột của $\tilde{X} \in \mathbb{R}^{k \times N}$ hoặc $\tilde{\mathbf{d}} \in \mathbb{R}^{k \times a}$ tương ứng với mỗi biểu diễn tài liệu trong không gian ngữ nghĩa. Với trường hợp ma trận biểu diễn là document-term, $\mathbf{d}$ và $\mathbf{t}$ sẽ hoán đổi vài trò cho nhau, cụ thể:

$$\tilde{\mathbf{t}} = U_k^\top \mathbf{t}$$

$$\overline{\mathbf{d}} = \mathbf{d} V_k$$

Khi này, mỗi hàng $\overline{\mathbf{d}}$ chính là một biểu diễn tài liệu trong không gian tiềm ẩn, mỗi cột $\tilde{\mathbf{t}}$ chính là một biểu diễn từ ngữ trong không gian tiềm ẩn

Nếu trong trường hợp áp dụng Truncated SVD lên ma trận đồng xuất hiện thì sao? Do tính chất đối xứng của ma trận đồng xuất hiện, ma trận $X \in \mathbb{R}^{n \times n}$ được biểu diễn: $X \approx U_k \Sigma_k U_k^\top$. Khi này, biểu diễn của một từ $\mathbf{t}$ trong không gian ngữ nghĩa tiềm ẩn:

$$\overline{\mathbf{t}} = \mathbf{t} U_k = (U_k^\top \mathbf{t}^\top)^\top = \tilde{\mathbf{t}}^\top \tag{12}$$

Có thể thấy rằng việc biểu diễn một từ khi sử dụng ma trận đồng xuất hiện rất đơn giản, nhưng nếu muốn biểu diễn một tài liệu thì lại khó khăn vì ma trận đồng xuất hiện chỉ là biểu diễn của từ. Để giải quyết, kĩ thuật pooling ra đời bằng cách tổng hợp các vector từ ngữ thành một vector tài liệu. Một số kỹ thuật pooling phổ biến bao gồm:

- Mean pooling: Tính trung bình các vector từ trong tài liệu.
- Sum pooling: Cộng tất cả các vector từ lại với nhau.
- Max pooling: Lấy giá trị lớn nhất theo từng chiều.

Với ý tưởng của kĩ thuật pooling, ta hoàn toàn có thể biểu diễn tất cả các cấp bậc khác của văn bản chỉ nhờ vào đơn vị cơ bản là từ ngữ. Sau khi hiểu được cách hoạt động của LSA, câu hỏi đặt ra là: làm sao xây dựng được ma trận biểu diễn từ phù hợp để tiến hành phân tích ngữ nghĩa tiềm ẩn? Như đã giới thiệu ở Mục 3.2, một số dạng ma trận được sử dụng phổ biến là ma trận document-term, term-document hoặc co-occurrence matrix, tùy thuộc vào mục tiêu biểu diễn và ngữ cảnh ứng dụng. Nhóm tác giả xin giới thiệu ba kỹ thuật phổ biến để xây dựng nên ma trận này, bao gồm: Bag-of-Words (BoW), Term Frequency - Inverse Document Frequence (TF-IDF), và Positive Pointwise Mutual Information (PPMI.

4.1.1 BoW

Xuất phát từ nhu cầu mã hóa văn bản để hỗ trợ xử lý bằng các công cụ toán học và thuật toán, các biểu diễn như ma trận đồng xuất hiện và ma trận tài liệu-từ đã được

phát triển. Trong bối cảnh này, mô hình BoW [3] ra đời, dựa trên nền tảng của ma trận đồng xuất hiện, với nguyên lý cốt lõi là biểu diễn tài liệu thông qua việc thống kê tần suất xuất hiện của các từ trong một tập từ vựng chuẩn hóa. Trong mô hình này, mỗi từ được xem như một đặc trưng độc lập, và toàn bộ tài liệu được ánh xạ thành một vector tần suất dựa trên tập từ vựng. Mặc dù không xét đến ý nghĩa ngữ nghĩa của từ hay ngữ cảnh sử dụng, mô hình BoW vẫn cung cấp một biểu diễn đủ mạnh mẽ, cho phép các thuật toán học máy hoạt động hiệu quả trong nhiều tác vụ xử lý ngôn ngữ tự nhiên.

Ví dụ. Xét hai văn bản đơn giản sau:

```
corpus = ["John likes to watch movies. Mary likes movies too.",
 "Mary also likes to watch football games.]"
```

Từ hai văn bản trên, ta xây dựng danh sách các từ theo thứ tự xuất hiện:

- Document 1: `"John"`, `"likes"`, `"to"`, `"watch"`, `"movies"`, `"Mary"`, `"likes"`, `"movies"`, `"too"`
- Document 2: `"Mary"`, `"also"`, `"likes"`, `"to"`, `"watch"`, `"football"`, `"games"`

Sau đó, biểu diễn mỗi văn bản dưới dạng Bag-of-Words bằng đối tượng JSON như sau:

```
BoW1 = {  "John":1, "likes":2, "to":1, "watch":1,
          "movies":2, "Mary":1, "too":1 }
BoW2 = {  "Mary":1, "also":1, "likes":1, "to":1,
          "watch":1, "football":1, "games":1 }
```

Mỗi cặp `key:value` thể hiện một từ và số lần từ đó xuất hiện trong văn bản, và thứ tự các phần tử không quan trọng.

Lưu ý, nếu một văn bản mới được ghép từ hai văn bản trên, thì biểu diễn Bag-of-Words của nó sẽ là:

```
BoW3 = {  "John":1, "likes":3, "to":2, "watch":2,
          "movies":2, "Mary":2, "too":1, "also":1,
          "football":1, "games":1 }
```

Điều này cho thấy rằng:

$$\text{BoW3} = \text{BoW1} \uplus \text{BoW2}$$

Trong đó ký hiệu $\uplus$ biểu diễn phép hợp rời, tức là cộng các số lần xuất hiện (multiplicities) của từng từ từ hai BoW ban đầu.

Quy trình xây dựng BoW:

1. Làm sạch văn bản (lowercase, loại bỏ ký tự không cần thiết, v.v.).
2. Tokenize văn bản thành từ đơn (theo whitespace hoặc hàm tùy chỉnh).
3. Xây dựng từ vựng:
   - Loại bỏ các từ có tần suất thấp hơn một ngưỡng `min_freq`.
   - Giới hạn số đặc trưng tối đa bằng `max_features`.
4. Mã hóa từng văn bản thành vector BoW bằng cách đếm số lần các từ trong từ vựng xuất hiện.

Ví dụ được xử lí theo quy trình:

Bước 1: Làm sạch văn bản: Toàn bộ văn bản đã được viết thường và loại bỏ dấu chấm cuối câu.

```
corpus = [
 "john likes to watch movies mary likes movies too",
 "mary also likes to watch football games"
]
```

Bước 2: Tokenization theo khoảng trắng (whitespace).

```
tokens1 = ["john", "likes", "to", "watch", "movies",
 "mary", "likes", "movies", "too"]

tokens2 = ["mary", "also", "likes", "to", "watch",
 "football", "games"]
```

Bước 3: Xây dựng từ vựng với điều kiện: `min_freq = 2`, `max_features = 5`. Tính tần suất toàn bộ từ trong corpus:

`"likes"`:	3
`"to"`:	2
`"watch"`:	2
`"movies"`:	2
`"mary"`:	2
`"john", "too", "also", "football", "games"`:	< 2 (loại bỏ)

Sau đó, lấy top 5 từ theo tần suất để tạo ra tập từ vựng:

```
vocab = ["likes", "to", "watch", "movies", "mary"]
```

Bước 4: Mã hóa BoW vector theo từ vựng đã chọn

```
BoW1 = {  "likes":2, "to":1, "watch":1, "movies":2, "mary":1 };
BoW2 = {  "likes":1, "to":1, "watch":1, "mary":1 };
BoW3 = {  "likes":3, "to":2, "watch":2, "movies":2, "mary":2 };
```

4.1.2 TF-IDF

Một trong những phương pháp cải tiến dựa trên BoW là TF-IDF. Phương pháp này được thiết kế để đánh giá mức độ quan trọng của một từ trong một tài liệu cụ thể so với toàn bộ tập tài liệu. Về nguyên tắc, TF-IDF cho rằng một từ càng xuất hiện nhiều lần trong một tài liệu thì càng quan trọng với nội dung của tài liệu đó. Tuy nhiên, nếu từ đó lại cũng xuất hiện phổ biến trong hầu hết các tài liệu, thì tầm quan trọng của nó nên bị giảm đi. Nhờ đó, TF-IDF làm nổi bật các từ mang tính đặc trưng riêng của từng tài liệu, đồng thời làm suy giảm vai trò của các từ phổ biến và ít mang tính phân biệt.

Để tạo nên ma trận TF-IDF, trước hết cần tạo một ma trận term-document hoặc document-term, sau đó áp dụng các biến đổi (1), (2), (3) để tạo ra ma trận. Ngoài ra, có một biến thể của ma trận TF-IDF đó là ma trận IDF.

Ví dụ. Cho tập tài liệu gồm các tài liệu:

```
D = {"Dog, cat, and hamster are pets","Dog is chasing cat.",
     "Cat is hiding dog.","Hamster is sleeping.",
     "Dog protects house.","Pets are cute."}
```

Sau khi tiền xử lý dữ liệu gồm khử các stopwords, dấu câu, và đưa về chữ thường, ta được tập tài liệu mới:

```
D' = {"dog cat hamster pets","dog chasing cat",
      "cat hiding dog","hamster sleeping",
      "dog protects house","pets cute"}
```

Như vậy, ta có $n = 10$ từ duy nhất và $N = 6$ tài liệu, tương ứng với 10 dòng và 6 cột trong ma trận term-document X. Thực hiện tính toán, ta được các bảng:

Bảng 1: Bảng Tần số từ

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**df**
dog	1	1	1	0	1	0	4
cat	1	1	1	0	0	0	3
hamster	1	0	0	1	0	0	2
pets	1	0	0	0	0	1	2
chasing	0	1	0	0	0	0	1
hiding	0	0	1	0	0	0	1
sleeping	0	0	0	1	0	0	1
protects	0	0	0	0	1	0	1
house	0	0	0	0	1	0	1
cute	0	0	0	0	0	1	1
Tổng số từ	4	3	3	2	3	2	

Bảng 2: TF và IDF

	D1 (TF)	**D2**	**D3**	**D4**	**D5**	**D6**	**IDF**
dog	1/4	1/3	1/3	0	1/3	0	log(6/4)
cat	1/4	1/3	1/3	0	0	0	log(6/3)
hamster	1/4	0	0	1/2	0	0	log(6/2)
pets	1/4	0	0	0	0	1/2	log(6/2)
chasing	0	1/3	0	0	0	0	log(6/1)
hiding	0	0	1/3	0	0	0	log(6/1)
sleeping	0	0	0	1/2	0	0	log(6/1)
protects	0	0	0	0	1/3	0	log(6/1)
house	0	0	0	0	1/3	0	log(6/1)
cute	0	0	0	0	0	1/2	log(6/1)

Bảng 3: TF-IDF

	D1	D2	D3	D4	D5	D6
dog	0.1014	0.1352	0.1352	0	0.1352	0
cat	0.1733	0.2310	0.2310	0	0	0
hamster	0.2747	0	0	0.5493	0	0
pets	0.2747	0	0	0	0	0.5493
chasing	0	0.5973	0	0	0	0
hiding	0	0	0.5973	0	0	0
sleeping	0	0	0	0.8959	0	0
protects	0	0	0	0	0.5973	0
house	0	0	0	0	0.5973	0
cute	0	0	0	0	0	0.8959

Bảng 3 chính là ma trận biểu diễn từ của tập tài liệu.

4.1.3 PPMI

PPMI [5] có thể được coi là bản nâng cấp của BoW. BoW có cách làm triển khai dễ hiểu, tuy nhiên các giá trị thô này thường không phản ánh chính xác mối quan hệ ngữ nghĩa - hai từ có thể xuất hiện nhiều lần mà không thực sự có liên hệ chặt chẽ, và ngược lại, những từ có quan hệ ngữ nghĩa sâu sắc đôi khi lại xuất hiện cùng nhau rất hiếm.

Để khắc phục hạn chế đó, chỉ số PPMI được đưa vào nhằm định lượng mức độ bất ngờ trong việc hai từ cùng xuất hiện, so với kỳ vọng nếu chúng độc lập về mặt thống kê. Áp dụng công thức (4) lên ma trận đồng xuất hiện, ta thu được được ma trận PPMI là một ma trận ngữ nghĩa, trong đó mỗi từ được ánh xạ thành một vector thể hiện mức độ liên kết của nó với toàn bộ các từ còn lại trong từ vựng. Khi này ma trận PPMI có thể thay thế cho ma trận TF-IDF trong việc thực hiện nhiệm vụ phân tích ngữ nghĩa tiềm ẩn.

Ví dụ. Xét tập văn bản sau:

```
corpus = [
 "cheese, bread, butter, and milk are common dairy products."
 "A goat eats cheese and drinks milk."
 "bread with cheese and butter is delicious."
 "The goat likes cheese, milk, and a bit of butter."
 "bread and cheese often go with milk."
]
```

Sau khi tiền xử lý gồm:

- Chuyển về chữ thường, loại bỏ dấu câu.
- Loại bỏ các stopwords: `and`, `is`, `are`, `a`, `the`, `with`, `of`, `often`, ...
- Token hóa theo khoảng trắng.
- Loại bỏ các từ chỉ xuất hiện một lần.

Ta thu được tập token sau:

```
doc1 =  ["cheese", "bread", "butter", "milk"]
doc2 =  ["goat", "cheese", "milk"]
doc3 =  ["bread", "cheese", "butter"]
doc4 =  ["goat", "cheese", "milk", "butter"]
doc5 =  ["bread", "cheese", "milk"]
```

Từ đó, ta xây dựng từ vựng gồm 5 từ xuất hiện nhiều hơn 1 lần:

```
vocab = ["cheese", "bread", "butter", "milk", "goat"]
```

Bước 1: Xây dựng ma trận đồng xuất hiện (window size = 2)

Ta xét các cặp từ xuất hiện trong cùng cửa sổ trượt kích thước ± 2. Ví dụ trên `doc1 = ["cheese", "bread", "butter", "milk"]`:

- Với từ `"cheese"` (vị trí 0), trong cửa sổ `[cheese, bread, butter]`, ta lấy các cặp:
  - (`cheese`, `bread`), (`cheese`, `butter`)
- Với từ `"bread"` (vị trí 1), cửa sổ `[cheese, bread, butter, milk]`, tạo cặp:
  - (`bread`, `cheese`), (`bread`, `butter`), (`bread`, `milk`)
- Với từ `"butter"` (vị trí 2), cửa sổ `[cheese, bread, butter, milk]`, tạo cặp:
  - (`butter`, `cheese`), (`butter`, `bread`), (`butter`, `milk`)
- Với từ `"milk"` (vị trí 3), cửa sổ `[bread, butter, milk]`, tạo cặp:
  - (`milk`, `bread`), (`milk`, `butter`)

Tổng hợp các cặp từ xuất hiện từ `doc1` (không phân biệt thứ tự trái phải) gồm:

```
(cheese, bread), (cheese, butter), (bread, butter), (bread, milk),
                          (butter, milk)
```

Những cặp này sẽ được đếm vào ma trận đồng xuất hiện C. Ta thực hiện tương tự trên những câu còn lại. Kết quả ma trận đồng xuất hiện C như sau:

Bảng 4: Ma trận đồng xuất hiện C

	`bread`	`butter`	`cheese`	`goat`	`milk`
`bread`	0	2	3	0	2
`butter`	2	0	3	0	2
`cheese`	3	3	0	2	3
`goat`	0	0	2	0	2
`milk`	2	2	3	2	0

Tổng số lần đồng xuất hiện là $N = 38$.

Bước 2: Tính xác suất biên và xác suất đồng xuất hiện

Tính xác suất biên:

$$P(\texttt{bread}) = \frac{7}{38}, \quad P(\texttt{butter}) = \frac{7}{38}, \quad P(\texttt{cheese}) = \frac{11}{38}, \quad P(\texttt{goat}) = \frac{4}{38}, \quad P(\texttt{milk}) = \frac{9}{38}$$

Tìm xác suất đồng xuất hiện:

$$\begin{aligned} P(\texttt{bread}, \texttt{butter}) &= \frac{2}{38}, \quad P(\texttt{bread}, \texttt{cheese}) = \frac{3}{38}, \\ P(\texttt{bread}, \texttt{milk}) &= \frac{2}{38}, \quad P(\texttt{butter}, \texttt{cheese}) = \frac{3}{38}, \\ P(\texttt{butter}, \texttt{milk}) &= \frac{2}{38}, \quad P(\texttt{cheese}, \texttt{milk}) = \frac{3}{38}, \\ P(\texttt{cheese}, \texttt{goat}) &= \frac{2}{38}, \quad P(\texttt{goat}, \texttt{milk}) = \frac{2}{38} \end{aligned}$$

Bước 3: Tính PMI và PPMI

PMI và PPMI được tính theo công thức:

$$PMI(w_i, w_j) = \log_2 \left(\frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)} \right), \quad PPMI(w_i, w_j) = \max(0, PMI(w_i, w_j))$$

Ví dụ 1: Tính PMI và PPMI giữa `bread` và `cheese`

$$\begin{aligned} PMI(\texttt{bread}, \texttt{cheese}) &= \log_2 \left(\frac{P(\texttt{bread}, \texttt{cheese})}{P(\texttt{bread}) \cdot P(\texttt{cheese})} \right) \\ &= \log_2 \left(\frac{3/38}{(7/38) \cdot (11/38)} \right) \\ &= \log_2 \left(\frac{3 \cdot 38}{7 \cdot 11} \right) \\ &= \log_2(1.4805) \approx 0.57 \end{aligned}$$

$$\begin{aligned} PPMI(\texttt{cheese}, \texttt{bread}) &= \max(0, PMI(\texttt{cheese}, \texttt{bread})) \\ &= \max(0, 0.57) = 0.57 \end{aligned}$$

Ví dụ 2: Tính PMI và PPMI giữa `bread` và `goat`

$$\begin{aligned} PMI(\texttt{bread}, \texttt{goat}) &= \log_2 \left(\frac{0}{(7/38) \cdot (4/38)} \right) \\ &= \log_2(0) = -\infty \end{aligned}$$

$$\begin{aligned} PPMI(\texttt{bread}, \texttt{goat}) &= \max(0, PMI(\texttt{bread}, \texttt{goat})) \\ &= \max(0, -\infty) = 0 \end{aligned}$$

Bước 4: Ma trận PPMI

Sau khi tính toán tất cả các giá trị PPMI, ta thu được bảng sau:

Bảng 5: Ma trận PPMI

	bread	butter	cheese	goat	milk
bread	0	0.63	0.57	0	0.27
butter	0.63	0	0.57	0	0.27
cheese	0.57	0.57	0	0.79	0.20
goat	0	0	0.79	0	1.08
milk	0.27	0.27	0.20	1.08	0

Mỗi giá trị trong bảng trên là $PPMI(w_i, w_j)$, làm tròn đến 2 chữ số thập phân. Các phần tử có PMI âm được đưa về 0 theo định nghĩa PPMI.

Diễn giải:

- Giá trị $PPMI(\texttt{goat}, \texttt{milk}) = 1.08$ là lớn nhất, cho thấy `goat` và `milk` có mối quan hệ ngữ nghĩa mạnh trong văn bản (ví dụ: dê thường đi kèm với sữa).
- Các cặp như (`bread`, `cheese`) và (`butter`, `cheese`) có $PPMI = 0.57$, phản ánh sự đồng xuất hiện đáng kể giữa các từ này trong ngữ cảnh.
- Những ô có giá trị 0 (như PPMI(`bread`, `goat`)) cho biết hai từ gần như không xuất hiện trong cùng ngữ cảnh (cửa sổ trượt).

4.2 Các mô hình Word Embedding

Word Embedding là các phương pháp giống với LSA, được dùng để biểu diễn các từ dưới dạng vector số thực trong không gian nhiều chiều, sao cho các từ có ngữ nghĩa giống nhau sẽ có vector gần nhau hơn.Tuy nhiên, khác với các phương pháp biểu diễn truyền thống như Bag of Words hay TF-IDF vốn bỏ qua ngữ cảnh, Word Embedding học được mối quan hệ ngữ nghĩa và ngữ cảnh giữa các từ nhờ vào dữ liệu huấn luyện. Nhóm tác giả giới thiệu tới bốn phương pháp, bao gồm: HellingerPCA, Word2Vec, GloVe, FastTest. Ngoài ra, do các mô hình Word Embedding chỉ là một hướng tiếp cận khác, nhóm tác sẽ chỉ giới thiệu mà không đề cập tới ví dụ cụ thể.

4.2.1 HellingerPCA

Mô hình HellingerPCA được xây dựng dựa trên giả thuyết phân phối trong ngôn ngữ học, cụ thể là quan điểm nổi tiếng của Firth (1957): *You shall know a word by the company it keeps*. Câu nói này thể hiện rằng ngữ nghĩa của một từ có thể được suy ra từ các từ ngữ cảnh mà nó thường đi kèm trong văn bản. Đây là nền tảng của rất nhiều mô hình biểu diễn từ hiện đại, trong đó có Word2Vec, GloVe và HellingerPCA.

Hướng tiếp cận. Đầu tiên, ta xây dựng một ma trận đồng xuất hiện $X \in \mathbb{R}^{n \times n}$. Ma trận này ban đầu chỉ chứa các giá trị đếm thô, nên chưa phản ánh bản chất ngữ nghĩa một cách tương đối. Để chuẩn hóa và trích xuất thông tin phân phối, ta chuyển từng hàng của ma trận X thành một phân phối xác suất có điều kiện, tức là biểu diễn xác suất $P(c \mid w)$ - xác suất từ ngữ cảnh c xuất hiện khi biết từ mục tiêu là w:

$$P(c \mid w) = \frac{X_{wc}}{\sum_{c'} X_{wc'}}$$

Kết quả là mỗi từ w giờ đây được biểu diễn như một vector phân phối xác suất trên tập từ ngữ cảnh. Ma trận chuẩn hóa này mô tả trực tiếp mối quan hệ ngữ nghĩa giữa từ và ngữ cảnh, theo nghĩa phân phối.

Tuy nhiên, khoảng cách Euclidean không phải là độ đo lý tưởng để đo lường khoảng cách giữa các phân phối xác suất, bởi vì nó không phản ánh đúng tính chất của không gian simplex (nơi mà tổng các xác suất bằng 1). Trong các nghiên cứu lý thuyết thông tin, khoảng cách Hellinger được coi là một lựa chọn tốt để đo sự khác biệt giữa các phân phối xác suất. Cụ thể, cho hai phân phối xác suất rời rạc $P = (p_1, p_2, \ldots, p_n)$ và $Q = (q_1, q_2, \ldots, q_n)$, khoảng cách Hellinger giữa P và Q được định nghĩa như sau:

$$H(P, Q) = \frac{1}{\sqrt{2}} \left(\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2 \right)^{1/2} \tag{13}$$

Nếu ta đặt $\tilde{P} = \sqrt{P}, \tilde{Q} = \sqrt{Q}$, khi này biểu thức (13) trở thành:

$$H(P, Q) = \frac{1}{\sqrt{2}} \left(\sum_{i=1}^{n} (\tilde{p}_i - \tilde{q}_i)^2 \right)^{1/2} = \frac{1}{\sqrt{2}} ||\tilde{P} - \tilde{Q}||_2 \approx ||\tilde{P} - \tilde{Q}||_2 \tag{14}$$

Biểu thức (14) có nghĩa khoảng cách Euclidean của phân phối được biến đổi xấp xỉ khoảng cách Hellinger của phân phối gốc. Điều này mang ý nghĩa rất lớn vì biến đổi này vừa giữ được tính chất của một số không gian đặc trưng như văn bản, vừa giữ được ý nghĩa toàn cục của khoảng cách Euclidean. Biến đổi $\tilde{P} = \sqrt{P}$ còn được gọi là biến đổi Hellinger.

Như vậy, ta áp dụng biến đổi Hellinger $\tilde{P}(c \mid w) = \sqrt{P(c \mid w)}$ đại diện cho từng từ, sau đó dụng phương PCA với mục tiêu là giảm chiều dữ liệu trong khi vẫn bảo toàn cấu trúc ngữ nghĩa bên trong. Kết quả cuối cùng là mỗi từ được ánh xạ thành một vector trong không gian thấp chiều, nơi khoảng cách giữa các từ phản ánh mức độ tương đồng ngữ nghĩa, dựa trên sự phân phối đồng xuất hiện với ngữ cảnh.

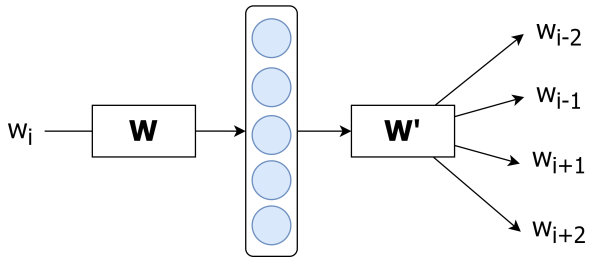
4.2.2 Word2Vec

Word2Vec [6] là một kỹ thuật học biểu diễn từ được phát triển nhằm ánh xạ các từ ngữ trong ngôn ngữ tự nhiên thành các vector số thực trong không gian nhiều chiều, sao cho các mối quan hệ ngữ nghĩa và cú pháp giữa các từ được bảo tồn.

Trước khi tìm hiểu Word2Vec, ta có thể liên hệ tới một kỹ thuật phổ biến khác trong học máy là PCA. PCA được sử dụng để giảm chiều dữ liệu trong khi cố gắng giữ lại càng nhiều thông tin càng tốt. Tuy nhiên, PCA là một kỹ thuật tuyến tính và không tận dụng được các mối quan hệ ngữ nghĩa phi tuyến tính phức tạp giữa các từ. Đây là lý do khiến Word2Vec ra đời: không chỉ để biểu diễn từ dưới dạng vector mà còn để học được mối liên hệ ngữ nghĩa giữa các từ dựa trên ngữ cảnh sử dụng trong văn bản.

Khác với PCA, Word2Vec dùng một mạng nơ-ron đơn giản với một lớp ẩn duy nhất, được huấn luyện để tối ưu hóa xác suất đồng xuất hiện giữa các từ. Kết quả sau huấn luyện là các vector từ nằm trong không gian số thực có thể so sánh được bằng các phép đo như cosine similarity.

Hướng tiếp cận. Word2Vec bao gồm hai kiến trúc chính: CBOW (Continuous Bag-of-Words) và Skip-Gram. Cả hai kiến trúc đều sử dụng một mạng nơ-ron gồm một lớp ẩn duy nhất. Các trọng số của lớp ẩn sau khi huấn luyện chính là vector biểu diễn từ. Do tính chất tương tự nhau, nhưng khả năng học từ hiếm tốt của Skip-Gram khiến phù hợp với bài toán truy xuất thông, nên nhóm tác giả chỉ trình bày đặc trưng của phương pháp Skip-Gram.



Hình 3: Cấu trúc mạng neuron của Skip-gram

Biểu diễn toán học của Skip-Gram: Giả sử từ vựng có kích thước V và ta chọn kích thước embedding là N. Mỗi từ w được biểu diễn bởi một one-hot vector $\mathbf{x} \in \mathbb{R}^V$.

- Ma trận trọng số đầu vào: $W \in \mathbb{R}^{V \times N}$.
- Ma trận trọng số đầu ra: $W' \in \mathbb{R}^{N \times V}$.

Khi một từ w_t được đưa vào mạng, vector one-hot $\mathbf{x}_t$ được nhân với W:

$$\mathbf{h} = W^T \mathbf{x}_t$$

Đầu ra của mạng là một vector xác suất $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \text{softmax}(W'^T \mathbf{h}) = \frac{\exp({\mathbf{v}'_{w_o}}^T \mathbf{v}_{w_t})}{\sum_{w=1}^{V} \exp({\mathbf{v}'_w}^T \mathbf{v}_{w_t})}$$

- $\mathbf{v}_{w_t}$ là vector biểu diễn từ đầu vào (lấy từ W).
- $\mathbf{v}'_{w_o}$ là vector đầu ra của từ dự đoán (lấy từ W').

Hàm mất mát: Hàm mất mát là negative log-likelihood của từ thực sự w_o trong ngữ cảnh từ w_t:

$$\mathcal{L} = -\log \hat{y}_{w_o} = -\log \left(\frac{\exp({\mathbf{v}'_{w_o}}^T \mathbf{v}_{w_t})}{\sum_{w=1}^{V} \exp({\mathbf{v}'_w}^T \mathbf{v}_{w_t})} \right)$$

Vấn đề là việc tính đạo hàm qua softmax rất tốn chi phí với từ vựng lớn vì phải tính tổng qua toàn bộ V từ. Để khắc phục vấn đề chi phí tính toán cao, kỹ thuật negative sampling được sử dụng. Thay vì cập nhật toàn bộ từ vựng, chỉ một số nhỏ các từ âm (không phải từ ngữ cảnh thật) được lấy mẫu và tối ưu hoá theo công thức:

$$\mathcal{L} = -\log \sigma({\mathbf{v}'_{w_o}}^T \mathbf{v}_{w_t}) - \sum_{i=1}^{k} \log \sigma(-{\mathbf{v}'_{w_i^-}}^T \mathbf{v}_{w_t})$$

- $\sigma(x) = \frac{1}{1+\exp(-x)}$ là hàm sigmoid.

- w_o là từ đúng.
- w_i^- là các từ âm (negative samples), được chọn ngẫu nhiên theo phân phối xác suất.
- k là số lượng negative samples (thường từ 5-20).

Ý nghĩa trực giác. Sau khi huấn luyện, các vector từ học được có những đặc tính ngữ nghĩa nổi bật. Những quan hệ tuyến tính như Hình 4 không phải là mục tiêu huấn luyện trực tiếp, mà là kết quả emergent từ việc học các đồng xuất hiện từ.

Hình 4: *king - man + woman ≈ queen*

4.2.3 GloVe

GloVe [7] (Global Vectors for Word Representation) là một phương pháp học biểu diễn từ không giám sát, tận dụng thông tin từ toàn bộ tập dữ liệu ngôn ngữ thông qua ma trận đồng xuất hiện. Khác với các phương pháp như Word2Vec vốn dựa trên ngữ cảnh cục bộ, GloVe khai thác các thống kê toàn cục của văn bản để xây dựng một không gian vector phản ánh mối quan hệ ngữ nghĩa giữa các từ.

Hướng tiếp cận. Giả sử ta có hai từ mục tiêu là *ice* (băng) và *steam* (hơi nước), đều liên quan đến khái niệm pha nhiệt động học. Để hiểu rõ hơn ý nghĩa của các từ này, ta khảo sát xác suất đồng xuất hiện của chúng với các từ ngữ cảnh k khác nhau. Chẳng hạn, từ $k = \text{solid}$ liên quan mạnh đến *ice* nhưng không liên quan đến *steam*, do đó ta kỳ vọng tỷ lệ $P(k \mid \text{ice})/P(k \mid \text{steam})$ sẽ lớn. Ngược lại, với từ $k = \text{gas}$, tỷ lệ đó sẽ nhỏ. Với những từ như $k = \text{water}$ hay $k = \text{fashion}$, không phân biệt rõ giữa hai từ mục tiêu, tỷ lệ này sẽ gần bằng 1. Bảng 6 trình bày các xác suất và tỷ lệ cụ thể, cho thấy rằng tỷ lệ này là một đặc trưng tốt hơn so với bản thân xác suất trong việc tách biệt các từ liên quan và không liên quan.

Probability and Ratio	$k = \text{solid}$	$k = \text{gas}$	$k = \text{water}$	$k = \text{fashion}$
$P(k \mid \text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k \mid \text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k \mid \text{ice}) \,/\, P(k \mid \text{steam})$	8.9	8.5×10^{-2}	1.36	0.96

Bảng 6: Xác suất đồng xuất hiện và tỷ lệ của các từ ngữ cảnh với hai từ mục tiêu *ice* và *steam*

Mô hình toán học. Giả sử ta có ma trận đồng xuất hiện $X \in \mathbb{R}^{V \times V}$, trong đó X_{ij} là số lần từ j xuất hiện trong ngữ cảnh của từ i. Tổng số lần từ j xuất hiện cùng i là $X_i = \sum_k X_{ik}$, và xác suất điều kiện là:

$$P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i} \tag{15}$$

Ý tưởng chính của GloVe là xây dựng các vector từ $\mathbf{w}_i$ và vector ngữ cảnh tương ứng $\tilde{\mathbf{w}}_j$ sao cho các quan hệ ngữ nghĩa được thể hiện thông qua cấu trúc tuyến tính giữa chúng. Ta kỳ vọng rằng:

$$F(\mathbf{w}_i - \mathbf{w}_j, \tilde{\mathbf{w}}_k) = \frac{P_{ik}}{P_{jk}}$$

Vì không gian vector có tính tuyến tính, hàm F hợp lý nhất là tích vô hướng:

$$(\mathbf{w}_i - \mathbf{w}_j)^T \tilde{\mathbf{w}}_k = \log \frac{P_{ik}}{P_{jk}} \Rightarrow \mathbf{w}_i^T \tilde{\mathbf{w}}_k - \mathbf{w}_j^T \tilde{\mathbf{w}}_k = \log P_{ik} - \log P_{jk} \tag{16}$$

Sử dụng phương pháp đồng nhất, ta được:

$$\mathbf{w}_i^T \tilde{\mathbf{w}}_k = \log P_{ik} = \log X_{ik} - \log X_i \tag{17}$$

Tuy nhiên, vì $\log X_i$ không phụ thuộc vào k, ta có thể gộp nó vào một hằng số bù b_i. Thêm một hằng số bù khác $\tilde{b}_k$ cho ngữ cảnh, phương trình (17) trở thành:

$$\mathbf{w}_i^T \tilde{\mathbf{w}}_k + b_i + \tilde{b}_k = \log X_{ik} \tag{18}$$

Xây dựng hàm mất mát: Hàm mất mát của GloVe được thiết lập dưới dạng bình phương có trọng số:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2$$

Trong đó, hàm trọng số $f(x)$ được thiết kế để giảm ảnh hưởng của các đồng xuất hiện hiếm gặp hoặc quá thường xuyên. Một lựa chọn điển hình là:

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}} \right)^{\alpha} & \text{nếu } x < x_{\max} \\ 1 & \text{nếu } x \geq x_{\max} \end{cases}$$

với $\alpha = 3/4$ và $x_{\max} = 100$ là các siêu tham số được chọn theo thực nghiệm.

GloVe cung cấp một cách tiếp cận hiệu quả và trực quan để học vector biểu diễn từ, nhờ vào việc kết hợp đặc điểm thống kê toàn cục và cấu trúc tuyến tính của không gian vector. Mô hình này thường được sử dụng làm bước đầu tiên trong các hệ thống NLP hiện đại, đặc biệt trong các tác vụ như phân tích cảm xúc, phân loại văn bản, và hệ thống hỏi đáp.

4.2.4 FastText

FastText [6] [8] được phát triển bởi nhóm nghiên cứu tại Facebook AI Research, là một kỹ thuật học biểu diễn từ tiên tiến, mở rộng từ mô hình Word2Vec. Khác với Word2Vec, vốn coi mỗi từ như một đơn vị độc lập, FastText phân tích từ thành tập hợp các subword (n-gram ký tự), cho phép mô hình học được thông tin hình thái học (morphological information) và xử lý hiệu quả các từ hiếm, từ ngoài từ vựng (out-of-vocabulary – OOV), cũng như các ngôn ngữ có cấu trúc hình thái học phức tạp như tiếng Việt, tiếng Đức, tiếng Ả Rập, hoặc tiếng Thổ Nhĩ Kỳ.

Hướng tiếp cận. FastText dựa trên giả thuyết phân phối của Firth, thế nhưng FastText mở rộng giả thuyết này bằng cách kết hợp thông tin subword, giúp cải thiện khả năng biểu diễn các từ hiếm và từ OOV. Ví dụ, trong tiếng Việt, từ *người chơi* và *cầu thủ* có thể xuất hiện trong ngữ cảnh tương tự (liên quan đến thể thao), nhưng nhờ n-gram, FastText còn học được rằng *người chơi* và *người học* chia sẻ thành phần *người*, phản ánh mối liên hệ hình thái học.

Ý tưởng cốt lõi của FastText bắt nguồn từ nhận thức rằng ý nghĩa và cấu trúc của từ trong nhiều ngôn ngữ không chỉ nằm ở từ như một đơn vị hoàn chỉnh mà còn ở các thành phần bên trong, chẳng hạn như tiền tố, hậu tố, hoặc gốc từ. Ví dụ, trong tiếng Việt, các từ như *chơi*, *đang chơi*, *người chơi*, *sự chơi đùa* chia sẻ gốc từ *chơi*, và việc học vector riêng cho từng từ có thể bỏ qua mối liên hệ hình thái học giữa chúng. FastText khắc phục điều này bằng cách phân tích từ thành các n-gram ký tự, cho phép mô hình học được cả ngữ nghĩa (dựa trên ngữ cảnh) và hình thái học (dựa trên cấu trúc từ).

Giả sử từ vựng có kích thước V. Mỗi từ w được biểu diễn bằng tập hợp các n-gram ký tự $\{g_1, g_2, \ldots, g_{|G_w|}\}$, trong đó G_w bao gồm tất cả n-gram có độ dài từ $n_{\min}$ đến $n_{\max}$ (thường $n_{\min} = 3$, $n_{\max} = 6$). Mỗi n-gram g_i được ánh xạ thành một vector $\mathbf{z}_{g_i} \in \mathbb{R}^d$, với d là số chiều của không gian biểu diễn (thường từ 100 đến 300). Vector biểu diễn của từ w được tính bằng trung bình cộng của các vector n-gram:

$$\mathbf{v}_w = \frac{1}{|G_w|} \sum_{g \in G_w} \mathbf{z}_g$$

FastText sử dụng kiến trúc Skip-gram, tương tự Word2Vec, nhằm dự đoán các từ ngữ cảnh xung quanh từ trung tâm w_t trong một cửa sổ ngữ cảnh (context window) có kích thước c. Tuy nhiên, nhờ sử dụng n-gram, FastText tạo ra các vector mang tính tổng quát cao hơn, phản ánh cả ngữ nghĩa và hình thái học.

Hàm mất mát được tối ưu hóa bằng negative sampling để giảm chi phí tính toán khi từ

vựng lớn:

$$\mathcal{L} = -\log \sigma\left(\mathbf{v}'^{T}_{w_o}\mathbf{v}_{w_t}\right) - \sum_{i=1}^{k} \log \sigma\left(-\mathbf{v}'^{T}_{w_i^-}\mathbf{v}_{w_t}\right)$$

- $\mathbf{v}_{w_t} \in \mathbb{R}^d$ là vector biểu diễn từ trung tâm, được tính từ các n-gram của w_t.
- $\mathbf{v}'_{w_o} \in \mathbb{R}^d$ là vector biểu diễn từ ngữ cảnh đúng (positive sample).
- $\mathbf{v}'_{w_i^-} \in \mathbb{R}^d$ là vector của các từ âm (negative samples), được chọn ngẫu nhiên theo phân phối xác suất tỷ lệ nghịch với tần suất từ.
- $\sigma(x) = \frac{1}{1+\exp(-x)}$ là hàm sigmoid.
- k là số lượng negative samples, thường từ 5 đến 20, tùy thuộc vào kích thước dữ liệu và tài nguyên tính toán.

Hàm mất mát được tối ưu hóa bằng gradient descent, điều chỉnh các vector $\mathbf{z}_g$ để tăng xác suất dự đoán đúng từ ngữ cảnh và giảm xác suất dự đoán các từ âm. Ngoài Skip-gram, FastText cũng hỗ trợ kiến trúc CBOW (Continuous Bag-of-Words), nhưng Skip-gram thường được sử dụng do hiệu quả cao hơn trong việc học biểu diễn ngữ nghĩa.

Subword n-grams. FastText sử dụng n-gram ký tự với độ dài từ 3 đến 6, có thể điều chỉnh tùy thuộc vào đặc điểm ngôn ngữ. Ví dụ:

- Với từ *where* và $n = 3$, các n-gram là: `<wh, whe, her, ere, re>`, trong đó `<` và `>` biểu thị ranh giới từ.
- Với từ *playing* và $n = 3$, các n-gram là: `<pl, pla, lay, ayi, yin, ing, ng>`.
- Trong tiếng Việt, với từ *người chơi* (nếu coi là một từ ghép), các n-gram có thể bao gồm: `<ng, ngu, guo, uoi, oi , ch, cho, hoi, oi>`, giúp mô hình học được thông tin từ các thành phần như *người* và *chơi*.

Phương pháp này cho phép FastText học được cấu trúc hình thái học, chẳng hạn như tiền tố (*un-*, *đang-*), hậu tố (*-ing*, *-er*), hoặc gốc từ (*play*, *chơi*). Đối với từ OOV, FastText tạo vector bằng cách tổng hợp các vector của các n-gram có trong từ đó. Ví dụ, nếu từ *playful* không có trong từ vựng huấn luyện, FastText vẫn có thể biểu diễn nó dựa trên các n-gram như `pla, lay, ayf, yfu, ful`.

Nhờ cơ chế này, FastText không chỉ cải thiện khả năng biểu diễn từ hiếm mà còn xử lý hiệu quả các từ chưa từng xuất hiện trong tập huấn luyện. Điều này đặc biệt hữu ích trong các ngôn ngữ giàu hình thái như tiếng Việt, nơi từ mới có thể được hình thành linh hoạt từ các yếu tố cấu thành.

5 Thuật toán và Mã minh họa

Như đã đề cập ở Mục 4.2, nhóm tác giả chỉ trình bày phần thuật toán của LSA, bao gồm các phương pháp tạo ma trận biểu diễn từ là BoW, TF-IDF, PPMI.

5.1 Thuật toán

Algorithm 1 LSA: Biểu diễn tài liệu trong không gian ngữ nghĩa tiềm ẩn

procedure LSA($\mathcal{D}, \mathbf{d}$)
    Tạo ma trận biểu diễn từ X từ $\mathcal{D}$
    Phân rã giá trị kỳ dị rút gọn (SVD) trên X để thu được ma trận U_k
    Tạo biểu diễn ban đầu $\hat{\mathbf{d}}$ cho tài liệu $\mathbf{d}$
    Xoay trục: $\overline{\mathbf{d}} \leftarrow U_k \hat{\mathbf{d}}$
    return $\overline{\mathbf{d}}$
end procedure

Algorithm 2 Xây dựng ma trận BoW từ tập tài liệu

procedure BuildBoW(D, `min_freq`, `max_features`)
    Tạo bộ đếm tần suất từ trên toàn tập và khởi tạo ma trận không $X \in \mathbb{R}^{N \times F}$
    Lọc từ có `freq < min_freq`, giữ lại top `max_features`
    Gán chỉ số cho từ vựng $T = \{t_1, ..., t_F\}$
    for $j = 1$ **to** N **do**
        Đếm số lần mỗi từ $w \in T$ xuất hiện trong d_j
        $X_{j,\text{idx}(w)} \leftarrow$ số lần w trong d_j
    end for
    return X
end procedure

Algorithm 3 Xây dựng ma trận TF-IDF từ tập tài liệu

procedure BuildTFIDF(D)
    Tìm tập từ $T = \{t_1, ..., t_n\}$, $N = |D|$
    Tạo ma trận term-document tần suất cơ bản X
    for $i = 1$ **to** n **do**
        Tính $IDF(t_i)$
        for $j = 1$ **to** N **do**
            Tính $TF(t_i, d_j)$
            $X_{i,j} \leftarrow TF(t_i, d_j) \cdot IDF(t_i)$
        end for
    end for
    return X
end procedure

Algorithm 4 Xây dựng ma trận PPMI từ tập tài liệu

procedure BUILDPPMI(D)
    Xây dựng ma trận đồng xuất hiện $\mathbf{C} \in \mathbb{R}^{n \times n}$
    $N \leftarrow \sum_{i,j} \mathbf{C}_{i,j}$
    for $i = 1$ **to** n **do**
        $P(w_i) \leftarrow \sum_j \mathbf{C}_{i,j}/N$
        for $j = 1$ **to** n **do**
            if $\mathbf{C}_{i,j} > 0$ **then**
                $P(w_i, w_j) \leftarrow \mathbf{C}_{i,j}/N$
                $PMI \leftarrow \log_2 \left(\frac{P(w_i,w_j)}{P(w_i) \cdot P(w_j)} \right)$
                $PPMI(w_i, w_j) \leftarrow \max(0, PMI)$
            else
                $PPMI(w_i, w_j) \leftarrow 0$
            end if
        end for
    end for
    return $PPMI$
end procedure

5.2 Mã minh họa

Khác với Mục 5.1 khi mà mỗi thuật toán tạo ma trận từ được làm độc lập rồi mới kết hợp với Truncated SVD giảm chiều, nhóm tác giả đã cài đặt thuật toán tạo ma trận và giảm chiều cùng vào trong một tệp tin Python để việc huấn luyện thuận tiện hơn. Phía dưới là mã minh họa cho BoW, TF-IDF, PPMI để biến đổi từ một tập tài liệu thành dạng biểu diễn LSA. Ở mỗi phần, nhóm tác giả sẽ trình bày các thuộc tính và phương thức của thuật toán, cùng với đó là cách sử dụng

5.2.1 Bag-of-Words

Lớp `BagOfWords`. Lớp này xây dựng biểu diễn Bag-of-Words cho các tài liệu văn bản đã làm sạch. Các phương thức chính gồm:

- `__init__`: Khởi tạo với các tham số như tần suất từ tối thiểu (`min_word_freq`), số lượng đặc trưng tối đa (`max_features`), và bộ tách từ (`tokenizer`).
- `fit(documents)`:
  - Tách từ và đếm số lần xuất hiện của mỗi từ trong tập văn bản.
  - Lọc các từ theo tần suất tối thiểu và giới hạn số lượng đặc trưng.
  - Xây dựng từ điển ánh xạ từ $\rightarrow$ chỉ số.
- `transform(documents)`: Chuyển tập tài liệu thành ma trận BoW có kích thước $(n\_documents, n\_features)$.
- `transform_single(document)`: Tạo vector BoW cho một tài liệu đơn lẻ.
- `fit_transform(documents)`: Gộp hai bước `fit` và `transform`.
- `get_feature_names()`: Trả về danh sách các từ trong từ điển.

- `get_vocabulary_size()`: Trả về kích thước của từ điển.
- `get_word_frequency(word)`: Trả về số lần xuất hiện của một từ.
- `print_vocabulary_info(top_n)`: In thông tin thống kê về từ điển.
- `save_vocabulary(filepath)` và `load_vocabulary(filepath)`: Lưu và tải từ điển từ tệp văn bản.

Lớp `SVDModel`. Lớp thực hiện giảm chiều bằng TruncatedSVD được cài đặt từ đầu bằng phân rã giá trị riêng. Các phương thức bao gồm:

- `__init__`: Khởi tạo với số chiều đầu ra mong muốn (`n_components`).
- `fit(X)`:
  - Tính ma trận hiệp phương sai $X^\top X$ và phân rã thành các trị riêng.
  - Lấy các thành phần chính (`U, S, Vt`) và tỷ lệ phương sai giải thích.
- `transform(X)`: Chiếu dữ liệu lên không gian giảm chiều.
- `fit_transform(X)`: Gộp hai bước `fit` và `transform`.
- `inverse_transform(X_transformed)`: Khôi phục dữ liệu từ không gian giảm chiều.

Lớp `BOW_SVD_Embedding`. Đây là pipeline kết hợp giữa Bag-of-Words và SVD để sinh vector biểu diễn tài liệu. Các phương thức chính gồm:

- `__init__`: Khởi tạo pipeline với tham số tùy chỉnh cho BoW (`bow_args`) và SVD (`dim_reduc_args`).
- `fit(documents)`: Học từ điển và giảm chiều từ tập tài liệu đầu vào.
- `transform(documents)`: Chuyển đổi tài liệu sang vector rút gọn đã học.
- `fit_transform(documents)`: Gộp hai bước `fit` và `transform`.
- `get_feature_names()`: Trả về danh sách từ trong từ điển BoW.
- `get_vocabulary_size()`: Trả về kích thước từ điển BoW.

Cách sử dụng. Khởi tạo tài liệu.

```
# List of corpus
corpus = [
    "John likes to watch movies. Mary likes movies too.",
    "Mary also likes to watch football games."
]
# Preprocessed corpus (This step is for illustration purposes only)
preprocessed_corpus = [re.sub(r'\.', '', sentence.lower()) for sentence in corpus]
print(preprocessed_corpus)
# Result of preprocessed corpus ['john likes to watch movies mary likes movies too', '
    mary also likes to watch football games']
```

Khai báo model BoW đã chuẩn bị.

```
from embedding.BoW import BagOfWords
# Initialize bow model
bow = BagOfWords(min_word_freq=2, max_features=5, tokenizer='whitespace')
```

Huấn luyện trên tập corpus vừa mới tiền xử lí.

```
# Train
matrix = bow.fit_transform(preprocessed_corpus)
print(matrix)
# array([[2, 1, 1, 2, 1],    # BOW1
#        [1, 1, 1, 0, 1]])  #BOW2
```

Việc có số 0 trong hàng thứ 2 là đảm bảo tuân thủ bảo toàn số lượng đặc trưng, thuận tiện cho việc tính toán.

```
# Print vocab info (This method provides a quick look at vocabulary)
bow.print_vocabulary_info()
# Vocabulary size: 5
# Total words in corpus: 16
# Unique words in corpus: 10
# Top 10 most common words:
#   1. 'likes           ':    3
#   2. 'to              ':    2
#   3. 'watch           ':    2
#   4. 'movies          ':    2
#   5. 'mary            ':    2
```

Thử nghiệm với trường hợp gộp documents.

```
# Now we have document 3 = document 1 + document 2
doc = "John likes to watch movies. Mary likes movies too. Mary also likes to watch
    football games."
# Also preprocessed
preprocessed_doc = re.sub(r'\.', '', doc.lower())
# Transform
bow.transform_single(preprocessed_doc)
# array([3, 2, 2, 2, 2])
```

Áp dụng TruncatedSVD vào ma trận đồng xuất hiện.

```
# Initialize SVD
from embedding.BoW import SVDModel
from sklearn.decomposition import TruncatedSVD
my_svd = SVDModel(n_components=3) # We set 3 components
svd = TruncatedSVD(n_components=3)
# Apply SVD
my_X = my_svd.fit_transform(matrix)
X = svd.fit_transform(matrix)
# Print results
print(f'My SVD result: \n{my_X}')
print('---')
print(f'Scikit-learn SVD result: \n{X}')
# My SVD result:
# [[ 3.27140989  0.54578137]
#  [ 1.70327776 -1.04825803]]
# ---
# Scikit-learn SVD result:
# [[ 3.27140989 -0.54578137]
# [ 1.70327776  1.04825803]]
```

Việc sai khác dấu ở kết quả là được chấp nhận về mặt toán học.

5.2.2 TF-IDF

Lớp `TruncatedSVD`. Đây là lớp thực hiện xoay trục ma trận sử dụng TruncatedSVD. Các phương thức bao gồm:

- `fit(X)`: Thực hiện phân rã SVD. Lưu trữ các thành phần chính (*components*), giá trị kỳ dị (*singular values*), và phương sai giải thích (*explained variance*).

- `transform(X)`: Chiếu dữ liệu lên không gian thành phần chính đã học.
- `choose_n_components(threshold)`: Chọn số chiều tối ưu để giữ lại tỷ lệ phương sai mong muốn, mặc định là 95%.
- `plot_cumulative_variance()`: Vẽ đồ thị biểu diễn tỷ lệ phương sai tích lũy theo số chiều giữ lại.

Lớp `TEmbedder`. Lớp này xử lý đầu vào là các tài liệu văn bản và sinh ra biểu diễn vector rút gọn. Các phương thức bao gồm:

- `__init__`: Người dùng có thể tùy chỉnh số chiều đầu ra (`n_components`), số từ tối đa (`max_features`), và lựa chọn chuẩn hóa (`norm`).
- `fit(documents)`: Xây dựng từ điển (dạng `term` × `document`) và tính toán IDF cho từng từ. Sau đó, tính toán TF-IDF và chuẩn hóa, cuối cùng áp dụng `TruncatedSVD` để giảm chiều.
- `transform_doc(documents)`: Chuyển đổi danh sách tài liệu sang vector biểu diễn rút gọn.
- `find_best_n_components(threshold, plot)`: Dựa trên phương sai tích lũy để xác định số chiều tối ưu. Biến `plot` là biến nhị phân, dùng để thực hiện việc vẽ lượng thông tin dựa theo số chiều đầu ra.

Cách sử dụng. Khởi tạo tập tài liệu.

```
# Documentation
docs = [
    "dog cat hamster pets",    "dog chasing cat",
    "cat hiding dog",    "hamster sleeping",
    "dog protects house",    "pets cute"
]
```

Khai báo và huấn luyện mô hình.

```
from embedding.Tfidf import TEmbedder
embedder = TEmbedder(max_features=None, n_components=4, smooth_idf=False, norm=None)
embedder.fit(docs)
X = embedder.transform_doc(docs)
```

Biến đổi ma trận.

```
print("Shape:", X.shape) # -> (6,4)
print("Vocab:", embedder.vocab)
print(X.round(4)) # each row stand for each doc

# result of X
# [[-0.2348  0.      -0.0292 -0.1302]
#  [-0.0196  0.      -0.0727 -0.4918]
#  [-0.0196 -0.      -0.0727 -0.4918]
#  [-0.7403 -0.7431  0.0119  0.0331]
# [-0.0094 -0.      -0.8506  0.0895]
# [-0.7403  0.7431  0.0119  0.0331]]
```

5.2.3 PPMI

Lớp `TruncatedSVD`. Tương tự với lớp cùng tên trong TF-IDF nhưng có một chút khác biệt, lớp `TruncatedSVD` ở đây được định nghĩa lại cho phù hợp với PPMI khi sử dụng ma trận đầu vào là ma trận thưa để tối ưu hiệu suất.

Lớp `PPMIEmbedder`. Đây là lớp thực hiện xây dựng mô hình biểu diễn từ dựa trên PPMI và giảm chiều bằng TruncatedSVD. Lớp xử lý đầu vào là danh sách văn bản, đầu ra là biểu diễn từ và văn bản ở dạng vector. Các phương thức bao gồm:

- `__init__`: Khởi tạo mô hình với các tham số như kích thước cửa sổ ngữ cảnh (`window_size`), số lượng từ tối đa (`max_features`), số chiều giảm (`n_components`), tần suất tối thiểu để chọn từ (`min_count`) và số luồng xử lý song song (`n_jobs`).
- `fit(documents)`: Tiền xử lý văn bản và xây dựng từ điển ánh xạ từ → chỉ số. Sau đó, tính trọng số IDF tương ứng, xây dựng ma trận đồng xuất hiện dạng thưa. Cuối cùng, tính toán ma trận PPMI, áp dụng TruncatedSVD để giảm chiều và chuẩn hóa vector biểu diễn từ.
- `transform_docs(documents)`: Chuyển đổi danh sách tài liệu sang vector biểu diễn bằng cách lấy trung bình có trọng số IDF của các vector từ trong văn bản.
- `transform(document)`: Hỗ trợ suy diễn cho một tài liệu đơn hoặc danh sách tài liệu, trả về vector biểu diễn tương ứng.

Cách sử dụng. Chuẩn bị văn bản đầu vào.

```
# Raw corpus
corpus = [
    "Cheese, bread, butter, and milk are common dairy products.",
    "A goat eats cheese and drinks milk.",
    "Bread with cheese and butter is delicious.",
    "The goat likes cheese, milk, and a bit of butter.",
    "Bread and cheese often go with milk."
]
```

Tiền xử lý văn bản: loại bỏ dấu câu, chữ thường hóa, lọc stopwords và loại bỏ các từ xuất hiện chỉ một lần.

```
import re
from collections import Counter

# Define stopwords
stopwords = {'and','is','are','a','the','with','of','often','likes','go','bit'}
# Preprocessing: lowercasing, removing punctuation, filtering stopwords
def preprocess(doc):
    doc = re.sub(r'[^a-zA-Z\s]', '', doc.lower())
    return ' '.join([w for w in doc.split() if w not in stopwords])
# Apply preprocessing
preprocessed = [preprocess(doc) for doc in corpus]
# Remove rare words (appearing only once)
all_tokens = ' '.join(preprocessed).split()
word_counts = Counter(all_tokens)
filtered_corpus = [' '.join([w for w in doc.split() if word_counts[w] > 1]) for doc in
    preprocessed]
```

Hiển thị các văn bản sau khi lọc.

```
# Print filtered documents
print("Filtered documents:")
for i, doc in enumerate(filtered_corpus, 1):
    print(f"doc{i} =", doc.split())

# Filtered documents:
# doc1 = ['cheese', 'bread', 'butter', 'milk']
# doc2 = ['goat', 'cheese', 'milk']
# doc3 = ['bread', 'cheese', 'butter']
# doc4 = ['goat', 'cheese', 'milk', 'butter']
# doc5 = ['bread', 'cheese', 'milk']
```

Khởi tạo mô hình PPMI và huấn luyện.

```
from embedding.ppmi import PPMIEmbedder

# Initialize PPMI model
ppmi = PPMIEmbedder(
    window_size=2, min_count=2, max_features=10, n_components=4, n_jobs=1
)

# Fit model on filtered corpus
ppmi.fit(filtered_corpus)
```

In ma trận đồng xuất hiện.

```
# Print co-occurrence matrix
import numpy as np

vocab_order = ['bread', 'butter', 'cheese', 'goat', 'milk']
word2idx, cooc = ppmi.vocab, ppmi.cooc_matrix.toarray()

print("\nCo-occurrence Matrix:")
print("{:>10}".format("") + "".join(f"{w:>10}" for w in vocab_order))
for w1 in vocab_order:
    i = word2idx[w1]
    row = [f"{cooc[i, word2idx[w2]]:.0f}" for w2 in vocab_order]
    print(f"{w1:>10}" + "".join(f"{v:>10}" for v in row))

# Co-occurrence Matrix:
#                 bread     butter     cheese       goat       milk
#       bread         0          2          3          0          2
#      butter         2          0          3          0          2
#      cheese         3          3          0          2          3
#        goat         0          0          2          0          2
#        milk         2          2          3          2          0
```

Tính toán và in ma trận PPMI.

```
# Print PPMI matrix
PPMI = ppmi.ppmi_sparse.toarray()

print("\nPPMI Matrix:")
print("{:>10}".format("") + "".join(f"{w:>10}" for w in vocab_order))
for w1 in vocab_order:
    i = word2idx[w1]
    row = []
    for w2 in vocab_order:
        j = word2idx[w2]
        value = PPMI[i, j]
        row.append(f"{value:.2f}" if value > 0 else "0")
    print(f"{w1:>10}" + "".join(f"{v:>10}" for v in row))

# PPMI Matrix:
#                 bread     butter     cheese       goat       milk
#       bread         0       0.63       0.57          0       0.27
#      butter      0.63          0       0.57          0       0.27
#      cheese      0.57       0.57          0       0.79       0.20
#        goat         0          0       0.79          0       1.08
#        milk      0.27       0.27       0.20       1.08          0
```

6 Thực nghiệm

Nhằm đánh giá trực quan và so sánh hiệu quả giữa các phương pháp đã đề xuất, nhóm tác giả tiến hành thực nghiệm trên tập số liệu cụ thể, đồng thời xây dựng một trang web minh hoạ để người dùng có thể dễ dàng quan sát và tương tác với kết quả. Việc kết hợp giữa phân tích định lượng và trình bày trực quan giúp làm nổi bật ưu điểm của từng

phương pháp, đồng thời hỗ trợ người đọc có cái nhìn toàn diện và trực giác hơn về hiệu quả biểu diễn và phân biệt của các mô hình.

6.1 Thiết lập thực nghiệm

Dữ liệu. Bộ dữ liệu được thu thập từ IMDB thông qua API, với gần 10.000 mẫu dữ liệu. Tập dữ liệu này được sử dụng làm tập huấn luyện, trong khi tập kiểm tra được trích ngẫu nhiên từ tập huấn luyện với kích thước hơn 2000 dữ liệu. Tập kiểm tra sau đó được áp dụng kĩ thuật sinh văn bản [9] từ mô hình ngôn ngữ lớn Gemma3 - 27b để tạo ra các câu truy vấn mang phong cách tự nhiên, gần với cách diễn đạt của con người.

Chỉ số đánh giá. Để đánh giá hiệu quả của hệ thống, chúng tôi sử dụng các chỉ số phổ biến trong lĩnh vực truy hồi thông tin và ngôn ngữ tự nhiên:

- P@K (Precision@K), R@K (Recall@K), F1@K [10]: phản ánh mức độ cân bằng giữa dự đoán đúng và sai, rất hữu ích trong bài toán phân loại không cân bằng.
- MRR (Mean Reciprocal Rank) [11]: Trung bình nghịch đảo thứ hạng của câu trả lời đúng đầu tiên trong danh sách kết quả. Đặc biệt phù hợp cho các hệ thống chỉ cần trả về một kết quả đúng.
- MAP (Mean Average Precision) [12]: Trung bình của Precision tại mỗi vị trí có một mục đúng, qua nhiều truy vấn. MAP đánh giá toàn bộ danh sách kết quả.
- nDCG@k (Normalized Discounted Cumulative Gain) [13]: Là chỉ số đánh giá có trọng số cho các mục đúng theo thứ tự sắp xếp, với các mục đúng ở vị trí đầu có giá trị cao hơn. Được chuẩn hóa để dễ so sánh giữa các truy vấn.
- Thời gian truy vấn: Là chỉ số đánh giá UX (user experience)

Trong số các chỉ số trên, các giá trị như P@K, R@K, F1@K, MRR, MAP và nDCG@K càng cao càng thể hiện hiệu quả mô hình tốt hơn, trong khi thời gian truy vấn càng thấp càng tốt để đảm bảo trải nghiệm người dùng.

6.2 Kết quả

Bảng 7: Kết quả thực nghiệm

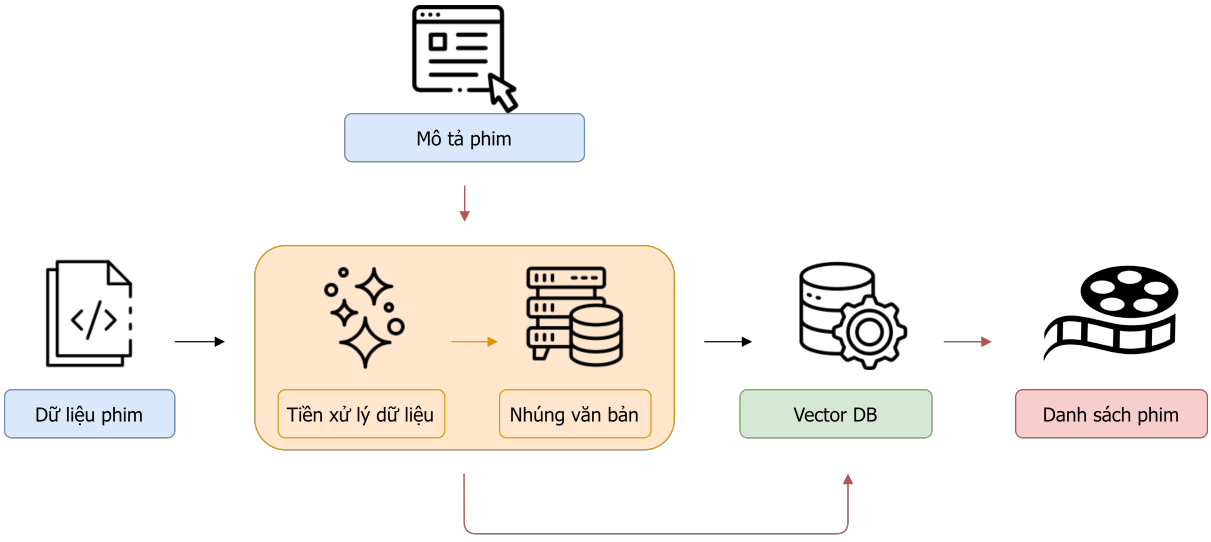
Method	P@10	R@10	F1@10	MRR	MAP	nDCG@10	Time
BoW	0.081	0.800	0.146	0.699	0.698	0.723	0.246
TF-IDF	0.072	0.714	0.130	0.562	0.561	0.598	0.236
PPMI	0.051	0.503	0.092	0.441	0.441	0.456	0.212
HellingerPCA	0.038	0.381	0.069	0.265	0.265	0.293	0.230
Word2Vec	0.048	0.482	0.09	0.446	0.446	0.456	0.246
GloVe	**0.107**	**0.816**	**0.196**	**0.784**	**0.783**	0.734	0.224
FastText	0.102	0.796	0.166	0.754	0.753	**0.764**	**0.143**

Ba phương pháp cơ bản gồm BoW, TF-IDF, và PPMI được xem như các baseline để đối chiếu hiệu quả với các kỹ thuật nâng cao. Trong số này, BoW đạt kết quả tốt nhất

với Precision@10 (P@10) là 0.081 và Recall@10 (R@10) lên đến 0.800 - cao hơn đáng kể so với TF-IDF và PPMI. Đồng thời, BoW cũng có MRR và MAP cao nhất trong nhóm baseline (0.699 và 0.698), cho thấy khả năng truy xuất tài liệu đúng thứ tự khá hiệu quả dù biểu diễn còn thô. TF-IDF - vốn phổ biến nhờ việc gán trọng số theo tầm quan trọng của từ - lại có độ chính xác thấp hơn (P@10 = 0.072), và toàn bộ chỉ số truy xuất đều kém hơn BoW, ngoại trừ thời gian xử lý có phần nhanh hơn. PPMI, dù có lợi thế về khả năng nắm bắt thông tin đồng xuất hiện, lại là phương pháp có hiệu suất thấp nhất trong nhóm, với F1@10 chỉ 0.092 và nDCG@10 thấp (0.456), cho thấy cách tiếp cận này khó cạnh tranh nếu không có bước giảm chiều hoặc lọc nhiễu đi kèm.

Ngược lại, các phương pháp thay thế như Word2Vec, GloVe, FastText, và biến thể HellingerPCA cho thấy hiệu quả vượt trội về mặt ngữ nghĩa. GloVe là phương pháp nổi bật nhất trong toàn bộ bảng, với P@10 đạt 0.107, F1@10 cao nhất là 0.196, và các chỉ số MRR, MAP đều trên 0.78 - chứng minh khả năng học tốt mối quan hệ ngữ nghĩa toàn cục từ văn bản. FastText tuy có Precision thấp hơn một chút (0.100), nhưng lại có thời gian xử lý nhanh nhất (0.143s), đồng thời vẫn duy trì hiệu suất tốt với nDCG@10 cao nhất (0.764), phù hợp với các hệ thống cần thời gian phản hồi nhanh. Word2Vec và HellingerPCA có kết quả thấp hơn, với HellingerPCA tỏ ra chưa hiệu quả khi chỉ đạt F1@10 là 0.069 và MRR là 0.265, cho thấy phương pháp giảm chiều bằng Hellinger + PCA có thể cần thêm tinh chỉnh để đạt được biểu diễn ngữ nghĩa ổn định. Nhìn chung, các mô hình embedding hiện đại đều vượt trội so với baseline truyền thống, không chỉ về độ chính xác mà còn ở khả năng sắp xếp tài liệu theo mức độ liên quan ngữ nghĩa trong bài toán truy xuất.

6.3 Xây dựng website ứng dụng LSA vào hệ thống truy xuất phim phim dựa trên mô tả



Hình 5: Pipeline xử lý dữ liệu và truy vấn tìm kiếm phim

Nhóm tác giả đã phát triển một ứng dụng web với pipeline được mình họa ở Hình 5. Ứng dụng web này cho phép người dùng nhập một đoạn mô tả nội dung phim (bằng tiếng Anh) sau đó lựa chọn mô hình và nhận lại danh sách các phim có nội dung tương đồng

nhất với mô tả đó. Mục tiêu chính là giúp người dùng tìm lại phim khi họ chỉ nhớ nội dung mà không nhớ tên phim, diễn viên hay năm phát hành thông qua các mô hình.

Chức năng chính của ứng dụng. Ứng dụng cho phép người dùng nhập mô tả nội dung hoặc thể loại phim mong muốn trực tiếp và chọn mô hình trên giao diện web. Sau khi tiếp nhận văn bản đầu vào, hệ thống sẽ xử lý mô tả này và ánh xạ nó thành vector nhúng thông qua các mô hình biểu diễn ngôn ngữ đã huấn luyện. Quá trình truy vấn được thực hiện trên cơ sở dữ liệu vector sử dụng công cụ tìm kiếm tương đồng, từ đó trả về danh sách các bộ phim có nội dung gần giống nhất. Kết quả hiển thị bao gồm tên phim, hình ảnh poster phim và các thông tin của phim trong cơ sở dữ liệu.

So sánh các thuật toán biểu diễn văn bản. Ứng dụng được thiết kế với khả năng tích hợp và so sánh nhiều kỹ thuật biểu diễn văn bản khác nhau, bao gồm TF-IDF, Word2Vec, FastText, PPMI, GloVe, Bag-of-Words (BoW) và HellingerPCA. Khi người dùng nhập mô tả phim,và chọn mô hình mong muốn hệ thống sẽ lần lượt ánh xạ mô tả này thành vector nhúng theo mô hình đã chọn, sau đó thực hiện truy vấn tìm kiếm trên cơ sở dữ liệu tương ứng để thu về danh sách các phim tương tự. Điều này cho phép hiển thị và so sánh trực quan kết quả giữa các mô hình khác nhau, hỗ trợ người dùng hoặc nhà nghiên cứu dễ dàng đánh giá hiệu quả, mức độ phù hợp và độ chính xác của từng phương pháp biểu diễn ngữ nghĩa.

Giao diện web đơn giản, dễ sử dụng. Giao diện của ứng dụng được xây dựng bằng Flask với thiết kế tối giản, trực quan, giúp người dùng dễ dàng nhập mô tả nội dung phim (hoặc dùng hệ thống nhận diện giọng nói tiếng Anh để nhập) và lựa chọn mô hình biểu diễn ngữ nghĩa để thực hiện tìm kiếm. Sau khi truy vấn được xử lý, hệ thống hiển thị kết quả một cách rõ ràng và trực quan, bao gồm tên phim, thông tin phim và hình ảnh poster phim đi kèm. Giao diện được tối ưu nhằm phục vụ mục đích thử nghiệm, so sánh mô hình cũng như hỗ trợ nghiên cứu trong lĩnh vực truy xuất thông tin.

Pipeline xử lý dữ liệu rõ ràng, có thể mở rộng. Pipeline của hệ thống được thiết kế mạch lạc và dễ dàng mở rộng. Đầu tiên, mô tả phim được tiền xử lý bao gồm các bước làm sạch dữ liệu, tách từ và chuẩn hoá văn bản. Sau đó, các mô hình biểu diễn ngữ nghĩa như TF-IDF, Word2Vec, PPMI hay GloVe sẽ được sử dụng để chuyển văn bản thành các vector đặc trưng. Những vector này sẽ được lưu trữ vào cơ sở dữ liệu vector Qdrant nhằm phục vụ việc truy vấn nhanh chóng. Khi người dùng nhập một mô tả mới, hệ thống sẽ chuyển đổi mô tả này thành vector tương ứng, truy vấn lên Qdrant để tìm ra top-k phim có vector gần nhất về mặt ngữ nghĩa sử dụng độ tương đồng Cosine. Nhờ thiết kế linh hoạt, pipeline này cho phép tích hợp thêm các mô hình mới hoặc mở rộng sang các tập dữ liệu lớn hơn mà không làm ảnh hưởng đến hiệu năng tổng thể.

7 Kết luận

LSA là phương pháp cổ điển nhưng là nền tảng vững chắc trong lĩnh vực xử lý ngôn ngữ tự nhiên, đặc biệt là trong các hệ thống truy hồi thông tin. Một trong những ưu điểm lớn của LSA là khả năng phát hiện mối quan hệ ngữ nghĩa giữa các từ và tài liệu, nhờ đó xử lý tốt hiện tượng đồng nghĩa. Việc giảm chiều dữ liệu thông qua SVD cũng giúp loại bỏ nhiễu, tiết kiệm chi phí tính toán và lưu trữ mà vẫn giữ lại được những yếu tố

cốt lõi về ngữ nghĩa. Ngoài ra, LSA không yêu cầu lượng dữ liệu lớn, điều này đặc biệt hữu ích trong các hệ thống IR ở những lĩnh vực có tài nguyên hạn chế như y tế, pháp lý hay thư viện số. Việc triển khai LSA cũng khá đơn giản nhờ nền tảng toán học rõ ràng, dễ kiểm soát và dễ diễn giải.

Tuy nhiên, LSA cũng mang nhiều hạn chế. Mô hình không giải quyết được vấn đề đa nghĩa, ví dụ, từ *đá* trong *đá lạnh* và *đá bóng* sẽ có cùng biểu diễn vector, dẫn đến mất thông tin ngữ cảnh quan trọng trong IR. Bên cạnh đó, LSA rất phụ thuộc vào bước tiền xử lý, đặc biệt với tiếng Việt hoặc tiếng Đức. Về mặt tính toán, việc phân rã SVD trên các ma trận lớn và thưa cũng đòi hỏi tài nguyên cao. Cuối cùng, việc lựa chọn số chiều giữ lại sau SVD là một bài toán đánh đổi: nếu giữ quá ít sẽ làm mất thông tin quan trọng, còn nếu giữ quá nhiều sẽ dẫn đến lời nguyền chiều cao, làm giảm độ chính xác của các phép đo tương đồng hoặc khoảng cách - vốn là cốt lõi trong IR.

Những khuyết điểm của LSA có thể được cải thiện bằng các phương pháp hiện đại hơn là Word2Vec, GloVe, FastText, hoặc HellingerPCA. Word2Vec có khả năng học được các mối quan hệ ngữ nghĩa và cú pháp giữa các từ thông qua ngữ cảnh. GloVe tận dụng thống kê toàn cục của văn bản, giúp mô hình học được các đặc trưng ngữ nghĩa ổn định ngay cả khi không có ngữ cảnh gần. Trong khi đó, FastText cải tiến hơn bằng cách biểu diễn từ dưới dạng các n-gram ký tự, cho phép mô hình học được cấu trúc hình thái và tạo vector cho cả những từ chưa từng xuất hiện trong tập huấn luyện (OOV). HellingerPCA mặc dù không đạt được kết quả cao trong bài toán truy xuất phim, nhưng vẫn còn có tiềm năng lớn trong các bài toán NLP nhờ kiến trúc đơn giản.

Tóm lại, LSA là một phương pháp hiệu quả, dễ triển khai và vẫn rất phù hợp cho các hệ thống truy hồi thông tin yêu cầu xử lý ngữ nghĩa nhưng không có điều kiện triển khai các mô hình học sâu hiện đại. Từ Bảng 7 cho thấy, với ứng dụng tìm kiếm phim dựa trên mô tả có lượng dữ liệu không quá lớn (gần 10.000 mẫu dữ liệu), LSA vẫn hoạt động vô cùng tốt và vẫn có thể cạnh tranh với các phương pháp hiện đại hơn. Với sự cân bằng giữa tính ngữ nghĩa, chi phí và độ diễn giải, LSA vẫn là một công cụ đáng tin cậy cho nhiều ứng dụng IR hiện nay.

References

[1] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936. DOI: 10.1007/BF02288367.

[2] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychological Review*, vol. 104, pp. 211–240, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:1144461.

[3] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975, ISSN: 0001-0782. DOI: 10.1145/361219.361220. [Online]. Available: https://doi.org/10.1145/361219.361220.

[4] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[5] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational Linguistics*, vol. 16, no. 1, pp. 22–29, 1990. [Online]. Available: https://aclanthology.org/J90-1003/.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1301.3781.

[7] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162/.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[9] H. A. Rahmani, N. Craswell, E. Yilmaz, B. Mitra, and D. Campos, "Synthetic test collections for retrieval evaluation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '24, Washington DC, USA: Association for Computing Machinery, 2024, pp. 2647–2651, ISBN: 9798400704314. DOI: 10.1145/3626772.3657942. [Online]. Available: https://doi.org/10.1145/3626772.3657942.

[10] C. J. V. Rijsbergen, *Information Retrieval*, 2nd. USA: Butterworth-Heinemann, 1979, ISBN: 0408709294.

[11] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," en, *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[12] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '00, Athens, Greece: Association for Computing Machinery, 2000, pp. 33–40, ISBN: 1581132263. DOI: 10.1145/345508.345543. [Online]. Available: https://doi.org/10.1145/345508.345543.

[13] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002, ISSN: 1046-8188. DOI: 10.1145/582415.582418. [Online]. Available: https://doi.org/10.1145/582415.582418.