Embedding

Harito ID

2025-10-09

Toàn bộ về Embedding trong Xử lý ngôn ngữ tự nhiên

Chương 1: Khởi nguồn và Mục đích của Biểu diễn Từ

1.1. Từ thách thức của ngôn ngữ đến bài toán Biểu diễn số

Trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), một trong những thách thức nền tảng và cốt lõi nhất là làm thế nào để biểu diễn ngôn ngữ của con người dưới một định dạng mà các thuật toán học máy và mô hình có thể hiểu và xử lý. Bản chất của văn bản thô, bao gồm các ký tự và từ, là phi cấu trúc và không thể trực tiếp sử dụng làm đầu vào cho các mô hình toán học.[1] Các thuật toán học máy được xây dựng để làm việc với dữ liệu số, vì vậy, việc chuyển đổi văn bản sang các vector số là một bước tiền xử lý không thể thiếu. **Embedding**, hay còn gọi là *phép nhúng*, ra đời chính là để giải quyết bài toán này. Nó đóng vai trò như một cầu nối, cho phép máy móc không chỉ "đọc" mà còn "hiểu" được ngữ nghĩa, ngữ pháp và mối quan hệ phức tạp giữa các từ và câu trong văn bản.[1]

Sự ra đời của Embedding đã mở ra cánh cửa cho hàng loạt ứng dụng NLP tiên tiến. Từ những nhiệm vụ cơ bản như phân loại văn bản (phân tích cảm xúc, phát hiện thư rác, phân loại chủ đề) đến các bài toán phức tạp hơn như nhận dạng thực thể có tên (NER), dịch máy, hệ thống truy vấn thông tin, hay hệ thống hỏi đáp.[1] Ví dụ, trong hệ thống hỏi đáp, Embedding giúp mô hình hiểu được ngữ cảnh của câu hỏi và tìm ra câu trả lời phù hợp dựa trên sự tương đồng ngữ nghĩa.[2] Nó cũng là nền tảng cho các hệ thống gợi ý, nơi các tài liệu có nội dung tương đồng được nhóm lại với nhau hoặc các sản phẩm tương tự được đề xuất dựa trên mô tả văn bản của chúng.[1]

Một lợi thế nổi bật của Embedding so với các phương pháp biểu diễn truyền thống là khả năng giảm chiều dữ liệu. Thay vì sử dụng các vector có chiều cực lớn và thưa thớt (chứa nhiều số 0), Embedding tạo ra các **vector dày đặc** (dense vectors) trong một không gian có chiều thấp hơn đáng kể.[2] Điều này không chỉ giúp giảm gánh nặng tính toán và yêu cầu bộ nhớ, làm cho các ứng dụng NLP quy mô lớn trở nên khả thi hơn, mà còn cho phép các mô hình học máy hoạt động hiệu quả hơn bằng cách cung cấp một biểu diễn ngôn ngữ cô đọng và giàu thông tin hơn.[2]

1.2. Biểu diễn truyền thống: One-Hot Encoding

Trước khi các kỹ thuật Embedding hiện đại xuất hiện, phương pháp phổ biến để biểu diễn từ là **One-Hot Encoding**. Đây là một phương pháp đơn giản và trực quan, hoạt động bằng cách gán cho mỗi từ trong từ điển một vector nhị phân duy nhất.[4] Chiều dài của vector này bằng với tổng số từ duy nhất (kích thước từ điển) trong toàn bộ kho văn bản (corpus).[5] Vị trí tương ứng với từ đó sẽ có giá trị là 1, và tất cả các vị trí còn

lại đều là 0.[5] Ví dụ, nếu từ điển chỉ có ba từ là “mèo”, “chó”, và “cá”, thì “mèo” có thể được biểu diễn là `[1, 0, 0]`, “chó” là `[0, 1, 0]`, và “cá” là `[0, 0, 1]`.

Mặc dù One-Hot Encoding có ưu điểm là dễ hiểu và dễ triển khai, nhưng nó bộc lộ những nhược điểm nghiêm trọng, đặt ra nhu cầu cấp thiết cho các phương pháp biểu diễn tinh vi hơn. Hạn chế lớn nhất là “lời nguyền chiều dữ liệu” (*curse of dimensionality*).[4] Khi kích thước từ điển tăng lên, chiều của vector One-Hot cũng tăng theo, dẫn đến việc phải sử dụng rất nhiều biến giả để mã hóa dữ liệu. Một từ điển gồm 30.000 từ sẽ tạo ra một vector có 30.000 chiều.[3] Điều này không chỉ gây tốn kém bộ nhớ và làm chậm đáng kể quá trình huấn luyện mô hình, mà còn dẫn đến hiện tượng dữ liệu thưa thớt (*sparsity*) vì phần lớn các giá trị trong vector đều là 0.[4]

Tuy nhiên, nhược điểm mang tính quyết định của One-Hot Encoding là việc nó hoàn toàn không thể nắm bắt được mối quan hệ ngữ nghĩa giữa các từ.[5] Trong không gian vector One-Hot, mọi từ đều có khoảng cách trực giao như nhau. Điều này có nghĩa là “vua” và “nữ hoàng” có khoảng cách không khác gì so với “vua” và “máy tính”. Mô hình không thể nhận biết được rằng “vua” và “nữ hoàng” có mối quan hệ gần gũi hơn về mặt ngữ nghĩa so với “vua” và “máy tính”.[5] Hơn nữa, việc cố gắng mã hóa từng từ hiếm cũng có thể dẫn đến hiện tượng *overfitting*, khi mô hình học quá kỹ các trường hợp cá biệt và không thể khái quát hóa.[7] Những hạn chế này chính là động lực mạnh mẽ thúc đẩy sự ra đời của các phương pháp Embedding hiện đại, nơi mục tiêu không chỉ là chuyển từ thành số, mà còn là bảo toàn và thể hiện ý nghĩa của chúng.

1.3. Nền tảng lý thuyết: Giả thuyết Phân bố (Distributional Hypothesis)

Sự chuyển dịch từ các phương pháp biểu diễn truyền thống như One-Hot Encoding sang Embedding dày đặc không chỉ là một thay đổi về kỹ thuật, mà còn là một bước ngoặt về tư duy lý thuyết. Nền tảng triết học cho sự thay đổi này là **“Giả thuyết Phân bố”** (*Distributional Hypothesis*), một khái niệm nền tảng trong ngữ nghĩa học ngôn ngữ.[2] Giả thuyết này khẳng định rằng:

> “Những từ xuất hiện trong cùng ngữ cảnh có xu hướng mang ý nghĩa tương đồng”.[2]

Nói một cách đơn giản, chúng ta có thể suy luận ý nghĩa của một từ bằng cách quan sát các từ xung quanh nó. Ví dụ, trong câu “Tôi đi câu cá ở bờ sông”, từ “câu” mang ý nghĩa hoạt động giải trí; trong khi đó, trong câu “Anh ấy kể một câu chuyện rất hay”, từ “câu” lại mang ý nghĩa khác. Nếu “cà phê” và “trà” thường xuất hiện trong các ngữ cảnh tương tự như “uống…”, “bữa sáng…”, thì chúng có khả năng mang ý nghĩa gần gũi nhau.

Giả thuyết này đã cung cấp một cơ sở lý thuyết vững chắc cho việc học các vector từ dày đặc (*dense vectors*).[2] Thay vì coi mỗi từ là một thực thể độc lập như trong One-Hot Encoding, các phương pháp Embedding hiện đại học cách biểu diễn từ dựa trên ngữ cảnh của chúng. Bằng cách phân tích các mô hình đồng xuất hiện của từ trong một kho văn bản khổng lồ, các thuật toán có thể tạo ra các vector số, nơi các từ có ý nghĩa tương tự sẽ nằm gần nhau trong không gian vector đa chiều.[10] Điều này tạo ra một biểu diễn liên tục và có ngữ nghĩa, cho phép các mô hình không chỉ nhận biết từ mà còn hiểu được mối quan hệ phức tạp giữa chúng.

Một chuỗi tư duy phát triển có thể được mô tả như sau: nhận thấy vấn đề của One-Hot Encoding là không thể hiện ngữ nghĩa, giả thuyết Phân bố được đề xuất như một lý thuyết giải pháp. Giả thuyết này sau đó được **Word2Vec**, một mô hình dự đoán, hiện thực hóa và chứng minh tính hiệu quả.[9] Từ đó, các phương pháp Embedding tiếp theo

như **GloVe** đã kết hợp giả thuyết này với phương pháp thống kê toàn cục để tạo ra một cách tiếp cận mới, tạo thành một câu chuyện tiến hóa mạch lạc trong lịch sử NLP.

Chương 2: Cuộc cách mạng đầu tiên: Embedding Từ tĩnh (Static Word Embeddings)

2.1. Phương pháp dựa trên tần suất: TF-IDF

Trước khi các mô hình học máy dựa trên dự đoán ra đời, các phương pháp thống kê dựa trên tần suất đã có những đóng góp quan trọng trong việc cải thiện chất lượng biểu diễn văn bản. Một trong những kỹ thuật nổi bật nhất là **TF-IDF** (*Term Frequency-Inverse Document Frequency*).[12] TF-IDF là một thống kê số học phản ánh mức độ quan trọng của một từ trong một tài liệu so với một tập hợp các tài liệu lớn hơn.[12] Nó giải quyết một vấn đề cốt lõi của các phương pháp đếm tần suất đơn thuần: những từ phổ biến như “the”, “a”, “and” thường xuất hiện rất nhiều nhưng lại ít mang ý nghĩa đặc trưng.[12]

Cơ chế của TF-IDF được xây dựng trên hai thành phần chính: - **Tần suất từ (Term Frequency - TF):** Đo lường mức độ thường xuyên một từ xuất hiện trong một tài liệu cụ thể.[12] Một giá trị TF cao cho thấy từ đó có ý nghĩa quan trọng trong tài liệu đó.[13] - **Tần suất nghịch tài liệu (Inverse Document Frequency - IDF):** Đo lường mức độ phổ biến hay hiếm của một từ trên toàn bộ kho văn bản.[12] Nó được tính bằng lôgarit của tỷ lệ giữa tổng số tài liệu và số tài liệu chứa từ đó. Một giá trị IDF cao cho thấy từ đó rất hiếm, và do đó, có ý nghĩa đặc trưng hơn.[12]

Bằng cách nhân hai giá trị này, TF-IDF tạo ra một điểm số làm nổi bật những từ vừa xuất hiện thường xuyên trong một tài liệu, vừa hiếm khi xuất hiện ở các tài liệu khác.[13] Ví dụ, từ “espresso” có thể có điểm TF-IDF cao trong một tài liệu về cà phê, trong khi các từ như “the” hoặc “and” sẽ có điểm số thấp.[1] TF-IDF được ứng dụng rộng rãi trong tìm kiếm, phân loại văn bản, và trích xuất từ khóa.[13] Tuy nhiên, mặc dù là một cải tiến đáng kể so với việc đếm tần suất đơn thuần, TF-IDF vẫn chỉ là một phương pháp thống kê và không thể nắm bắt được mối quan hệ ngữ nghĩa sâu sắc giữa các từ. Nó vẫn thuộc trường phái “dựa trên tần suất” (*frequency-based*) và không thể hiểu được rằng “vua” và “nữ hoàng” có ý nghĩa gần gũi.

2.2. Phương pháp dựa trên dự đoán: Word2Vec

Word2Vec, được phát triển bởi một nhóm nghiên cứu tại Google, đã đánh dấu một cuộc cách mạng trong lĩnh vực biểu diễn từ bằng cách giới thiệu một phương pháp mới dựa trên dự đoán thay vì thống kê tần suất.[14] Ý tưởng cốt lõi của Word2Vec là học các vector từ bằng cách dự đoán mối quan hệ giữa các từ và ngữ cảnh của chúng.[9] Mô hình này xây dựng các vector dày đặc (*dense vectors*), nơi các từ có ý nghĩa tương tự sẽ có các vector gần nhau trong không gian số.[3] Word2Vec có hai kiến trúc chính:

- **CBOW (Continuous Bag of Words):** Mô hình này hoạt động bằng cách lấy một tập hợp các từ ngữ cảnh xung quanh một từ đích và cố gắng dự đoán từ đích đó.[11] Kiến trúc mạng nơ-ron của CBOW khá đơn giản, bao gồm ba lớp: lớp đầu vào là các từ ngữ cảnh được mã hóa One-Hot, một lớp ẩn (projection layer) tạo ra vector trung bình của các từ đầu vào, và một lớp đầu ra sử dụng hàm Softmax để dự đoán xác suất của từ đích.[16] Ví dụ, trong câu “The cat sat on the mat” và từ đích là “sat”, CBOW sẽ sử dụng các từ “The”, “cat”, “on”, “the” để dự đoán “sat”.[11] CBOW được biết đến với tốc độ huấn luyện nhanh hơn và hiệu quả hơn trên các tập dữ liệu lớn.[3]

- **Skip-gram:** Ngược lại với CBOW, Skip-gram hoạt động bằng cách lấy một từ đích và cố gắng dự đoán các từ ngữ cảnh xung quanh nó trong một cửa sổ cố định.[11] Với cùng ví dụ trên và từ đích là "sat", Skip-gram sẽ cố gắng dự đoán các từ "The", "cat", "on", "the".[11] Mặc dù Skip-gram huấn luyện chậm hơn và tốn tài nguyên hơn, nó lại vượt trội trong việc nắm bắt các mối quan hệ ngữ nghĩa sâu sắc và đặc biệt hiệu quả với các từ hiếm.[3]

Mặc dù Word2Vec đã tạo ra những vector từ có khả năng biểu diễn ngữ nghĩa vượt trội, nó vẫn có một hạn chế lớn. Word2Vec là một mô hình **"tĩnh"** (*static*), nghĩa là mỗi từ trong từ điển chỉ có một vector duy nhất, bất kể ngữ cảnh nó xuất hiện.[17] Điều này gây ra vấn đề với các từ đa nghĩa (*polysemy*).[18] Ví dụ, từ "bank" sẽ có cùng một vector trong cả hai câu "I went to the river bank" (bờ sông) và "I went to the national bank" (ngân hàng), khiến mô hình không thể phân biệt được các ý nghĩa khác nhau của từ này. Hạn chế này đã mở đường cho sự ra đời của các mô hình Embedding từ ngữ cảnh tiếp theo.

2.3. Cải tiến từ thống kê toàn cục: GloVe

Trong khi Word2Vec tận dụng thông tin ngữ cảnh "cục bộ" (*local context*) bằng cách huấn luyện dựa trên các cặp từ trong một cửa sổ trượt, **GloVe** (*Global Vectors for Word Representation*) đã ra đời như một phương pháp lai, kết hợp cả lợi thế của phương pháp dự đoán với thông tin thống kê "toàn cục" (*global statistics*) của toàn bộ kho văn bản.[19]

Thay vì sử dụng mạng nơ-ron, GloVe xây dựng một **ma trận đồng xuất hiện** (*co-occurrence matrix*) cho toàn bộ corpus.[19] Ma trận này ghi lại tần suất hai từ xuất hiện cùng nhau trong một ngữ cảnh xác định.[19] Sau đó, GloVe tối ưu hóa một hàm mất mát đặc biệt để học các vector từ sao cho tích vô hướng (*dot product*) của chúng xấp xỉ với logarit của xác suất đồng xuất hiện.[19] Điều này đảm bảo rằng GloVe không chỉ học được mối quan hệ ngữ cảnh gần như Word2Vec, mà còn tận dụng được thông tin toàn diện về tần suất và cấu trúc của toàn bộ ngôn ngữ.[20]

Ưu điểm nổi bật của GloVe là khả năng tận dụng thông tin toàn cục.[19] Trong khi Word2Vec chỉ xem xét các cặp từ lân cận, GloVe phân tích toàn bộ ma trận đồng xuất hiện của corpus, cho phép nó nắm bắt các mối quan hệ ngữ nghĩa tinh tế và toàn diện hơn.[19] Mặc dù trong thực tế, hiệu suất của GloVe và Word2Vec có thể tương đương nhau trên nhiều bài toán [20], GloVe được đánh giá là có cơ sở lý thuyết mạnh mẽ hơn và dễ dàng song song hóa để huấn luyện trên các tập dữ liệu cực lớn.[23] Cả hai mô hình này đều thuộc nhóm Embedding từ tĩnh, không thể xử lý vấn đề đa nghĩa, nhưng sự ra đời của GloVe đã khẳng định rằng việc kết hợp thông tin thống kê toàn cục có thể tạo ra các vector biểu diễn mạnh mẽ.

2.4. Khắc phục từ chưa từng thấy: FastText

Một trong những hạn chế cố hữu của các mô hình Embedding tĩnh như Word2Vec và GloVe là chúng không thể tạo ra vector cho các từ chưa từng xuất hiện trong bộ dữ liệu huấn luyện (*out-of-vocabulary - OOV*).[15] Điều này là một vấn đề lớn, đặc biệt đối với các ngôn ngữ có cấu trúc từ phong phú, các thuật ngữ chuyên ngành, hoặc khi mô hình phải xử lý các lỗi chính tả.

Để giải quyết vấn đề này, Facebook đã phát triển **FastText**, một mô hình mở rộng của Word2Vec.[15] Thay vì coi mỗi từ là một thực thể nguyên tử, FastText biểu diễn mỗi từ như là một "túi" của các **n-gram ký tự**.[18] Vector cuối cùng của một từ là tổng hợp của các vector của tất cả các n-gram ký tự con của nó.[15] Ví dụ, từ "running" có thể được biểu diễn bởi các n-gram ký tự như `<ru`, `run`, `unn`, `nni`, `nin`, `ing>`, và `nning>`.[24]

Ưu điểm của FastText là vượt trội. Bằng cách sử dụng các n-gram ký tự, FastText có thể tạo ra một vector có ý nghĩa cho một từ ngay cả khi nó chưa từng được nhìn thấy trước đây. Lý do là các từ mới thường được cấu thành từ các n-gram ký tự quen thuộc mà mô hình đã học được từ các từ khác.[18] Điều này làm cho FastText đặc biệt hữu ích cho các ngôn ngữ có cấu trúc phức tạp như tiếng Việt hoặc các ngôn ngữ có nhiều từ ghép. FastText cũng hoạt động tốt với các từ hiếm và giúp giải quyết vấn đề sai chính tả ở một mức độ nhất định.[18]

Chương 3: Bước ngoặt của Ngữ cảnh: Embedding từ Ngữ cảnh (Contextual Word Embeddings)

3.1. Tại sao cần Contextual Embeddings?

Mặc dù các mô hình Embedding từ tĩnh như Word2Vec, GloVe và FastText đã tạo ra một cuộc cách mạng trong NLP, chúng vẫn tồn tại một hạn chế cốt lõi: mỗi từ chỉ được gán một vector duy nhất, không phụ thuộc vào ngữ cảnh của nó.[18] Hạn chế này không chỉ giới hạn khả năng của mô hình trong việc xử lý các từ đa nghĩa (*polysemy*) mà còn bỏ qua những sắc thái tinh tế của ngôn ngữ. Ví dụ, từ “apple” có thể là một loại trái cây, một tên công ty công nghệ, hoặc một biệt danh. Một mô hình tĩnh sẽ biểu diễn cả ba ý nghĩa này bằng cùng một vector, dẫn đến sự nhầm lẫn và giảm hiệu suất trong các bài toán phức tạp.[17] Sự thiếu hụt này chính là động lực thúc đẩy sự ra đời của các mô hình **Embedding từ ngữ cảnh**. Mục tiêu là tạo ra một hệ thống, nơi vector biểu diễn của một từ là một hàm của toàn bộ câu mà nó xuất hiện, cho phép cùng một từ có các vector khác nhau tùy thuộc vào ngữ cảnh cụ thể của nó.[17]

3.2. ELMo: Từ Bi-directional LSTM

ELMo (*Embeddings from Language Models*) được coi là một trong những mô hình tiên phong giải quyết vấn đề đa nghĩa của từ bằng cách đưa ngữ cảnh vào biểu diễn của chúng.[17] ELMo không tạo ra một vector tĩnh cho mỗi từ. Thay vào đó, nó tạo ra các vector từ ngữ cảnh bằng cách sử dụng một mạng **LSTM hai chiều** (*bi-directional LSTM*).[17]

Cơ chế hoạt động của ELMo dựa trên một mô hình ngôn ngữ hai chiều (*biLM*) với nhiều lớp. Mỗi lớp trong mô hình này học một biểu diễn khác nhau của từ. Mô hình LSTM hai chiều bao gồm một luồng tiến (*forward pass*) đọc câu từ trái sang phải và một luồng lùi (*backward pass*) đọc câu từ phải sang trái. Luồng tiến nắm bắt ngữ cảnh của từ trước nó, trong khi luồng lùi nắm bắt ngữ cảnh của từ sau nó.[17] Vector cuối cùng của một từ trong ELMo là tổng hợp có trọng số của các vector từ các lớp khác nhau, từ lớp đầu vào (biểu diễn từ thô) cho đến các lớp ẩn của mô hình LSTM hai chiều.[17] Điều này cho phép ELMo tạo ra các vector từ khác nhau cho cùng một từ trong các ngữ cảnh khác nhau, giải quyết thành công vấn đề đa nghĩa mà các mô hình tĩnh không thể làm được.[17] Mặc dù vậy, ELMo vẫn có một điểm hạn chế: nó học ngữ cảnh tiến và lùi một cách độc lập và sau đó chỉ nối chúng lại với nhau (*concatenate*), không phải là một mô hình “hai chiều sâu” (*deeply bi-directional*) thực sự.[25]

3.3. BERT và kiến trúc Transformer

Sự ra đời của **BERT** (*Bidirectional Encoder Representations from Transformers*) do Google AI phát triển đã đánh dấu một bước đột phá lớn, giải quyết triệt để hạn chế của các mô hình trước đó.[26] BERT không sử dụng mạng LSTM như ELMo, mà dựa hoàn toàn vào kiến trúc **Transformer Encoder**.[25] Cơ chế Attention trong Transformer cho

phép BERT xử lý tất cả các từ trong câu cùng một lúc, học được mối quan hệ hai chiều sâu sắc giữa chúng một cách đồng thời. Điều này khác biệt hoàn toàn với ELMo, vốn chỉ nối ngữ cảnh tiến và lùi lại với nhau.[25]

BERT được huấn luyện tiền xử lý (*pre-training*) trên một kho văn bản khổng lồ không gắn nhãn (Wikipedia, sách) với hai nhiệm vụ chính: - **Masked Language Model (MLM):** Mô hình sẽ che đi ngẫu nhiên 15% số từ trong câu (thay bằng token `[MASK]`) và cố gắng dự đoán những từ đã bị che đó dựa trên toàn bộ ngữ cảnh xung quanh.[26] Nhiệm vụ này buộc mô hình phải học một biểu diễn ngữ cảnh hai chiều sâu sắc cho mỗi từ.[28] - **Next Sentence Prediction (NSP):** Nhiệm vụ này yêu cầu mô hình dự đoán liệu một câu thứ hai có phải là câu tiếp theo của câu thứ nhất hay không.[27] Điều này giúp mô hình hiểu được mối quan hệ ngữ nghĩa giữa các câu, một khả năng cần thiết cho các bài toán như hỏi đáp.[26]

Sự kết hợp của kiến trúc Transformer và hai nhiệm vụ tiền xử lý trên đã tạo ra một mô hình ngôn ngữ mạnh mẽ, có khả năng nắm bắt được mối quan hệ logic giữa các từ và ý tưởng trong câu và đoạn văn một cách chính xác hơn so với các mô hình NLP trước đây.[27] BERT đã trở thành một nền tảng cơ bản, có thể được tinh chỉnh (*fine-tuned*) dễ dàng để giải quyết nhiều bài toán NLP khác nhau với hiệu suất vượt trội.[18]

Chương 4: Vươn xa hơn từ: Embedding câu và Đa phương thức

4.1. Từ Word Embedding đến Sentence Embedding

Sự phát triển của Embedding từ ngữ cảnh đã làm nổi bật một nhu cầu tiếp theo: biểu diễn toàn bộ một câu hoặc một đoạn văn dưới dạng một vector duy nhất.[29] **Sentence Embedding** (*Embedding câu*) ra đời với mục tiêu này, nhằm mã hóa toàn bộ ngữ nghĩa và sắc thái của một câu thành một vector số duy nhất, cho phép so sánh ngữ nghĩa giữa các câu một cách hiệu quả.[30]

Những phương pháp đơn giản ban đầu để tạo Sentence Embedding là tổng hợp các Word Embedding.[29] Cách tiếp cận trực tiếp nhất là lấy trung bình cộng các vector từ của tất cả các từ trong câu (*average-pooling*).[29] Mặc dù phương pháp này đơn giản và hiệu quả về mặt tính toán, nó có một hạn chế nghiêm trọng: nó bỏ qua trật tự và ngữ pháp của từ.[32] Một số phương pháp khác sử dụng các mạng nơ-ron hồi quy như LSTM hoặc GRU, tận dụng trạng thái ẩn cuối cùng của mạng làm vector đại diện cho cả câu.[29] Các mô hình này đã có một bước tiến lớn trong việc nắm bắt trình tự của từ, nhưng vẫn còn những hạn chế về khả năng xử lý song song và hiệu suất.

4.2. Sentence-BERT (SBERT): Giải pháp cho bài toán so sánh câu

Một vấn đề bất ngờ đã nảy sinh khi các nhà nghiên cứu cố gắng sử dụng các mô hình mạnh mẽ như BERT để tạo Sentence Embedding. Mặc dù BERT rất hiệu quả trong các nhiệm vụ phân loại câu (*sentence classification*) [28], việc sử dụng trực tiếp vector của token `[CLS]` (token đặc biệt được thêm vào đầu mỗi câu để tóm tắt ngữ nghĩa) để so sánh hai câu lại cho kết quả rất kém.[29] Hiệu suất này thậm chí còn tệ hơn so với việc đơn giản là lấy trung bình các vector từ không ngữ cảnh.[29] Nguyên nhân là vì BERT không được tối ưu hóa trong quá trình huấn luyện tiền xử lý để tạo ra một vector câu có ý nghĩa, phù hợp để so sánh bằng *cosine similarity*. Các chiều dữ liệu đầu ra của nó không được "chuẩn hóa" để có ý nghĩa ngang nhau cho các bài toán không có nhãn.[33]

Sentence-BERT (SBERT) đã ra đời như một giải pháp chuyên biệt, giải quyết trực tiếp vấn đề này.[29] SBERT là một phiên bản tinh chỉnh của BERT, sử dụng kiến trúc **mạng Siamese**.[34] Mạng Siamese bao gồm hai mô hình BERT song sinh, cùng trọng

số, xử lý hai câu đầu vào độc lập.[34] SBERT được huấn luyện trên các tập dữ liệu so sánh câu bằng cách sử dụng các hàm mục tiêu đặc biệt (như classification, regression, hoặc triplet loss) để học cách tạo ra các vector câu có ngữ nghĩa tương đồng cao.[34] Các hàm mục tiêu này ép buộc mô hình phải tạo ra các vector sao cho khoảng cách giữa các cặp câu tương đồng nhỏ hơn nhiều so với các cặp câu không tương đồng.[34] Nhờ đó, SBERT đạt được hiệu suất vượt trội trên các nhiệm vụ so sánh câu, tìm kiếm ngữ nghĩa (*semantic search*), và phân cụm văn bản.[29] SBERT đã chứng minh rằng việc tinh chỉnh mô hình Transformer cho các nhiệm vụ cụ thể là chìa khóa để đạt được hiệu quả cao nhất.

4.3. Xu hướng mới: Embedding Đa phương thức (Multimodal Embeddings)

Sau khi đạt được những bước tiến lớn trong việc biểu diễn ngôn ngữ con người dưới dạng vector, câu chuyện của Embedding tiếp tục phát triển với sự ra đời của các **mô hình đa phương thức** (*multimodal embeddings*).[36] Đây là bước tiến tiếp theo, nơi các vector từ nhiều loại dữ liệu khác nhau, như văn bản, hình ảnh, âm thanh và video, được tích hợp vào một không gian vector chung.[36] Mục tiêu là tạo ra một biểu diễn thống nhất, nơi các loại dữ liệu khác nhau nhưng có cùng ý nghĩa ngữ nghĩa sẽ được đặt gần nhau trong không gian vector.

Nền tảng của phương pháp này là việc sử dụng các kỹ thuật học máy tiên tiến, thường là các mạng nơ-ron, để học các mối tương quan phức tạp giữa các phương thức dữ liệu.[36] Ví dụ, một hình ảnh của một chú mèo và đoạn văn bản mô tả “con mèo dễ thương” sẽ được biểu diễn bởi các vector nằm gần nhau trong không gian Embedding chung.[37]

Các ứng dụng của Embedding đa phương thức là rất đột phá. Nó mở ra khả năng **tìm kiếm chéo phương thức** (*cross-modal retrieval*), nơi người dùng có thể sử dụng một truy vấn văn bản để tìm kiếm hình ảnh hoặc video liên quan, và ngược lại.[36] Điều này đặc biệt hữu ích trong thương mại điện tử, hệ thống gợi ý, và quản lý tài sản kỹ thuật số.[36] Ví dụ, một mô hình đa phương thức như của Google có thể tạo ra các vector 1408 chiều cho văn bản, hình ảnh và video trong cùng một không gian ngữ nghĩa, cho phép các vector này được sử dụng thay thế cho nhau cho các tác vụ như tìm kiếm hình ảnh bằng văn bản hoặc video bằng hình ảnh.[37] Xu hướng này hứa hẹn sẽ tạo ra các hệ thống AI thực sự “thông minh”, có khả năng lý luận và tương tác trên nhiều loại dữ liệu khác nhau.

Chương 5: Tổng hợp, So sánh và Đánh giá Toàn diện

5.1. Bức tranh toàn cảnh: So sánh các phương pháp qua các tiêu chí

Sự tiến hóa của Embedding từ các phương pháp truyền thống đến các mô hình ngữ cảnh hiện đại phản ánh một quá trình học hỏi không ngừng, nơi mỗi phương pháp mới ra đời để khắc phục những hạn chế của phương pháp trước đó. Bảng dưới đây cung cấp một cái nhìn tổng hợp về lộ trình này, so sánh các phương pháp dựa trên các tiêu chí quan trọng để làm rõ bức tranh toàn cảnh.

Tiêu chí	One-Hot Encoding	TF-IDF	Word2Vec (CBOW/Skip-gram)	GloVe	FastText	ELMo	BERT	SBERT
Cơ sở Lý thuyết	Mã hóa nhị phân	Thống kê tần suất	Dự đoán, Mạng NN (CBOW, Skip-gram)	Thống kê đồng xuất hiện toàn cục	Dự đoán, N-gram ký tự	Mô hình ngôn ngữ 2 chiều (Bi-LSTM)	Transformer Encoder, MLM & NSP	Tinh chỉnh BERT trên mạng Siamese
Mức độ Biểu diễn	Từ	Từ, Tài liệu	Từ	Từ	Từ (từ N-gram ký tự)	Từ trong ngữ cảnh	Từ, Câu, Đoạn văn	Câu
Tính Ngữ cảnh	Tĩnh (Không ngữ nghĩa)	Tĩnh (Không ngữ nghĩa)	Tĩnh (Không ngữ cảnh)	Tĩnh (Không ngữ cảnh)	Tĩnh (Không ngữ cảnh)	Động (Ngữ cảnh 2 chiều)	Động (Ngữ cảnh 2 chiều sâu)	Động (Ngữ cảnh câu)
Xử lý Đa nghĩa	Không	Không	Không	Không	Không	Có	Có	Có
Xử lý từ OOV	Không	Không	Không	Không	Có	Có	Có	Có
Điểm nổi bật	Đơn giản, trực quan	Phân biệt từ khóa & từ chung	Vector dày đặc, hiểu ngữ nghĩa tương đồng	Kết hợp thống kê toàn cục & ngữ cảnh cục bộ	Xử lý từ chưa biết hiệu quả	Tiên phong về ngữ cảnh 2 chiều	Nền tảng Transformer, 2 chiều sâu sắc	Tối ưu hóa đặc biệt cho Sentence Embedding
Ứng dụng chính	Biểu diễn đơn giản	Tìm kiếm, phân loại văn bản	Dịch máy, NER, phân loại	Phân tích ngữ nghĩa, tìm kiếm từ tương đồng	Ngôn ngữ hiếm, từ mới	NER, Q&A, Phân tích cảm xúc	Rất nhiều bài toán NLP, là nền tảng	So sánh câu, tìm kiếm ngữ nghĩa

Bảng so sánh này làm nổi bật quá trình tiến hóa từ các phương pháp đơn giản, không ngữ nghĩa (One-Hot), đến các mô hình thống kê (TF-IDF), rồi đến các mô hình dự đoán từ tĩnh (Word2Vec, GloVe), và cuối cùng là các mô hình ngữ cảnh mạnh mẽ dựa trên mạng nơ-ron (ELMo, BERT). Sự chuyển đổi từ “tĩnh” sang “động” là một bước ngoặt quan trọng, cho phép các mô hình xử lý những sắc thái phức tạp của ngôn ngữ mà trước đây không thể tiếp cận được.

5.2. Nhận định chiến lược: Lựa chọn mô hình phù hợp với bài toán

Trong bối cảnh hiện tại, không có một mô hình Embedding nào là "tốt nhất" cho mọi trường hợp.[18] Việc lựa chọn phương pháp phù hợp phụ thuộc vào nhiều yếu tố thực tế như đặc thù của bài toán, nguồn lực tính toán sẵn có, và yêu cầu về hiệu suất.

- **Khi nào nên dùng các mô hình tĩnh (Word2Vec, GloVe, FastText)?** Mặc dù các mô hình Transformer đã chiếm ưu thế, các mô hình tĩnh vẫn là lựa chọn tuyệt vời cho các bài toán yêu cầu tốc độ cao và tài nguyên hạn chế. Chúng vẫn tạo ra các vector chất lượng tốt và có thể được sử dụng làm lớp Embedding khởi đầu cho các mô hình học máy truyền thống.[3] **FastText** là một "cứu cánh" đặc biệt khi phải xử lý các từ hiếm hoặc các ngôn ngữ có cấu trúc phong phú, nơi các từ mới thường xuyên xuất hiện.[18]
- **Khi nào nên dùng các mô hình ngữ cảnh (BERT, SBERT)?** Khi bài toán đòi hỏi độ chính xác cao nhất và khả năng hiểu ngữ cảnh sâu sắc là yếu tố quyết định, các mô hình Transformer là lựa chọn tối ưu. Chúng vượt trội trong việc xử lý các từ đa nghĩa và các mối quan hệ phức tạp giữa các từ trong câu.[18] **BERT** là nền tảng cho các bài toán phân loại và hỏi đáp, trong khi **SBERT** là mô hình chuyên dụng, không thể thay thế cho các nhiệm vụ so sánh câu hoặc tìm kiếm ngữ nghĩa.[34] Việc tinh chỉnh (*fine-tuning*) các mô hình này cho nhiệm vụ cụ thể là chìa khóa để đạt được hiệu suất tối đa.

Tóm lại, mặc dù BERT và các biến thể của nó được coi là "state-of-the-art" cho nhiều nhiệm vụ [18], việc hiểu rõ ưu và nhược điểm của từng phương pháp trong lộ trình tiến hóa sẽ giúp các nhà phát triển và nhà nghiên cứu đưa ra lựa chọn chiến lược phù hợp, tối ưu hóa giữa hiệu suất và nguồn lực.

Chương 6: Tương lai của Embedding và Ứng dụng đột phá

6.1. Hướng đi của các mô hình ngôn ngữ lớn (LLMs)

Khái niệm Embedding, từ chỗ là một kỹ thuật biểu diễn từ, đã trở thành nền tảng cốt lõi của các **mô hình ngôn ngữ lớn (LLMs)** hiện đại như GPT, Llama và các mô hình Transformer khác.[24] Các mô hình này tiếp tục mở rộng khái niệm Embedding để xử lý các chuỗi văn bản dài hơn, thậm chí là toàn bộ đoạn văn và tài liệu. Thay vì chỉ học mối quan hệ giữa các từ, chúng học các mô hình ngữ nghĩa và ngữ pháp phức tạp ở quy mô chưa từng có. Các LLM không chỉ sử dụng Embedding để hiểu ngôn ngữ mà còn tạo ra các biểu diễn mới trong quá trình sinh văn bản, giúp chúng tạo ra các chuỗi đầu ra mạch lạc và có ngữ cảnh chặt chẽ.[2] Sự phát triển của Embedding đã trực tiếp dẫn đến khả năng đáng kinh ngạc của các LLM trong việc giải quyết các bài toán phức tạp, từ tóm tắt văn bản đến tạo nội dung sáng tạo.

6.2. Tiềm năng của Embedding Đa phương thức

Một trong những hướng đi đầy hứa hẹn nhất của Embedding là sự phát triển của các **mô hình đa phương thức** (*multimodal*).[36] Thay vì chỉ tập trung vào văn bản, các mô hình này học cách tích hợp thông tin từ nhiều nguồn dữ liệu khác nhau (văn bản, hình ảnh, video, âm thanh) vào một không gian vector thống nhất.[36] Điều này cho phép chúng thực hiện các tác vụ phức tạp hơn nhiều, chẳng hạn như tạo ra một mô tả văn bản cho một hình ảnh, hoặc sử dụng một đoạn âm thanh để tìm kiếm một video liên quan. Các mô hình đa phương thức đại diện cho bước tiến tiếp theo, hướng đến việc xây dựng các hệ thống AI thực sự "thông minh", có khả năng lý luận và tương tác với thế giới thực thông qua nhiều giác quan khác nhau.[37]

6.3. Embedding và Cơ sở dữ liệu Vector (Vector Databases)

Sự ra đời của các vector ngữ nghĩa dày đặc và có chiều cao đã tạo ra một nhu cầu mới trong lĩnh vực công nghệ thông tin: một loại cơ sở dữ liệu có thể lưu trữ, lập chỉ mục và tìm kiếm các vector này một cách hiệu quả.[36] Các cơ sở dữ liệu truyền thống không được thiết kế cho nhiệm vụ này, do đó, đã xuất hiện một lĩnh vực mới là **Cơ sở dữ liệu Vector** (*Vector Databases*).

Sự kết hợp giữa Embedding và Cơ sở dữ liệu Vector đã tạo ra một kiến trúc đột phá, được gọi là **RAG** (*Retrieval-Augmented Generation* - Sinh nội dung có tăng cường truy xuất).[29] RAG đã trở thành một kiến trúc tiêu chuẩn để vượt qua giới hạn của các LLM về kiến thức và khả năng truy cập thông tin. Cơ chế hoạt động của nó như sau:

1. **Lập chỉ mục:** Các tài liệu văn bản được chia thành các đoạn nhỏ và được chuyển đổi thành các vector Embedding. Các cặp (đoạn tài liệu, vector Embedding) này sau đó được lưu trữ trong một cơ sở dữ liệu vector.[29]
2. **Truy vấn:** Khi người dùng đưa ra một câu hỏi bằng ngôn ngữ tự nhiên, câu hỏi đó cũng được chuyển đổi thành một vector Embedding.[29]
3. **Truy xuất:** Một thuật toán tìm kiếm tương đồng (*similarity search*) được sử dụng để tìm kiếm các vector tài liệu gần nhất với vector truy vấn trong cơ sở dữ liệu vector. Điều này cho phép truy xuất các đoạn tài liệu có ngữ nghĩa liên quan nhất.[29]
4. **Sinh nội dung:** Các đoạn tài liệu đã được truy xuất sau đó được cung cấp làm ngữ cảnh cho LLM để tạo ra câu trả lời chi tiết và chính xác.

Kiến trúc RAG minh họa một cách rõ ràng sự chuyển mình của Embedding từ một khái niệm học thuật thuần túy thành một thành phần không thể thiếu trong các hệ thống AI ứng dụng. Nó cho thấy Embedding không chỉ là một công cụ tiền xử lý mà còn là một phần cốt lõi của các hệ thống tìm kiếm và lý luận phức tạp, đóng vai trò then chốt trong việc mở khóa tiềm năng của AI.

Works cited

1. Understanding Word Embeddings in Natural Language Processing | by Tahir | Medium, accessed August 27, 2025, https://medium.com/@tahirbalarabe2/understand word-embeddings-in-natural-language-processing-ac6679e96719
2. What Are Word Embeddings? | IBM, accessed August 27, 2025, https://www.ibm.com/th embeddings
3. Word Embeddings in NLP - GeeksforGeeks, accessed August 27, 2025, https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp/
4. One Hot Encoding in Machine Learning - GeeksforGeeks, accessed August 27, 2025, https://www.geeksforgeeks.org/machine-learning/ml-one-hot-encoding/
5. One-hot encoding of text data in natural language processing - Educative.io, accessed August 27, 2025, https://www.educative.io/answers/one-hot-encoding-of-text-data-in-natural-language-processing
6. Encoding categorical features in Machine learning - Viblo, accessed August 27, 2025, https://viblo.asia/p/encoding-categorical-features-in-machine-learning-bJzKm4mw59N
7. Mã hóa one-hot — Machine Learning cho dữ liệu dạng bảng, accessed August 27, 2025, https://machinelearningcoban.com/tabml_book/ch_data_processing/onehot.htm
8. Word Embedding - Tìm hiểu khái niệm cơ bản trong NLP - Viblo, accessed August 27, 2025, https://viblo.asia/p/word-embedding-tim-hieu-khai-niem-co-ban-trong-

nlp-1Je5E93G5nL
9. Word2vec — Machine Learning cho dữ liệu dạng bảng, accessed August 27, 2025, https://machinelearningcoban.com/tabml_book/ch_embedding/word2vec.html
10. Understanding and Creating Word Embeddings - Programming Historian, accessed August 27, 2025, https://programminghistorian.org/en/lessons/understandin creating-word-embeddings
11. Learning Word Embeddings with CBOW and Skip-gram | by Rishabh Singh | Medium, accessed August 27, 2025, https://medium.com/@RobuRishabh/learning-word-embeddings-with-cbow-and-skip-gram-b834bde18de4
12. TF-IDF in NLP (Term Frequency Inverse Document Frequency) | by Abhishek Jain | Medium, accessed August 27, 2025, https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d
13. Understanding TF-IDF (Term Frequency-Inverse Document Frequency) - GeeksforGeeks, accessed August 27, 2025, https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/
14. Comparison Between CBOW and Skip-Gram Models - GeeksforGeeks, accessed August 27, 2025, https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp-comparison-between-cbow-and-skip-gram-models/
15. All about Embeddings - Word2Vec, Glove, FastText, ELMo, InferSent and Sentence-BERT | Medium, accessed August 27, 2025, https://medium.com/@kashyapkathrani/all about-embeddings-829c8ff0bf5b
16. Bài 3 - Mô hình Word2Vec - Khoa học dữ liệu, accessed August 27, 2025, https://phamdinhkhanh.github.io/2019/04/29/ModelWord2Vec.html
17. Guide to Learn ELMo for Extracting Features from Text - Analytics Vidhya, accessed August 27, 2025, https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/
18. [D] What are the main differences between the word embeddings of ELMo, BERT, Word2vec, and GloVe? : r/MachineLearning - Reddit, accessed August 27, 2025, https://www.reddit.com/r/MachineLearning/comments/aptwxm/d_what_are_the_main_
19. GloVe (Global Vector) — An extension to word2vec embedding technique - Medium, accessed August 27, 2025, https://medium.com/@abhishekjainindore24/glov global-vector-an-extension-to-word2vec-embedding-technique-359ce4289908
20. How is GloVe different from word2vec? - Quora, accessed August 27, 2025, https://www.quora.com/How-is-GloVe-different-from-word2vec
21. GloVe: Global Vectors for Word Representation, accessed August 27, 2025, https://nlp.stanford.edu/projects/glove/
22. What are word embeddings like Word2Vec and GloVe? - Milvus, accessed August 27, 2025, https://milvus.io/ai-quick-reference/what-are-word-embeddings-like-word2vec-and-glove
23. What are the main differences between the word embeddings of ELMo, BERT, Word2vec, and GloVe? - Quora, accessed August 27, 2025, https://www.quora.com/Wha are-the-main-differences-between-the-word-embeddings-of-ELMo-BERT-Word2vec-and-GloVe
24. Word Embeddings: The AI Revolution - Number Analytics, accessed August 27, 2025, https://www.numberanalytics.com/blog/word-embeddings-ai-revolution
25. ELMo vs BERT - YouTube, accessed August 27, 2025, https://www.youtube.com/shorts/c
26. Giới thiệu BERT và ứng dụng vào bài toán phân loại văn bản, accessed August 27, 2025, https://blog.vietnamlab.vn/gioi-thieu-bert-va-ung-dung-vao-bai-toan-phan-loai-van-ban/
27. Google Bert Là Gì? SEO Như Thế Nào để Phù Hợp Với Thuật Toán Này?, accessed August 27, 2025, https://aznet.vn/google-bert-la-gi-cach-su-dung-khi-seo.html
28. SỬ DỤNG BERT VÀ CÂU PHỤ TRỢ CHO TRÍCH XUẤT KHÍA CẠNH TRONG

VĂN BẢN TIẾNG VIỆT - Tạp chí Khoa học, accessed August 27, 2025, https://jstic.ptit.edu.vn/jstic-ptit/index.php/jstic/article/download/858/363/4108
29. Sentence embedding - Wikipedia, accessed August 27, 2025, https://en.wikipedia.org/w
30. Sentence embedding models - GitHub Pages, accessed August 27, 2025, https://aajanki.github.io/fi-sentence-embeddings-eval/models.html
31. Top 4 Sentence Embedding Techniques using Python - Analytics Vidhya, accessed August 27, 2025, https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/
32. Unlocking Sentence Embeddings in NLP - Number Analytics, accessed August 27, 2025, https://www.numberanalytics.com/blog/ultimate-guide-sentence-embeddings-nlp
33. Question: Why BERT underperforms · Issue #80 · UKPLab/sentence-transformers - GitHub, accessed August 27, 2025, https://github.com/UKPLab/sentence-transformers/issues/80
34. Sentence Embeddings using Siamese BERT-Networks | by Sik-Ho Tsang | Medium, accessed August 27, 2025, https://sh-tsang.medium.com/brief-review-sentence-bert-sentence-embeddings-using-siamese-bert-networks-afb473c2ff51
35. Sentence Embeddings using Siamese BERT-Networks(2019) - Blanket, accessed August 27, 2025, https://ryotaro.dev/en/posts/sentence-bert_sentence_embeddings_u networks/
36. What are multimodal embeddings? - Milvus, accessed August 27, 2025, https://milvus.io/ai-quick-reference/what-are-multimodal-embeddings
37. Get multimodal embeddings | Generative AI on Vertex AI - Google Cloud, accessed August 27, 2025, https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings