Tìm kiếm dữ liệu bằng vector embedding và vector Databases trong hệ thống chatbot.

Tóm tắt:

Nghiên cứu này giới thiệu một hệ thống sử dụng kỹ thuật vector embedding và vector database nhằm cải thiện quá trình tìm kiếm dữ liệu liên quan cho hệ thống chatbot quy mô lớn. Phương pháp đề xuất sử dụng mô hình embedding để biểu diễn dữ liệu dưới dạng vector trong không gian đa chiều, và lưu trữ trong vector database cho phép truy vấn dữ liệu dựa trên tính tương đồng vector. Các thử nghiệm cho thấy hệ thống của chúng tôi đạt hiệu suất cao về tốc độ và chính xác khi tìm kiếm dữ liệu liên quan, đồng thời khả năng mở rộng để xử lý dữ liệu lớn tốt hơn so với các phương pháp truyền thống. Kết quả đạt được đóng góp giải pháp ứng dụng mới vào lĩnh vực trợ lý ảo và chatbot thông minh. Trong tương lai, nghiên cứu sẽ mở rộng...

1. Giới thiệu .

1.1 Bối cảnh và tầm quan trọng của vấn đề tìm kiếm, lọc dữ liệu liên quan.

Tháng 6 năm 2020, OpenAI đã cho ra mắt ChatGPT, một mô hình ngôn ngữ lớn có khả năng tạo ra các cuộc trò chuyện tự nhiên và trả lời chính xác câu hỏi của người dùng [1]. ChatGPT đã nhanh chóng thu hút sự chú ý của công chúng và trở thành một hiện tượng văn hóa, đồng thời khơi dậy làn sóng quan tâm và đầu tư vào công nghệ AI. Sự phổ biến của ChatGPT và các mô hình ngôn ngữ lớn khác đã góp phần thúc đẩy sự gia tăng lượng dữ liệu được tạo ra và lưu trữ là do các mô hình này được đào tạo trên một lượng lớn dữ liệu văn bản và cần dữ liệu bổ sung để tiếp tục cải thiện và phát triển [2].

Ngay sau đó, hàng loạt hệ thống chatbot hay hệ thống đối thoại dựa trên ngôn ngữ, đã trở thành một công cụ quan trọng trong nhiều lĩnh vực, bao gồm hỗ trợ khách hàng, giáo dục, y tế và thương mại điện tử [3]. Với khả năng tương tác bằng ngôn ngữ tự nhiên, chatbot có thể tự động hóa các nhiệm vụ như trả lời câu hỏi, cung cấp thông tin và thậm chí thực hiện các giao dịch [4]. Tuy nhiên, dữ liệu mà chatbot phải xử lý thường là phi cấu trúc và đến từ nhiều nguồn khác nhau, chẳng hạn như văn bản, hình ảnh, video và mạng xã hội [5]. Điều này đặt ra thách thức cho các phương pháp tìm kiếm và lọc dữ liệu truyền thống đặc biệt là khi các dữ liệu lại được lưu trữ bằng các database như: Mongodb, Sql, Oracle,... Nếu không thể tìm kiếm và lọc thông tin liên quan một cách hiệu quả, chatbot có thể đưa ra câu trả lời không chính xác, không đầy đủ hoặc mất nhiều thời gian để phản hồi, dẫn đến trải nghiệm người dùng kém [6].

Các nghiên cứu gần đây chỉ ra rằng khả năng tìm kiếm và lọc dữ liệu liên quan có tác động trực tiếp đến chất lượng và hiệu suất của các hệ thống chatbot. Trong [5], các tác giả đã chỉ ra rằng việc cải thiện khả năng tìm kiếm thông tin liên quan có thể làm tăng 8-20% độ chính xác của chatbot trong các tác vụ trả lời câu hỏi. Vì vậy, việc nghiên cứu và phát triển các phương pháp tìm kiếm, lọc dữ liệu liên quan hiệu quả và các phương pháp này cần có khả năng xử lý dữ liệu phi cấu trúc, hiểu ngữ nghĩa của truy vấn và mở rộng cho

dữ liệu lớn một cách hiệu quả., đặc biệt là cho các hệ thống chatbot quy mô lớn, là một vấn đề cấp thiết và có tầm quan trọng cao trong lĩnh vực này.

1.2 Thách thức trong việc tìm kiếm và lọc dữ liệu liên quan.

Theo một báo cáo của IDC được đề cập bởi Patrizio [7] dự kiến năm 2025 lượng dữ liệu trên toàn thế giới sẽ đạt 175 zettabytes tăng gấp 5 lần so với 33 zettabytes của năm 2018 và sự gia tăng này được thúc đẩy chính bởi dữ liệu từ điện toán đám mây, di động, video giải trí và thiết bị IoT . Với lượng dữ liệu tăng theo cấp số nhân tăng theo từng năm thì đây có lẽ là thách thức về tìm kiếm và lọc dữ liệu liên quan cho chatbot để trả lời chính xác câu hỏi của người dùng trở nên cực kỳ khó khăn và tiêu tốn rất nhiều thời gian và công sức cho các nhà phát triển vì một vài lý do sau:

- Hệ thống chatbot tìm kiếm và truy xuất thông tin liên quan đến câu hỏi của người dùng sẽ trở nên khó khăn hơn là do lượng dữ liệu phi cấu trúc quá lớn và đa dạng từ nhiều nguồn khác nhau dẫn đến độ sai lệch kết quả rất cao gây tốn kém chi phí tính toán và ảnh hưởng đến trải nghiệm của người dùng [5].
- Nhà phát triển luôn muốn hệ thống chatbot có thể trả lời các câu hỏi phức tạp nên chatbot phải có khả năng tổng hợp và xử lý dữ liệu liên quan từ nhiều nguồn thông tin khác nhau. Việc xử lý cùng một lúc nhiều nguồn dữ liệu dẫn đến độ nhiễu và sai lệch càng tăng gây ảnh hưởng đến kết quả tìm kiếm và nguy hiểm hơn là câu hỏi có thể sai hoặc không đúng ý người dùng [8].
- Khi cơ sở dữ liệu ngày càng lớn thì các database truyền thống sẽ mất nhiều thời gian hơn để tìm kiếm và truy vấn dữ liệu để đem đi huấn luyện và phản hồi câu hỏi của người dùng, dẫn đến hiệu suất thấp và trải nghiệm người dùng kém [9].
- Khối lượng dữ liệu càng lớn thì tỉ lệ thuận với số tiền cần bỏ ra để lưu trữ, tổ chức và xử lý một cách hiệu quả và muốn làm được điều đó thì phải đòi hỏi hạ tầng máy chủ, lưu trữ đắt tiền và tiêu tốn nhiều năng lượng [2].

1.3 Giới hạn của một số phương pháp tìm kiếm thống truyền.

1.3.1 Phương pháp tìm kiếm dựa trên từ khóa.

Phương pháp tìm kiếm dựa trên từ khóa (Keyword-based Search) là một trong những kỹ thuật tìm kiếm thông tin lâu đời và phổ biến nhất. Nó hoạt động bằng cách so sánh các từ khóa trong truy vấn của người dùng với các từ khóa có trong tài liệu. Các tài liệu có chứa nhiều từ khóa phù hợp nhất sẽ được ưu tiên và trả về cho người dùng [10].

Phương pháp này chỉ tập trung vào sự xuất hiện của các từ khóa mà không quan tâm đến ngữ nghĩa của chúng. Điều này có thể dẫn đến việc bỏ sót các tài liệu liên quan hoặc đưa vào các tài liệu không liên quan. Ví dụ, một truy vấn tìm kiếm về "apple" có thể trả về các tài liệu về trái cây hoặc công ty Apple, ngay cả khi người dùng chỉ quan tâm đến một trong hai nghĩa đó [11]. Việc so sánh các từ khóa trong truy vấn với tất cả

các từ khóa trong tập dữ liệu lớn có thể rất tốn kém về mặt tính toán [9]. Hạn chế này trở nên nghiêm trọng hơn khi xử lý dữ liệu phi cấu trúc. Phương pháp tìm kiếm dựa trên từ khóa có thể không phù hợp để tìm kiếm và lọc dữ liệu liên quan cho các hệ thống chatbot có quy mô lớn.

1.3.2 Phương pháp truy vấn ngữ nghĩa.

Phương pháp truy vấn ngữ nghĩa (Semantic Query) là một kỹ thuật tìm kiếm thông tin cố gắng hiểu ý nghĩa của truy vấn của người dùng và tìm kiếm các tài liệu có liên quan về mặt ngữ nghĩa. Bằng cách hiểu ý nghĩa của truy vấn và ngữ cảnh của dữ liệu. Hệ thống phân tích truy vấn, ánh xạ các thành phần của nó vào cơ sở tri thức - một kho lưu trữ thông tin có cấu trúc về các thực thể, thuộc tính và mối quan hệ - sau đó sử dụng suy luận logic để rút ra kết luận và tạo truy vấn dữ liệu có cấu trúc. Cuối cùng, hệ thống trả về kết quả phù hợp nhất [12].

Một trong những giới hạn chính của phương pháp truy vấn ngữ nghĩa là khó khăn trong việc xử lý các truy vấn phức tạp chứa nhiều mệnh đề hoặc các truy vấn yêu cầu suy luận logic [13]. Khi xử lý dữ liệu có kích cỡ lớn và dữ liệu phi cấu trúc thì phương pháp này trở nên kém hiệu quả và tốn kém về mặt tính toán khi xử lý lượng dữ liệu khổng lồ vì phương pháp truy vấn ngữ nghĩa thường được thiết kế để xử lý dữ liệu có cấu trúc, chẳng hạn như dữ liệu được lưu trữ trong cơ sở dữ liệu quan hệ [14].

Mặc dù phương pháp truy vấn ngữ nghĩa có thể cải thiện hiệu quả tìm kiếm và lọc dữ liệu cho chatbot, nhưng nó vẫn có một số giới hạn, đặc biệt là khi xử lý dữ liệu có quy mô rất lớn.

1.3.3 Phương pháp tìm kiếm dựa trên thông tin trích xuất.

Các kỹ thuật dựa trên thông tin trích xuất như Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) và Đại số Ternary phụ thuộc vào các phép tính trên ma trận từ-tài liệu hoặc cấu trúc dữ liệu đặc biệt. Chúng thường chỉ xem xét sự xuất hiện của các từ riêng lẻ hoặc cụm từ, mà không tính đến bối cảnh ngữ nghĩa rộng lớn hơn hoặc ngữ cảnh tự nhiên của câu hỏi và dữ liệu đầu vào.

Các tác giả trong [15] đã chỉ ra rằng thời gian xử lý của LSA và LDA tăng lên đáng kể khi kích thước dữ liệu vượt quá 10 gigabyte và điều này rất khó để các hệ thống chatbot có quy mô dữ liệu lớn sử dụng để tìm kiếm và lọc các dữ liên quan.

Phương pháp dựa trên thông tin trích xuất đã đóng góp lớn cho lĩnh vực tìm kiếm thông tin nhưng chúng gặp phải giới hạn về khả năng xử lý ngôn ngữ tự nhiên phức tạp và khó mở rộng cho dữ liệu lớn trong bối cảnh của các hệ thống chatbot hiện đại.

1.3.4 Phương pháp tìm kiếm dựa trên xếp hạng.

TF-IDF (Term Frequency-Inverse Document Frequency) là một trong những phương pháp tìm kiếm dựa trên xếp hạng phổ biến nhất để đánh giá mức độ liên quan của tài liệu trong hệ thống tìm kiếm thông tin dựa trên tần suất xuất hiện của các từ [16]. Mặc dù đơn giản và hiệu quả trong một số trường hợp, TF-IDF lại bỏ qua ngữ nghĩa của từ và mối quan hệ giữa các từ.

Phương pháp TF-IDF chỉ tập trung vào tần suất xuất hiện của từ mà không quan tâm đến ngữ nghĩa hay mối quan hệ giữa các từ. Điều này dẫn đến việc đánh giá sai mức độ liên quan của tài liệu, đặc biệt khi các từ có nhiều nghĩa hoặc xuất hiện trong các ngữ cảnh khác nhau. Ví dụ, từ "bank" có thể mang nghĩa là "bờ sông" hoặc "ngân hàng", nhưng TF-IDF không thể phân biệt được hai nghĩa này [11]. Ngoài ra TF-IDF là không thể phân biệt được các từ đồng nghĩa hoặc trái nghĩa, dẫn đến việc bỏ sót các tài liệu liên quan hoặc đưa vào các tài liệu không liên quan.

Phương pháp TF-IDF sẽ có xu hướng ưu tiên các tài liệu dài hơn, ngay cả khi chúng không có nội dung liên quan vì TF-IDF không tính đến ngữ nghĩa và chỉ xem xét tần suất xuất hiện [16]. Điều này dẫn đến kết quả của câu hỏi sau tìm kiếm được đem đi huấn luyện bởi các mô hình và trả về cho người dùng sẽ không được chính xác và gây ảnh hưởng trải nghiệm xấu, tiêu tốn tài nguyên và chi phí huấn luyện cho dữ liệu.

Một phương pháp cải tiến hơn của TF-IDF là BM25 (Best Match 25), nó xem xét độ dài tài liệu và tần suất xuất hiện của từ trong toàn bộ tập dữ liệu. Nhưng nó vẫn gặp khó khăn trong việc nắm bắt ngữ nghĩa và xử lý các từ đồng nghĩa/trái nghĩa [17].

1.4 Mục tiêu và đóng góp của nghiên cứu.

Nghiên cứu nhằm giải quyết hạn chế của các phương pháp tìm kiếm truyền thống khi đối mặt với dữ liệu lớn, phi cấu trúc và truy vấn phức tạp trong chatbot quy mô lớn. Để đạt mục tiêu này, nghiên cứu tập trung vào việc sử dụng kỹ thuật vector embedding và vector database để tối ưu tìm kiếm và lọc dữ liệu liên quan cho chatbot.

Đóng góp chính:

- Sử dụng vector embedding để biểu diễn nhiều loại dữ liệu khác nhau như: văn bản, hình ảnh, âm thanh dưới dạng vector, đo lường sự tương đồng ngữ nghĩa, nâng cao khả năng xử lý ngôn ngữ tự nhiên, dữ liệu phi cấu trúc, tăng tính linh hoạt và khả năng mở rộng về sau cho hệ thống chatbot.
- Lưu trữ dữ liệu vector trong vector database, cho phép hệ thống mở rộng quy mô, tìm kiếm gần đúng hiệu quả, tăng tốc xử lý, giảm chi phí tính toán.
- Kết hợp kỹ thuật vector embedding và vector database mang lại giải pháp hiệu quả, sáng tạo để giải quyết thách thức tìm kiếm, lọc dữ liệu liên quan cho hệ thống chatbot quy mô lớn.

2. Tổng quan về vector embedding và vector database .

2.1 Cách thức hoạt động của Vector Embedding.

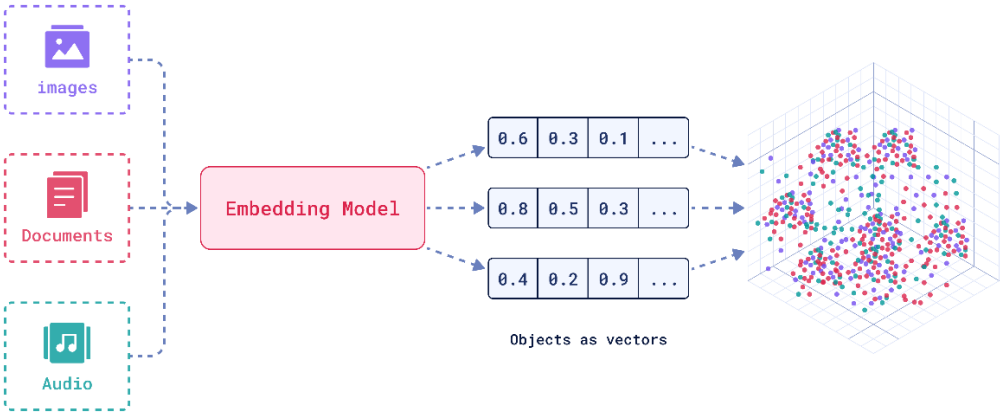
2.1.1 Các mô hình embedding.

Các mô hình embedding như Word2Vec, GloVe và BERT là những kỹ thuật phổ biến để biểu diễn dữ liệu văn bản dưới dạng vector trong không gian nhiều chiều. Các mô hình này sử dụng các phương pháp học máy để học cách biểu diễn các từ hoặc câu dựa trên những mối quan hệ ngữ nghĩa và ngữ pháp giữa chúng trong văn bản.

Theo Mikolov et al. [18], Word2Vec sử dụng mạng nơ-ron nông để học hai mô hình biểu diễn từ vựng: Skip-gram và CBOW. Mô hình Skip-gram học cách dự đoán một từ dựa trên các từ xung quanh, trong khi CBOW dự đoán các từ xung quanh dựa trên một từ đầu vào. Quá trình học tập này giúp các vector từ có độ tương đồng cao giữa các từ có nghĩa tương tự.

Mô hình GloVe [19] sẽ học các vector biểu diễn bằng cách phân tích thống kê ma trận đồng xuất hiện của các từ, khai thác cả thông tin cục bộ và toàn cục về các từ. Điều này cho phép GloVe học được biểu diễn vector có ý nghĩa ngữ nghĩa phong phú.

Một mô hình embedding tiên tiến hơn là BERT [20] sử dụng mạng Transformer hai chiều để học biểu diễn ngữ cảnh nhạy cảm của các từ giúp cho mô hình này biểu diễn chính xác hơn ý nghĩa của từ dựa trên cả ngữ cảnh đi trước và đi sau.



Hình 1 Luồng hoạt động của embedding model

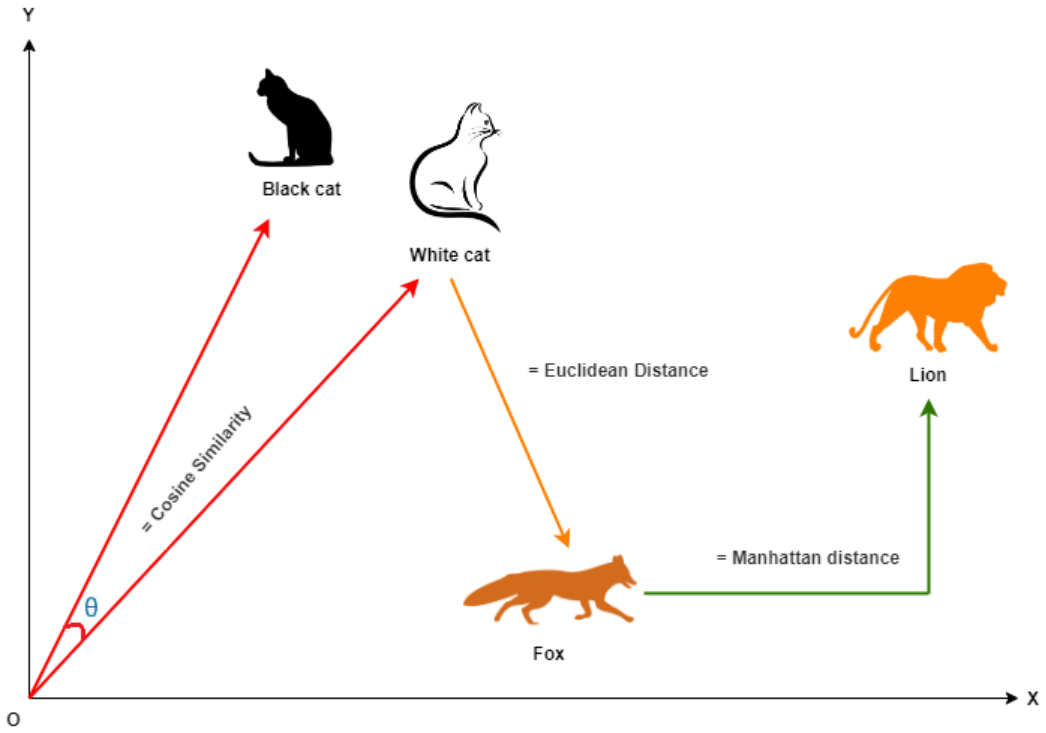
(Nguồn ảnh: https://qdrant.tech/articles/what-are-embeddings)

Các mô hình embedding như Word2Vec, GloVe và BERT đều sử dụng các kỹ thuật học máy khác nhau để học cách biểu diễn các từ, câu hoặc văn bản dưới dạng vector số, phản ánh mối quan hệ ngữ nghĩa giữa chúng.

2.1.2 Đo lường tương tự ngữ nghĩa dựa trên vector.

Sau khi sử dụng các mô hình embedding như Word2Vec, GloVe hay BERT để biểu diễn các từ, câu và tài liệu dưới dạng vector trong không gian nhiều chiều,

việc tiếp theo là định lượng mức độ tương tự ngữ nghĩa giữa các vector này bằng cách sử dụng các phương pháp như: Cosine similarity, Euclidean distance và Manhattan distance.



Hình 2 - Minh họa mức độ tương tự ngữ nghĩa trong không gian 2 chiều

Cosine similarity [18] là một trong những phương pháp phổ biến nhất để so sánh độ tương đồng giữa hai vector. Nó tính toán cosine của góc giữa hai vector, cho ra một giá trị trong khoảng từ -1 đến 1. Hai vector càng giống nhau thì cosine của góc giữa chúng càng gần 1, nghĩa là độ tương đồng càng cao. Ưu điểm của cosine similarity là nó không bị ảnh hưởng bởi độ dài của vector, mà chỉ quan tâm đến hướng của vector.

Công thức tính cosine là: cos(u, v) = u · v / (||u|| · ||v||)

Trong đó:

- u · v là tích vô hướng (dot product) của hai vector u và v
- ||u|| và ||v|| là độ dài (norm) của vector u và v, tính theo công thức:
- ||u|| = √(u1^2 + u2^2 + ... + un^2)

Cosine similarity cho ra giá trị trong khoảng [-1, 1], với:

- cos(u, v) = 1 khi u và v hoàn toàn giống nhau (góc giữa chúng là 0 độ)
- cos(u, v) = 0 khi u và v hoàn toàn độc lập (góc giữa chúng là 90 độ)
- cos(u, v) = -1 khi u và v hoàn toàn trái ngược (góc giữa chúng là 180 độ)

Euclidean distance [19] tính khoảng cách Euclide giữa hai vector trong không gian vector. Hai vector càng gần nhau thì khoảng cách Euclide càng nhỏ, tức độ tương tự ngữ nghĩa càng cao. Ưu điểm của Euclidean distance là đơn giản và dễ tính toán, nhưng nó bị ảnh hưởng bởi độ dài của vector.

Công thức tính khoảng cách Euclide là: d(u, v) = √Σ(ui - vi)^2
Trong đó:

- u = (u1, u2, ..., un) và v = (v1, v2, ..., vn) là hai vector n chiều.
- ui là giá trị tại chiều thứ i của vector u.
- vi là giá trị tại chiều thứ i của vector v.

Manhattan Distance [21] tính tổng giá trị chênh lệch tuyệt đối của từng chiều giữa hai vector. Tương tự Euclidean distance, hai vector càng gần nhau thì Manhattan distance càng nhỏ, tức độ tương tự ngữ nghĩa càng cao. Manhattan distance ít bị ảnh hưởng bởi độ dài vector hơn Euclidean distance.
Công thức tính khoảng cách Manhattan là: d(u, v) = Σ |ui - vi|
Trong đó:

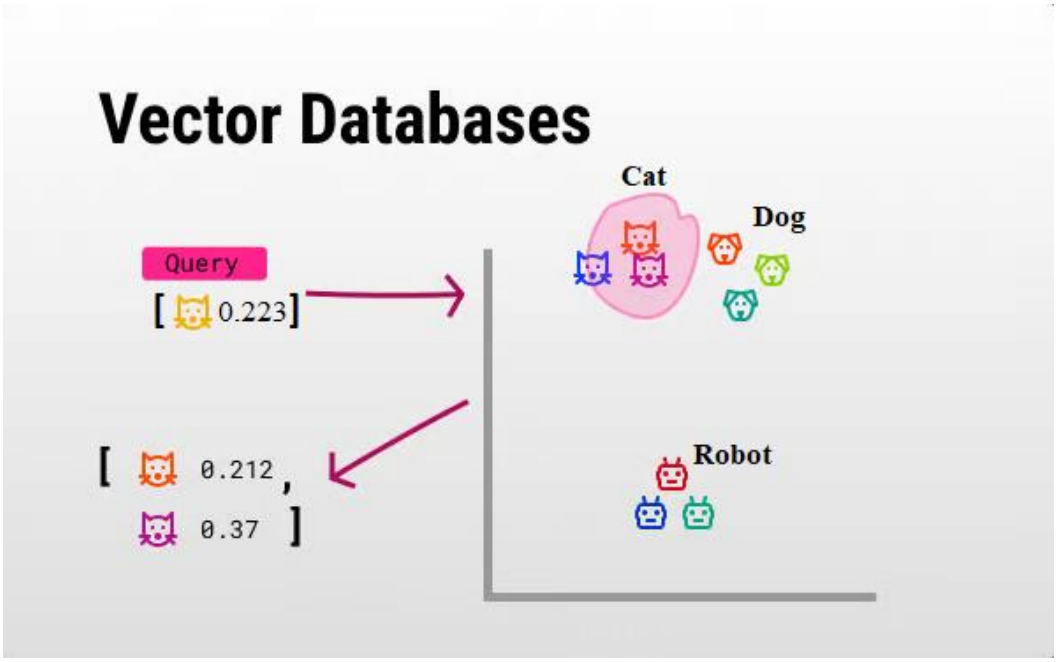
- u = (u1, u2, ..., un) và v = (v1, v2, ..., vn) là hai vector n chiều.
- ui là giá trị tại chiều thứ i của vector u.
- vi là giá trị tại chiều thứ i của vector v.

2.2 Tổng quan về Vector Database.

2.2.1 Lưu trữ dữ liệu dạng vector

Sau khi sử dụng các mô hình embedding như: Word2Vec, GloVe hoặc BERT để biểu diễn dữ liệu văn bản, hình ảnh, âm thanh dưới dạng các vector, một thách thức tiếp theo là tổ chức và lưu trữ các vector này một cách hiệu quả.

Các database truyền thống như SQL hay NoSQL thường không phù hợp để lưu trữ và truy vấn các vector đa chiều này. Theo Cattell [9], khi kích thước dữ liệu tăng lên, hiệu suất của các database truyền thống sẽ giảm đi đáng kể do những hạn chế trong cấu trúc lưu trữ và cơ chế truy vấn. Do đó, các vector database như Faiss, Milvus, Annoy, HNSW được phát triển để giải quyết vấn đề này bằng cách cung cấp các cấu trúc và thuật toán tối ưu để lưu trữ, tổ chức và truy vấn dữ liệu dạng vector một cách hiệu quả.



Hình 3-Hình ảnh minh họa vector database

(Nguồn ảnh: https://www.e2enetworks.com/blog/an-introduction-to-vector-databases)

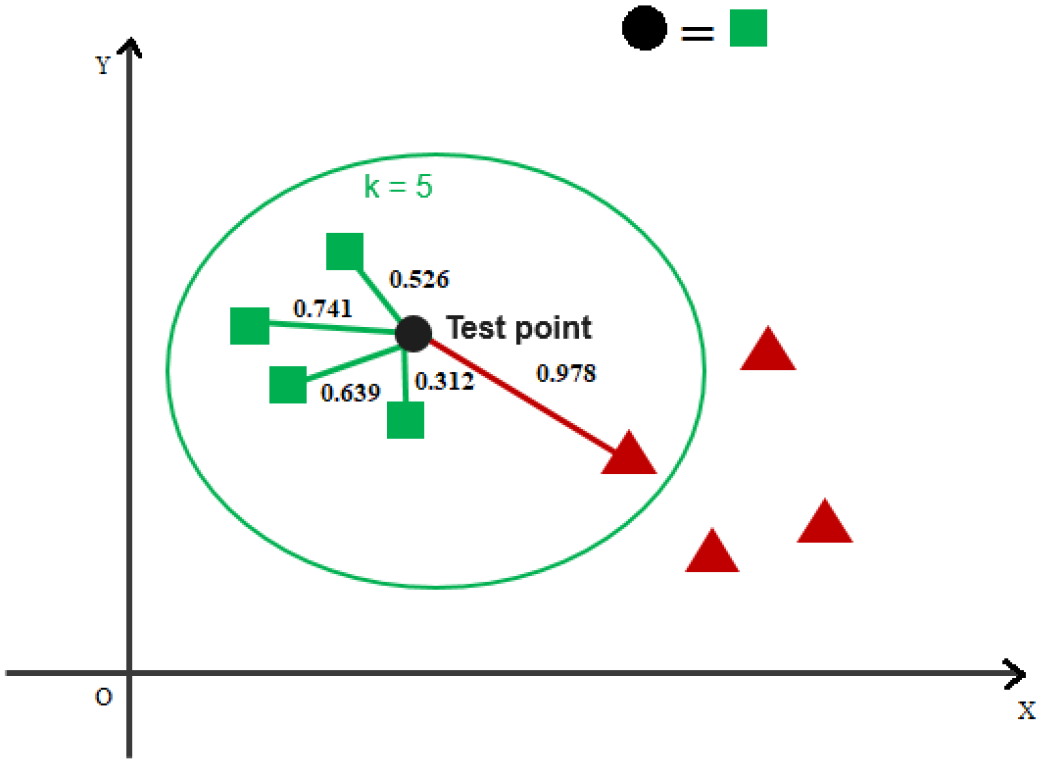
Các vector database này thường sử dụng các cấu trúc dữ liệu như inverted index, hashing, tree-based index để tổ chức các vector [22]. Nhờ đó, chúng có thể thực hiện các truy vấn tìm kiếm gần đúng (approximate nearest neighbor search) với độ chính xác và hiệu suất cao, thích hợp cho các ứng dụng như tìm kiếm ảnh, tìm kiếm văn bản dựa trên ngữ nghĩa, và cả những hệ thống chatbot quy mô lớn [23].

2.2.2 Tìm kiếm gần đúng và xếp hạng dựa trên khoảng cách vector.

Tìm kiếm gần đúng (Approximate Nearest Neighbor Search), một trong những tính năng quan trọng của vector database là khả năng thực hiện tìm kiếm gần đúng (approximate nearest neighbor search) các vector với độ chính xác và hiệu suất cao [22, 23]. Điều này rất cần thiết trong việc tìm kiếm và lọc dữ liệu liên quan trong các hệ thống chatbot.

Theo Johnson et al. [23], vector database sử dụng các kỹ thuật lập chỉ mục tiên tiến như inverted index, hashing hoặc tree-based index để tổ chức các vector. Nhờ đó, chúng có thể thực hiện truy vấn tìm kiếm gần đúng một cách hiệu quả, thay vì phải so sánh từng vector một như trong cách tiếp cận truyền thống.

Cụ thể, Hajebi et al. [22] đề xuất một thuật toán tìm kiếm gần đúng dựa trên k-nearest neighbor graph. Thuật toán này xây dựng một đồ thị k-nearest neighbor, trong đó mỗi vector được liên kết với k vector gần nhất. Khi thực hiện truy vấn, thuật toán sẽ khởi động từ một vài vector láng giềng và di chuyển dọc theo đồ thị để tìm các vector gần nhất.



Hình 4-Minh họa thuật toán tìm kiếm gần đúng dựa trên k-nearest neighbor graph

Ngoài tìm kiếm gần đúng, các vector database cũng hỗ trợ khả năng xếp hạng kết quả truy vấn dựa trên khoảng cách vector. Ví dụ, có thể xếp các kết quả theo độ tương tự ngữ nghĩa (đo bằng cosine similarity) hoặc khoảng cách Euclidean giữa vector truy vấn và các vector trong database đã được trình bày ở phía trên.

Việc xếp hạng dựa trên các phép đo khoảng cách vector như cosine similarity, Euclidean distance hay Manhattan distance giúp vector database có thể trả về các kết quả tìm kiếm theo thứ tự relevance, phù hợp với nhu cầu của người dùng. Các phép đo này cung cấp thông tin về mức độ tương tự ngữ nghĩa giữa các vector, từ đó hệ thống có thể lọc và sắp xếp các kết quả tìm kiếm tốt hơn.

Kết hợp tìm kiếm gần đúng và xếp hạng dựa trên khoảng cách vector giúp vector database có thể xử lý hiệu quả các truy vấn tìm kiếm dữ liệu liên quan trong bối cảnh của các hệ thống chatbot quy mô lớn [5].

2.2.3 Ưu nhược điểm so với phương pháp truyền thống.

Ưu điểm của vector embedding và vector database so với các phương pháp lưu trữ và tìm kiếm truyền thống:

- Theo Johnson et al. [23], vector database có thể thực hiện tìm kiếm gần đúng (approximate nearest neighbor search) các vector với độ chính xác và tốc độ cao, nhờ sử dụng các kỹ thuật lập chỉ mục tiên tiến như inverted index, hashing hoặc tree-based index giúp xử lý hiệu quả các

truy vấn tìm kiếm dữ liệu liên quan, đặc biệt là các hệ thống chatbot có quy mô lớn.

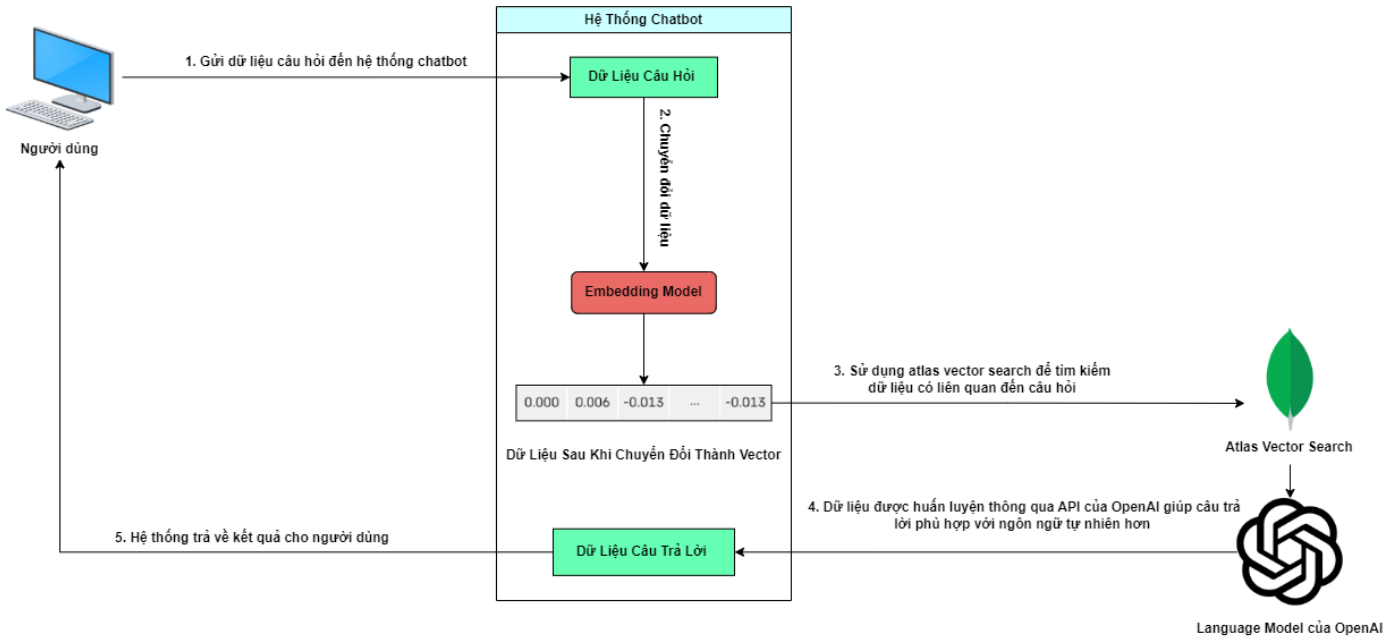
- Theo Cattell [9] chỉ ra rằng khi lượng dữ liệu tăng lên, các database truyền thống như SQL hay NoSQL sẽ gặp phải những hạn chế về hiệu suất. Ngược lại, vector database được thiết kế để có thể mở rộng quy mô lưu trữ và xử lý dữ liệu lớn một cách hiệu quả hơn.
- Vector database được xây dựng để lưu trữ và truy vấn dữ liệu dạng vector, như các vector biểu diễn văn bản, hình ảnh hay âm thanh giúp các hệ thống chatbot có thể xử lý được các dạng dữ liệu đa dạng hơn ngoài văn bản [5, 23].

Nhược điểm của vector embedding và vector database so với các phương pháp lưu trữ và tìm kiếm truyền thống:

- Yêu cầu phức tạp hơn về triển khai Việc triển khai và vận hành vector database có độ phức tạp cao hơn so với các database truyền thống. Nó yêu cầu các kỹ năng chuyên sâu về lập chỉ mục, tối ưu hóa truy vấn và quản lý hạ tầng [9].
- Chi phí lưu trữ và tính toán Lưu trữ và xử lý dữ liệu dạng vector có thể đòi hỏi nhiều tài nguyên hơn về mặt lưu trữ và tính toán, đặc biệt là khi quy mô dữ liệu lớn. Điều này dẫn đến chi phí cao hơn so với các giải pháp truyền thống [19].

3. Sử dụng vector embedding và vector database vào hệ thống chatbot.

3.1 Tổng quan kiến trúc hệ thống .



Hình 5 Tổng quan kiến trúc hệ thống

Hệ thống chatbot được thiết kế dựa trên việc kết hợp các công nghệ vector embedding và vector database nhằm cung cấp khả năng tìm kiếm và trả lời câu hỏi hiệu quả cho người dùng. Kiến trúc tổng thể của hệ thống bao gồm các thành phần chính sau:

Giao diện người dùng (User Interface): Đây là nơi tiếp nhận yêu cầu từ người dùng, như nhập câu hỏi hoặc nội dung trao đổi và gửi nội dung câu hỏi về hệ thống chatbot để xử lý.

Embedding Model: Sau khi hệ thống nhận được dữ liệu câu hỏi của người dùng thì thành phần này chịu trách nhiệm biểu diễn dữ liệu đầu vào (văn bản, hình ảnh, âm thanh) dưới dạng các vector số hay còn gọi là vector embedding. Việc này giúp chuyển đổi dữ liệu phi cấu trúc thành các vector giúp việc tìm kiếm dữ liệu liên quan đến câu hỏi bằng atlas vector search trở nên đơn giản hơn.

MongoDB atlas vector search: Một tính năng trong dịch vụ MongoDB Atlas, cho phép lưu trữ và tìm kiếm dữ liệu dạng vector như embeddings văn bản, hình ảnh, âm thanh. Nó sử dụng công nghệ tìm kiếm dựa trên vector thay vì tìm kiếm dựa trên từ khóa, giúp tìm kiếm dựa trên ý nghĩa và nội dung. Vector Search tích hợp sâu với MongoDB, cho phép lưu trữ và truy vấn dữ liệu vector cùng với dữ liệu cấu trúc khác. Tính năng này có hiệu suất cao và khả năng mở rộng linh hoạt.

Language Model của OpenAI: Sau khi xác định được các vector liên quan từ bước trước, thông tin này sẽ được sử dụng để lấy câu trả lời từ mô hình ngôn ngữ lớn của OpenAI. Mô hình này có khả năng hiểu ngữ cảnh và trả lời câu hỏi một cách tự nhiên.

3.2 Công nghệ sử dụng.

ReactJS- Thư viện JavaScript phổ biến để xây dựng giao diện người dùng:

- Cung cấp kiến trúc component-based, giúp xây dựng giao diện linh hoạt, tái sử dụng và dễ bảo trì.
- Sử dụng cơ chế Virtual DOM hiệu quả, giúp tối ưu hiệu suất ứng dụng web.
- Được hỗ trợ bởi một cộng đồng lớn và nhiều plugin/thư viện bổ sung.
- Có thể dễ dàng tích hợp với các công nghệ backend như Node.js để xây dựng ứng dụng full-stack.
- Phù hợp để xây dựng giao diện người dùng phong phú, tương tác tốt cho hệ thống chatbot.

Node.js - Nền tảng cho phía máy chủ:

- Cho phép sử dụng JavaScript từ backend, tạo ra một ngôn ngữ thống nhất từ frontend đến backend.
- Có thể xử lý các yêu cầu HTTP, quản lý kết nối, xử lý dữ liệu một cách hiệu quả.
- Hệ sinh thái NPM cung cấp hàng nghìn gói thư viện và công cụ hỗ trợ phát triển nhanh.
- Thích hợp để xây dựng RESTful API, xử lý logic nghiệp vụ, tích hợp với các dịch vụ bên ngoài cho hệ thống chatbot.

MongoDB Atlas - Dịch vụ cơ sở dữ liệu NoSQL trên đám mây:

- Cung cấp tính năng Vector Search giúp lưu trữ và tìm kiếm hiệu quả các vector biểu diễn văn bản, hình ảnh, âm thanh.
- Cho phép tìm kiếm dựa trên tương đồng vector, giúp tìm kiếm dữ liệu liên quan dựa trên ngữ nghĩa.
- Được quản lý và vận hành hoàn toàn trên đám mây, giảm chi phí quản trị cơ sở hạ tầng.
- Có khả năng mở rộng linh hoạt, phù hợp để xử lý các khối lượng dữ liệu lớn của hệ thống chatbot.

OpenAI - Mô hình ngôn ngữ lớn (Large Language Model):

- Cung cấp các mô hình ngôn ngữ tiên tiến như GPT-3, Davinci, có khả năng hiểu và tạo ra văn bản tự nhiên.
- Có thể tích hợp mô hình này vào hệ thống chatbot để cung cấp câu trả lời thông minh, linh hoạt dựa trên hiểu biết sâu rộng.
- Được đào tạo trên lượng dữ liệu văn bản khổng lồ, giúp mô hình có khả năng xử lý ngôn ngữ tự nhiên tiên tiến.
- Cung cấp API dễ dàng tích hợp vào hệ thống, cho phép tận dụng sức mạnh của mô hình ngôn ngữ lớn.

3.3 Chi tiết triển khai.

4. Thực nghiệm và Đánh giá .

4.1 Thiết lập thực nghiệm .

4.1.1 Dữ liệu .

4.1.2 Cài đặt môi trường và công cụ.

4.1.3 Phương pháp và tiêu chí đánh giá.

4.2 Kết quả.

4.2.1 Hiệu suất.

4.2.2 Độ chính xác và liên quan.

4.2.3 So sánh với các phương pháp khác.

4.3 Phân tích và Thảo luận

5. Kết luận

5.1 Tóm tắt đóng góp

5.2 Hạn chế và Hướng nghiên cứu tiếp theo

[1] OpenAI, "OpenAI API," ed, 2020, June 5.

[2] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big??," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610-623.

[3] B. A. Shawar and E. Atwell, "Chatbots: are they really useful?," *Journal for Language Technology and Computational Linguistics,* vol. 22, no. 1, pp. 29-49, 2007.

[4] B. Dhingra *et al.*, "Towards end-to-end reinforcement learning of dialogue agents for information access," *arXiv preprint arXiv:1609.00777,* 2016.

[5] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906,* 2020.
[6] H. Fang, T. Tao, and C. Zhai, "A formal study of information retrieval heuristics," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 49-56.
[7] A. Patrizio, "IDC: Expect 175 Zettabytes of Data Worldwide by 2025," ed, 2018.
[8] S. Zhang, Zhang, C., & Zhang, W., "A survey on deep learning based chatbot models. In International Conference on Intelligent Computing, Automation and Systems," 2018, April.
[9] R. Cattell, "Scalable SQL and NoSQL data stores," *Acm Sigmod Record,* vol. 39, no. 4, pp. 12-27, 2011.
[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management,* vol. 24, no. 5, pp. 513-523, 1988.
[11] J. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.
[12] J. L. Ambite and D. Kapoor, "Automatically Composing Data Workflows with Relational Descriptions and Shim Services," Berlin, Heidelberg, 2007: Springer Berlin Heidelberg, in The Semantic Web, pp. 15-29.
[13] P. Hitzler, M. Krotzsch, and S. Rudolph, *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009.
[14] A. Jankowski, A. Skowron, and M. Szczuka, "Interactive Granular Computing in Rightly Judging Systems," Berlin, Heidelberg, 2009: Springer Berlin Heidelberg, in Rough Sets and Knowledge Technology, pp. 1-16.
[15] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *ACM Sigir Forum*, 2017, vol. 51, no. 2: ACM New York, NY, USA, pp. 268-276.
[16] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Cornell University, 1987.
[17] S. E. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne, "Okapi at TREC-4," *Nist Special Publication Sp,* pp. 73-96, 1996.
[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems,* vol. 26, 2013.
[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532-1543.
[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018.
[21] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City,* vol. 1, no. 2, p. 1, 2007.
[22] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
[23] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data,* vol. 7, no. 3, pp. 535-547, 2019.