1. Tổng quan về Pre-training

1.1. Pre-training là gì?

Pre-training (Tiền huấn luyện) là giai đoạn đầu tiên trong quy trình huấn luyện các mô hình ngôn ngữ lớn hiện đại. Ở giai đoạn này, mô hình sẽ học các đặc trưng, cấu trúc và ngữ nghĩa của ngôn ngữ từ một kho dữ liệu văn bản khổng lồ không cần gán nhãn (unlabeled data). Mục tiêu chính là xây dựng một nền tảng kiến thức tổng quát về ngôn ngữ , giúp mô hình "hiểu" được các mối quan hệ phức tạp và ngữ cảnh trong văn bản.

Có thể hình dung một cách đơn giản, pre-training giống như chương trình giáo dục phổ thông, cung cấp cho mô hình những kiến thức nền tảng và bao quát nhất. Sau khi hoàn thành giai đoạn này, mô hình có thể được "fine-tuning" (tinh chỉnh) để chuyên sâu vào một nhiệm vụ cụ thể, tương tự như việc học lên đại học chuyên ngành.

1.2. Mục tiêu và Lợi ích

- **Xây dựng biểu diễn ngôn ngữ (Embeddings):** Mục tiêu cốt lõi của pre-training là tạo ra các "biểu diễn" (representations hay embeddings) cho từ và câu, nắm bắt được cấu trúc ngữ pháp, ngữ nghĩa và các mối liên hệ trong văn bản.
- **Học từ dữ liệu không gán nhãn:** Pre-training tận dụng được nguồn dữ liệu văn bản khổng lồ có sẵn trên Internet mà không cần con người gán nhãn thủ công, giúp tiết kiệm chi phí và thời gian.
- **Nền tảng cho Fine-tuning:** Một mô hình đã được pre-trained, như GPT-3 của OpenAI , có thể được tinh chỉnh nhanh chóng để thực hiện các nhiệm vụ đa dạng như viết mã, dịch thuật, sáng tác thơ, hoặc trả lời câu hỏi với hiệu suất vượt trội. Ví dụ, một mô hình BERT đã được pre-trained có thể ban đầu không hiểu văn bản y khoa, nhưng sau khi được fine-tune với dữ liệu chuyên ngành, nó có thể thực hiện tốt các bài toán như nhận dạng thực thể y khoa (NER).

2. Các Dạng Pre-training Sơ khai: Từ One-Hot đến Word2Vec

Trước khi các mô hình phức tạp ra đời, việc biểu diễn từ đã trải qua nhiều giai đoạn phát triển.

2.1. One-Hot Vector

Đây là phương pháp biểu diễn mỗi từ trong từ vựng thành một vector riêng biệt. Nếu từ vựng có N từ, mỗi từ sẽ là một vector N chiều với một phần tử bằng 1 và các phần tử còn lại bằng 0. Hạn chế lớn nhất của phương pháp này là các vector từ không thể hiện được mối quan hệ ngữ nghĩa với nhau (tích vô hướng của chúng luôn bằng 0).

2.2. Word2Vec

Word2Vec là một bước đột phá, đề xuất kỹ thuật "word embedding" – ánh xạ mỗi từ thành một vector số thực có độ dài cố định. Những vector này có khả năng thể hiện mối quan hệ tương đồng và tương tự giữa các từ. Word2Vec bao gồm hai mô hình chính:

- **Continuous Bag of Words (CBOW):** Mô hình này dự đoán một từ trung tâm dựa vào các từ xung quanh nó (ngữ cảnh). CBOW hoạt động bằng cách lấy trung bình các vector của các từ ngữ cảnh để tạo ra một vector đại diện, sau đó dùng vector này để dự đoán từ trung tâm.
- **Skip-Gram:** Ngược lại với CBOW, Skip-Gram sử dụng một từ trung tâm để dự đoán các từ trong ngữ cảnh xung quanh nó. Mô hình này gán cho mỗi từ hai vector: một khi nó đóng vai trò là từ trung tâm và một khi là từ ngữ cảnh.

Thách thức tính toán và giải pháp: Cả hai mô hình Word2Vec đều đối mặt với chi phí tính toán hàm softmax rất lớn trên toàn bộ từ vựng. Để giải quyết vấn đề này, các kỹ thuật xấp xỉ như **Negative Sampling** và **Hierarchical Softmax** đã được giới thiệu, giúp giảm đáng kể độ phức tạp tính toán mà vẫn đảm bảo chất lượng của các vector từ.

2.3. GloVe (Global Vectors for Word Representation)

Skip-Gram with Global Corpus Statistics

1. Nhắc lại về Skip – Gram

   Với mô hình Skip-Gram, ta coi mỗi lần một từ “trung tâm” xuất hiện trong câu, mô hình sẽ thử sinh ra từng từ “ngữ cảnh” xung quanh sao cho có xác suất cao nhất. Cách tính xác suất này được chia làm hai bước:

   Trước hết mỗi từ trong từ điển được gán hai loại biểu diễn riêng biệt dưới dạng các vectơ số, một vectơ khi nó đóng vai trò “trung tâm” và một vectơ khi nó đóng vai trò “ngữ cảnh”.

   Để dự đoán khả năng một từ ngữ cảnh xuất hiện bên cạnh từ trung tâm, mô hình tính “độ tương đồng” giữa hai vectơ tương ứng (vectơ của từ trung tâm và vectơ của từ ngữ cảnh), rồi biến kết quả đó thành một con số dương – càng lớn càng có nghĩa mô hình tin hai từ có liên quan chặt chẽ.

   Cuối cùng, để đảm bảo rằng tổng mọi xác suất cho tất cả các từ trong từ điển bằng 1, mô hình chia con số “độ tương đồng” của từng cặp cho tổng tất cả các độ tương đồng của cặp từ trung tâm với mọi từ khác trong từ điển.

Kết quả là với mỗi cặp (từ trung tâm, từ ngữ cảnh), Skip-Gram gán một xác suất thể hiện độ chắc chắn của mô hình rằng "từ ngữ cảnh đó" thực sự nên xuất hiện xung quanh "từ trung tâm" trong dữ liệu huấn luyện.

2. Thống kê toàn cục với Skip-Gram

Đếm hết mọi cặp từ trung tâm–ngữ cảnh trên toàn văn bản

- Trước hết, với mỗi từ đóng vai trò "trung tâm", ta quét toàn bộ văn bản và đếm xem mỗi từ khác xuất hiện bao nhiêu lần trong vùng ngữ cảnh quanh nó. Con số này gọi là **đếm cặp**.

- Đồng thời, ta cũng tính tổng số lần mọi từ (bất kể từ nào) từng xuất hiện bên cạnh từ trung tâm đó. Kết quả là hai con số:

    - Số lần mỗi từ kia làm "ngữ cảnh" (gọi là multiplicity).

    - Tổng số lần xuất hiện của mọi ngữ cảnh với từ đấy (tổng multiplicity).

Chuyển các con số đếm thành phân phối thực nghiệm

- Từ những đếm này, ta tạo ra một "phân phối thực nghiệm" theo cách rất đơn giản: chỉ lấy số lần một từ đóng vai trò ngữ cảnh chia cho tổng số lần mọi từ được làm ngữ cảnh. Kết quả là, với mỗi cặp trung tâm–ngữ cảnh, ta có xác suất thực tế mà từ đó xuất hiện.

Đặt target cho mô hình là khớp phân phối đó

- Mô hình của chúng ta cũng sinh ra một phân phối "dự đoán" cho mỗi cặp trung tâm–ngữ cảnh.

- Muốn làm sao cho phân phối dự đoán và phân phối thực nghiệm càng giống nhau càng tốt, ta dùng **cross-entropy** – về cơ bản là đo độ sai khớp giữa hai phân phối.

Hàm mất mát tổng hợp toàn bộ

Cuối cùng, hàm mất mát được tính bằng cách **cộng** lại độ sai khớp (cross-entropy) của mọi cặp, mỗi cặp được **trọng số** theo số lần nó xuất hiện

trong corpus. Nhờ vậy, những cặp xuất hiện nhiều lần sẽ đóng góp nhiều hơn vào loss, còn cặp hiếm thì ít ảnh hưởng.

3. Hạn chế

   Vấn đề với softmax đầy đủ và cross-entropy

- Nếu với mỗi bước huấn luyện ta phải tính hàm softmax trên toàn bộ từ điển để chuẩn hóa xác suất, chi phí sẽ rất lớn, nhất là khi từ điển có hàng trăm nghìn từ.
- Thêm nữa, khi dùng cross-entropy để so sánh phân phối dự đoán và phân phối thật, mọi cặp từ—even rất hiếm—đều được gán trọng số như nhau, khiến các "sự kiện hiếm" đôi khi lại đóng góp quá nhiều vào tổng loss và gây nhiễu cho việc học.

Vì vậy người ta mới dùng Negative Sampling hoặc Hierarchical Softmax để:

- Giảm đáng kể chi phí tính toán (không phải duyệt hết cả từ điển).
- Đồng thời kiểm soát ảnh hưởng của các cặp từ rất ít xuất hiện.

GloVe

GloVe là một mô hình cải tiến bằng cách kết hợp cả thông tin thống kê toàn cục (global statistics) từ ma trận đồng xuất hiện (co-occurrence matrix) với các phương pháp học cục bộ (local context window) của Word2Vec. Thay vì dùng cross-entropy, GloVe tối ưu hóa một hàm mất mát dựa trên bình phương lỗi (squared loss) giữa log của số lần đồng xuất hiện và tích vô hướng của các vector từ, có thêm các hệ số bias.

1. **Dùng squared loss thay vì negative-sampling/softmax**

   Trong GloVe, thay vì đối chiếu hai phân phối xác suất đã chuẩn hóa, người ta so sánh trực tiếp "độ lớn" của số lần hai từ cùng xuất hiện với "độ lớn" mà mô hình dự đoán. Cụ thể:

   - Với mỗi cặp từ, ta ghi nhận số lần chúng đi cùng nhau trong toàn bộ dữ liệu.
   - Mô hình – thông qua hai véc-tơ đại diện cho hai từ – sinh ra một điểm số, điểm số này càng cao khi hai véc-tơ càng "hợp nhau."

- Người ta chuyển cả hai con số (số lần xuất hiện chung và điểm số do mô hình đưa ra) sang dạng logarit để giảm ảnh hưởng của các giá trị quá lớn.

- Cuối cùng, ta tính hiệu giữa hai giá trị logarit đó và bình phương nó lên. Kết quả bình phương này là độ "chênh" giữa thực tế và dự đoán.

- Khi huấn luyện, GloVe sẽ điều chỉnh các véc-tơ sao cho hiệu logarit ấy càng sát, nghĩa là mô hình học được cách cho ra điểm số phù hợp với tần suất xuất hiện chung của từ trong dữ liệu.

2. Thêm bias terms cho mỗi từ

Trong GloVe, mỗi từ được gán thêm hai con số điều chỉnh nhỏ:

1. Bias trung tâm cho trường hợp từ đó đóng vai trò "từ chính" (center).

2. Bias ngữ cảnh cho trường hợp từ đó đóng vai trò "từ ngữ cảnh" (context).

Khi mô hình tính mức "hợp nhau" của hai từ dựa trên hai véc-tơ đại diện, nó sẽ cộng thêm những con số bias này để có thể "dịch" (shift) kết quả cho khớp với tần suất xuất hiện chung của hai từ đó trong dữ liệu. Nhờ vậy, mô hình không chỉ dựa vào độ tương đồng của véc-tơ mà còn linh hoạt điều chỉnh sao cho phù hợp với thống kê thực tế.

3. Trọng số hoá theo $h(x_{ij})$

Trong GloVe, người ta nhận thấy rằng không phải mọi cặp từ đều quan trọng như nhau:

- Những cặp từ chỉ xuất hiện rất ít lần không nên được "tăng cường" quá mạnh, để tránh làm nhiễu tín hiệu.

- Ngược lại, những cặp từ xuất hiện quá thường xuyên cũng không được phép chi phối quá mức, kẻo làm mất cân bằng trong quá trình huấn luyện.

Để khắc phục, GloVe gán cho mỗi cặp từ một "hàm trọng số" đặc biệt, có những đặc điểm sau:

- Khi tần suất chung rất nhỏ, trọng số gần như bằng không.

- Khi tần suất tăng lên, trọng số cũng tăng theo nhưng rất chậm, bởi vì người ta dùng một quy luật lũy thừa với số mũ nhỏ hơn một.

- Đến một ngưỡng tần suất xác định (ví dụ xung quanh vài chục đến vài trăm lần), trọng số sẽ giữ nguyên ở mức tối đa và không tăng thêm nữa.

4. **Lưu ý**

GloVe có tính “đối xứng”: nếu một từ A nằm trong ngữ cảnh của từ B thì ngược lại từ B cũng nằm trong ngữ cảnh của từ A. Nói cách khác, số lần hai từ này xuất hiện chung với nhau là như nhau.

Về lý thuyết, việc coi một vector là “trung tâm” và vector kia là “ngữ cảnh” là tương đương. Tuy nhiên trong thực tế, người ta thường lấy tổng của hai vector này để làm kết quả cuối cùng, nhằm gom cả hai nguồn thông tin.

Ưu điểm của GloVe là:

- Kết hợp được thông tin cục bộ (tương tự skip-gram, dựa vào những từ xung quanh) và thông tin tổng quát (tần suất xuất hiện chung của các cặp từ).

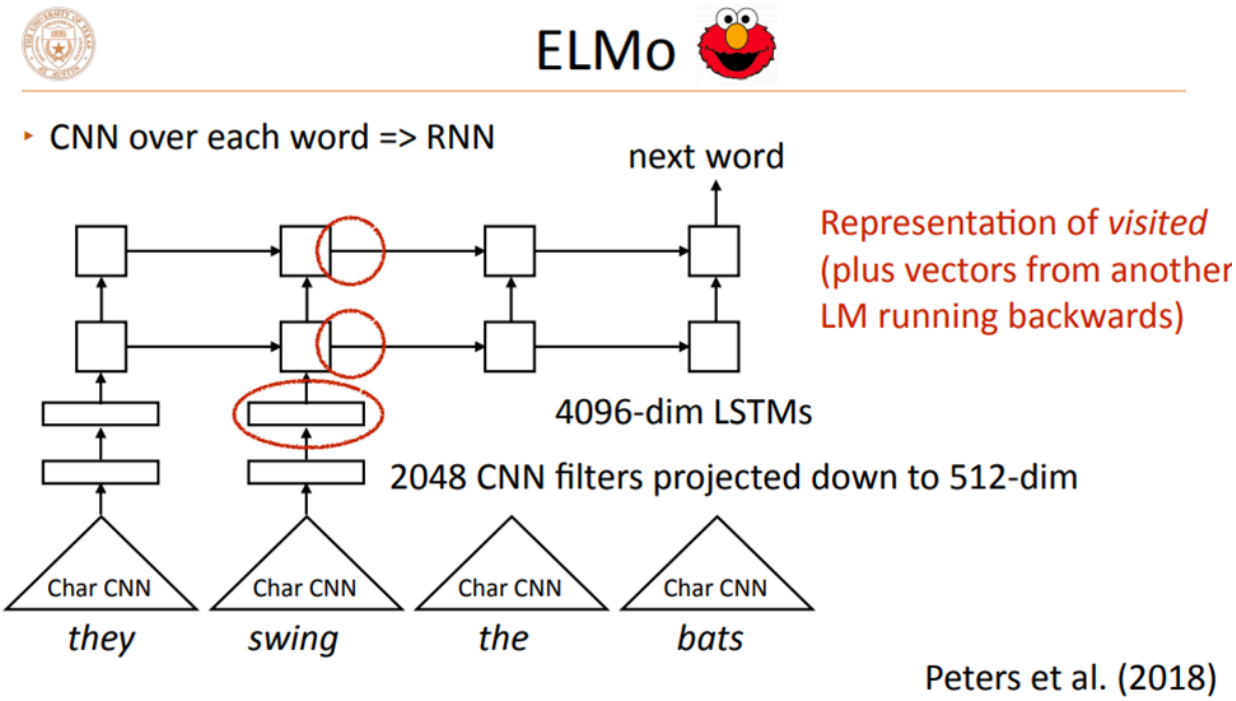
- Sử dụng hàm lỗi rất đơn giản, chỉ cần so sánh kết quả dự đoán với thước đo tần suất đã chuyển đổi.

- Dễ dàng huấn luyện theo từng lô nhỏ (mini-batch) bằng thuật toán SGD, chỉ tập trung vào những cặp từ đã từng xuất hiện chung ít nhất một lần, nên tiết kiệm đáng kể chi phí tính toán.

3. Sự Trỗi dậy của các Mô hình Phụ thuộc vào Ngữ cảnh

Một hạn chế lớn của Word2Vec và GloVe là chúng tạo ra một vector biểu diễn cố định cho mỗi từ, bất kể ngữ cảnh xuất hiện của từ đó. Ví dụ, từ "bank" trong "go to the bank to deposit money" (ngân hàng) và "go to the bank to sit down" (bờ sông) sẽ có cùng một vector biểu diễn, gây mất mát thông tin ngữ nghĩa. Để khắc phục điều này, các mô hình phụ thuộc vào ngữ cảnh đã ra đời.

3.1. ELMo (Embeddings from Language Models)



ELMo là một trong những mô hình tiên phong trong việc tạo ra các embedding phụ thuộc vào ngữ cảnh.

- **Kiến trúc:** ELMo sử dụng một mạng Bi-directional LSTM (BiLSTM) hai chiều đã được pre-trained trên tác vụ mô hình hóa ngôn ngữ (dự đoán từ tiếp theo và từ phía trước).
- **Cơ chế hoạt động:** Khi áp dụng cho một tác vụ cụ thể, các trọng số của BiLSTM được giữ nguyên (frozen). Với mỗi câu đầu vào, ELMo tạo ra các biểu diễn từ các lớp của BiLSTM. Các biểu diễn này sau đó được kết hợp lại (thường bằng tổng có trọng số) để tạo ra "vector ELMo" cuối cùng cho mỗi token. Vector này sẽ được ghép nối với embedding đầu vào của mô hình cho tác vụ đó.

Luồng hoạt động của ELMo

1. Pretrain một mạng BiLSTM hai chiều trên tác vụ language modeling (dự đoán token tiếp theo & token trước đó) trên tập dữ liệu lớn.
2. Khi dùng cho downstream task (ví dụ phân tích tình cảm, NER, QA…), giữ nguyên (freeze) toàn bộ trọng số của BiLSTM đã pretrained.
3. Với mỗi câu đầu vào, chạy qua BiLSTM để thu được representation ở từng lớp và tại từng vị trí từ.
4. Kết hợp các representation này (thường là weighted sum) làm “ELMo vector” cho mỗi token.
5. Nối (concatenate) ELMo vector vào input embedding cũ (ví dụ GloVe) của mô hình downstream, rồi train mô hình downstream như bình thường (các weight của mạng task-specific được cập nhật).

3.2. GPT (Generative Pre-trained Transformer)

GPT sử dụng kiến trúc Transformer nhưng chỉ theo một hướng từ trái sang phải (unidirectional), làm cho nó rất mạnh trong các tác vụ sinh văn bản. Tuy nhiên, chính vì kiến trúc một chiều này, GPT không thể nắm bắt trọn vẹn ngữ cảnh hai chiều. Ví dụ, khi xử lý từ "bank", GPT chỉ có thể dựa vào ngữ cảnh bên trái ("I went to the bank") và sẽ tạo ra cùng một biểu diễn cho từ này dù nghĩa của nó là gì trong phần còn lại của câu.

Đặc điểm của GPT:

1. Kiến trúc “task-agnostic”
2. Fine-tuning toàn bộ mô hình
3. Hiệu năng trên nhiều tác vụ
4. Vì GPT dựa trên mô hình sinh văn bản tự hồi quy, nó chỉ “nhìn” ngữ cảnh bên trái của mỗi token.

Ví dụ: trong hai câu

“I went to the bank to deposit cash.”

“I went to the bank to sit down.”

khi đến token “bank”, cả hai câu đều có cùng ngữ cảnh bên trái (“I went to the bank”), nên GPT sẽ sinh ra cùng một representation cho “bank” dù một lần nó là tổ chức tài chính, lần khác là cần cẩu (nghĩa “bến sông”).

Đó là lý do GPT không phân biệt được nghĩa của từ dựa vào ngữ cảnh hai chiều như các mô hình bidirectional (ELMo, BERT) làm được**.**

4. BERT: Cuộc Cách mạng về Biểu diễn Ngữ cảnh Hai chiều

BERT (Bidirectional Encoder Representations from Transformers) ra đời năm 2018 và đã tạo nên một cuộc cách mạng thực sự bằng cách kết hợp những ưu điểm của các mô hình trước đó.

- **Kiến trúc hai chiều:** BERT sử dụng bộ mã hóa của Transformer (Transformer Encoder) để xử lý toàn bộ chuỗi văn bản cùng một lúc, cho phép mỗi token có thể "nhìn thấy" và học hỏi từ tất cả các token khác trong câu, cả bên trái và bên phải.

- **Kiến trúc linh hoạt:** Giống GPT, BERT có thể được tinh chỉnh cho nhiều tác vụ NLP khác nhau với thay đổi tối thiểu về kiến trúc. Trong quá trình fine-tuning, toàn bộ các tham số của mô hình pre-trained đều được cập nhật.

Bốn thay đổi chính của BERT so với ELMo:

- Sử dụng kiến trúc Transformer thay cho LSTM
- Mô hình hai chiều với mục tiêu “Masked LM” thay vì ngôn ngữ mô hình thông thường
- Tiến hành tinh chỉnh (fine-tune) thay vì giữ cố định khi đánh giá
- Làm việc trên các “mảnh từ” (word pieces) thông qua mã hóa cặp byte (byte-pair encoding)

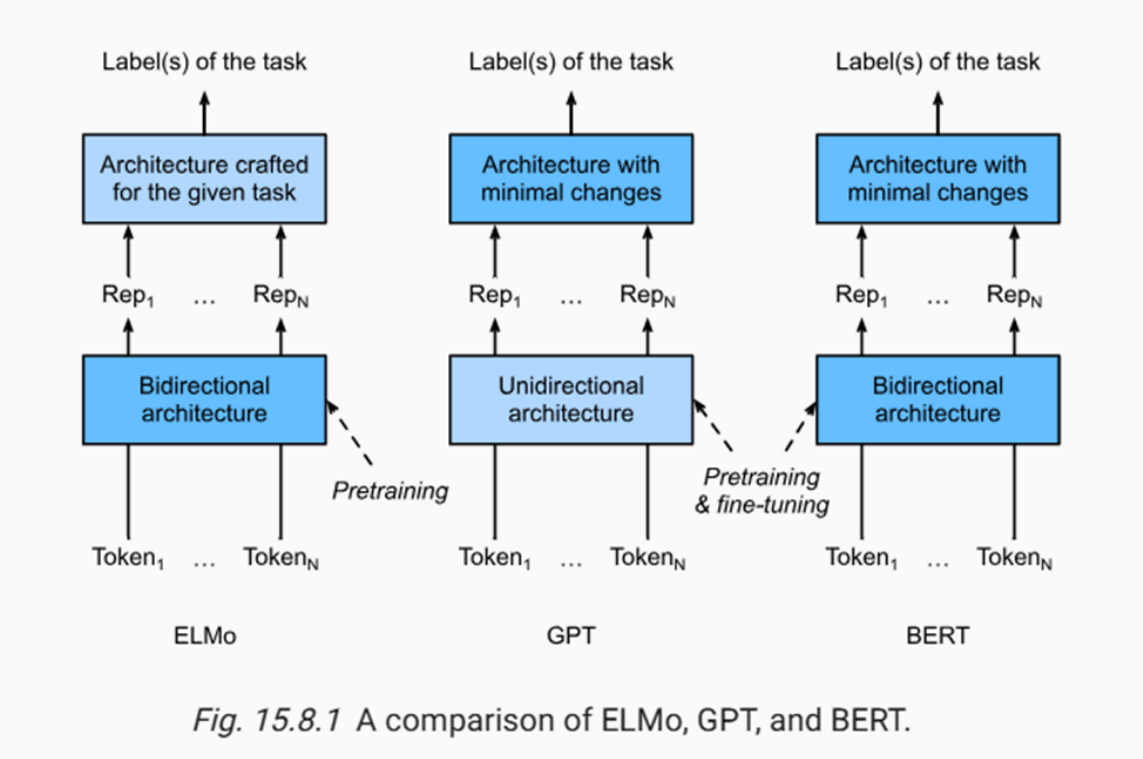


Fig. 15.8.1 A comparison of ELMo, GPT, and BERT.

Ảnh minh họa ELMo, GPT, BERT

4.1. Biểu diễn đầu vào của BERT (Input Representation)

4.1.1. Xây dựng chuỗi token và segment IDs

Ở phần này có hai loại “token đặc biệt”:

- **CLS**: luôn được thêm vào đầu dãy, và lớp biểu diễn tại vị trí này sẽ được dùng để đại diện cho toàn bộ chuỗi (ví dụ khi làm tác vụ phân loại).

- **SEP**: dùng để đánh dấu kết thúc mỗi câu; nếu bạn có hai câu ghép đôi, token này cũng sẽ đứng giữa chúng.

Với một chuỗi chỉ gồm một câu, ta thực hiện như sau:

1. Đặt token **CLS** lên đầu.

2. Tiếp theo là tất cả các token tạo nên câu đó.

3. Cuối cùng thêm token **SEP** để đóng lại.

Về phần "phân đoạn" (segment): vì chỉ có một câu, nên tất cả các token trong dãy (từ CLS đến SEP) đều thuộc cùng một phân đoạn (gọi là phân đoạn A).

4.1.2. Kết hợp ba loại embedding

Để hiểu được ngữ cảnh một cách toàn diện, đầu vào của BERT là tổng hợp của ba loại embedding:

1. **Token Embeddings:** Biểu diễn cho mỗi token (từ hoặc mảnh từ).
2. **Segment Embeddings:** Giúp mô hình phân biệt giữa các câu khác nhau khi đầu vào là một cặp câu (ví dụ, câu A và câu B).
3. **Positional Embeddings:** Cung cấp thông tin về vị trí của mỗi token trong chuỗi.

Nhờ vậy BERT vừa biết từ nào đang ở đó (token embeddings), biết thuộc câu A hay B (segment embeddings) và biết vị trí của nó (positional embeddings) — kết hợp ba thông tin này cho mọi downstream task một input representation chung mạnh mẽ.

4.2. Các Nhiệm vụ Pre-training của BERT

BERT được pre-trained đồng thời trên hai nhiệm vụ độc đáo:

a. Masked Language Modeling (MLM)

Thay vì dự đoán từ tiếp theo như các mô hình ngôn ngữ truyền thống, MLM cho phép BERT học ngữ cảnh hai chiều.

- **Cơ chế:** Khoảng 15% số token trong chuỗi đầu vào được chọn ngẫu nhiên. Sau đó, các token này được xử lý theo một trong ba cách:

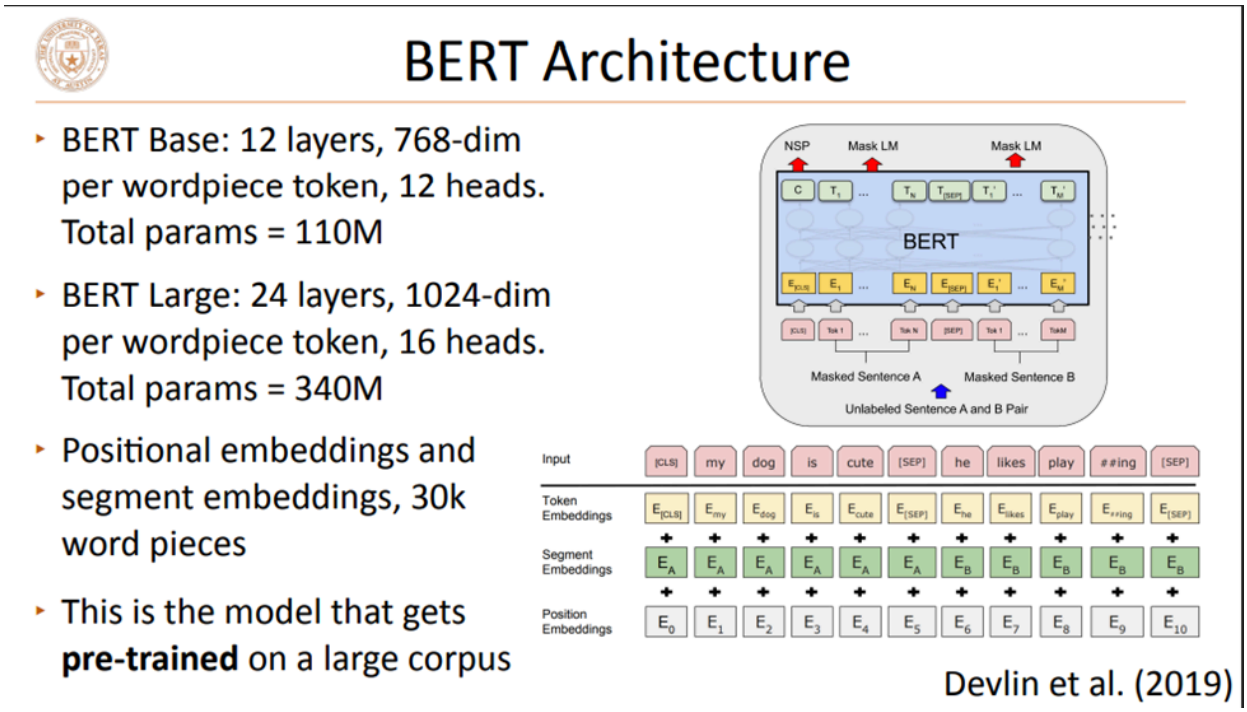
    - **80%** bị thay thế bằng một token đặc biệt là `[MASK]`.
    - **10%** bị thay thế bằng một token ngẫu nhiên khác từ từ vựng.
    - **10%** được giữ nguyên.
- **Mục tiêu:** Nhiệm vụ của mô hình là dự đoán đúng các token gốc đã bị thay đổi. Việc thêm "nhiễu" bằng cách thay thế ngẫu nhiên và giữ nguyên giúp mô hình giảm sự phụ thuộc vào token `[MASK]` và buộc nó phải dựa nhiều hơn vào ngữ cảnh xung quanh để đưa ra dự đoán.

b. Next Sentence Prediction (NSP)

Nhiệm vụ này giúp BERT hiểu được mối quan hệ logic giữa các câu, một điều mà MLM không thể làm được.

- **Cơ chế:** Mô hình nhận vào hai câu (A và B) và phải xác định xem câu B có phải là câu thực sự đi liền sau câu A trong văn bản gốc hay không.
- **Tạo dữ liệu:** 50% các cặp câu là các cặp liền kề thực sự (nhãn `IsNext`), và 50% còn lại là các cặp mà câu B được chọn ngẫu nhiên từ kho dữ liệu (nhãn `NotNext`).
- **Lợi ích:** NSP rất quan trọng đối với các tác vụ đòi hỏi sự suy luận trên nhiều câu như Question Answering (QA) và Natural Language Inference (NLI).

4.3. Kiến trúc của BERT



Hai phiên bản chính

- **BERT Base** gồm 12 lớp Transformer, mỗi từ dạng “word-piece” được biểu diễn thành một vector dài 768 chiều, với 12 đầu chú ý, tổng số tham số khoảng 110 triệu.

- **BERT Large** mở rộng lên 24 lớp, mỗi vector dài 1024 chiều, với 16 đầu chú ý, tổng số tham số khoảng 340 triệu.

Embedding đầu vào

- Mỗi token (từ hoặc mảnh từ) được chuyển qua một lớp “token embedding”.
- Thêm vào đó có hai loại nhúng khác:
    - **Segment embedding**, để phân biệt token thuộc câu thứ nhất hay câu thứ hai.
    - **Position embedding**, để ghi nhớ vị trí của token trong chuỗi.
- Tất cả ba loại nhúng này được cộng lại thành một vector duy nhất rồi đưa vào tầng Transformer.

Kích thước từ vựng và tiền huấn luyện

- BERT dùng bảng từ vựng khoảng 30 nghìn mảnh từ.
- Trước khi mang đi tinh chỉnh (fine-tune) cho các tác vụ cụ thể, BERT được tiền huấn luyện trên một lượng văn bản rất lớn với hai nhiệm vụ:
    - **Masked Language Modeling**: tự động đoán những token bị che mất.
    - **Next Sentence Prediction**: nhận diện xem hai câu có phải là liên tiếp hay không.

5. Kết luận

Pre-training đã trở thành một giai đoạn không thể thiếu trong Xử lý Ngôn ngữ Tự nhiên hiện đại. Bắt đầu từ những kỹ thuật biểu diễn từ tĩnh như Word2Vec và GloVe, lĩnh vực này đã có những bước tiến nhảy vọt với sự ra đời của các mô hình dựa trên ngữ cảnh như ELMo và đặc biệt là các mô hình Transformer hai chiều như BERT. Bằng cách học các biểu diễn ngôn ngữ sâu sắc và phức tạp từ lượng dữ liệu khổng lồ, các mô hình được pre-trained đã thiết lập những kỷ lục

mới về hiệu suất trên hàng loạt các tác vụ NLP và mở ra một kỷ nguyên mới cho sự phát triển của trí tuệ nhân tạo.