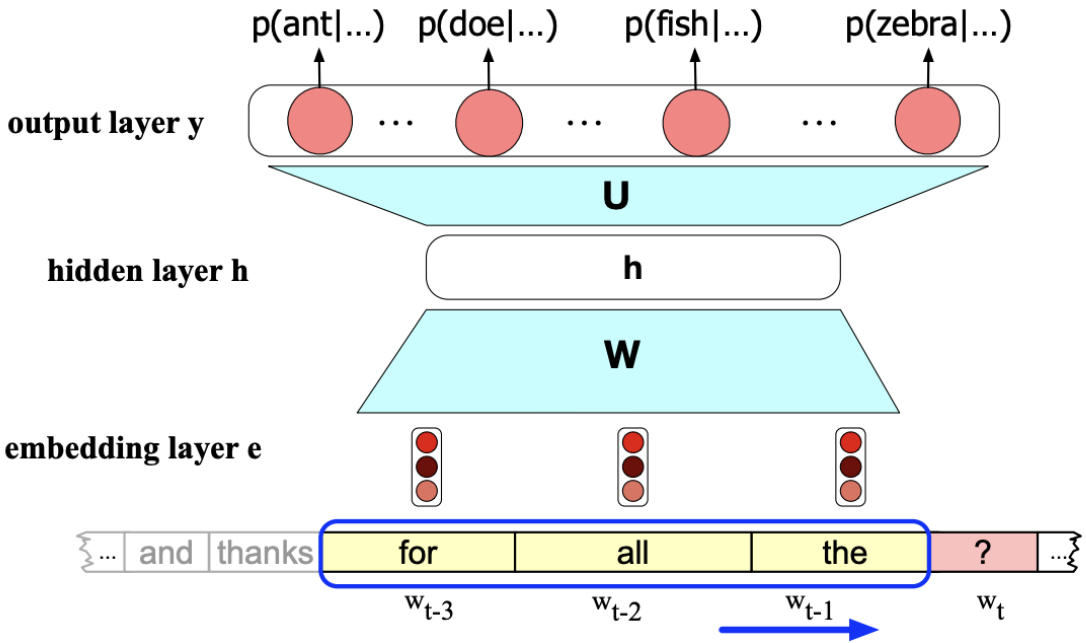
Các kiến trúc Deep Learning cho xử lý chuỗi

Mô hình ngôn ngữ N-gram

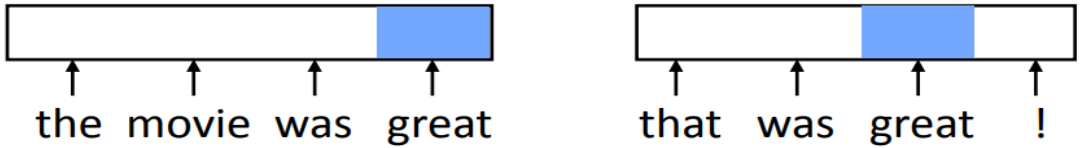
P(fish|Thanks for all the)	*5-gram*
P(fish|for all the)	*4-gram*
P(fish|all the)	*3-gram*
P(fish|the)= C(the fish)/C(the)	*2-gram*

Mô hình ngôn ngữ thần kinh



RNN motivation

- Feedforward NNs không thể xử lý đầu vào có độ dài thay đổi: mỗi vị trí trong vector đặc trưng có ngữ nghĩa cố định.



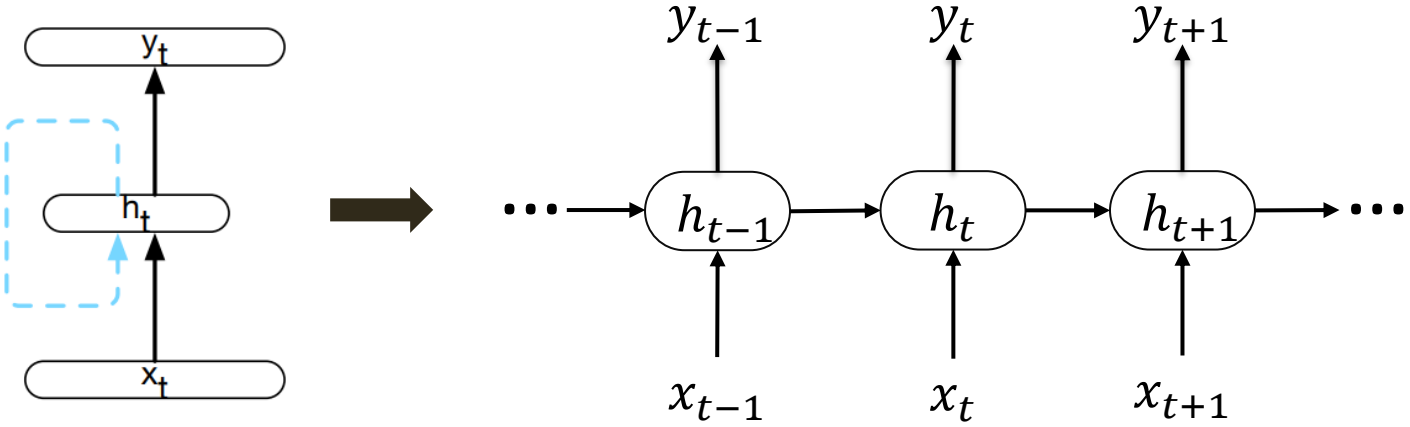
- Bị giới hạn ngữ nghĩa, không thể khái quán hóa qua các ngữ cảnh của những từ tương tự.
- Việc sử dụng sliding windows gây khó khăn cho mạng học các mẫu có phát sinh: cách ý nghĩa của các từ trong cụm từ kết hợp với nhau.

Recurrent Neural Networks

Transformer Networks

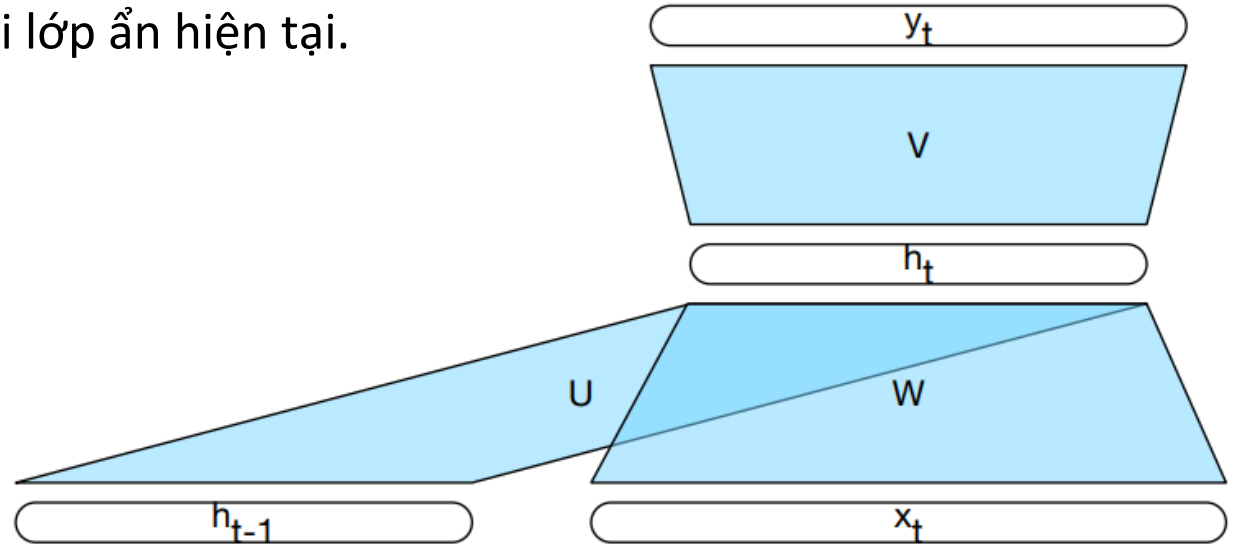
Cấu trúc RNN đơn giản (Elman, 1990)

- Giá trị kích hoạt (activation value) của lớp ẩn (hidden layer) phụ thuộc vào đầu vào hiện tại cũng như giá trị kích hoạt của lớp ẩn từ bước thời gian trước đó.

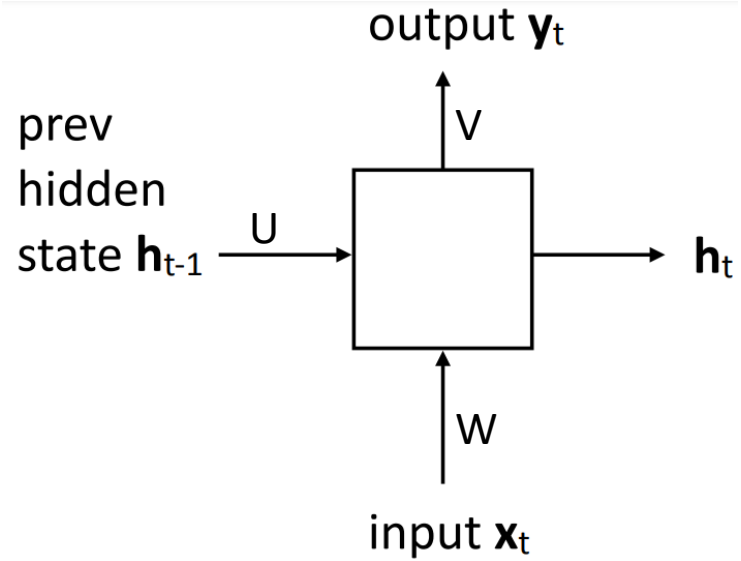


RNN như một feedforward network

- Nhận một vectơ đầu vào và các giá trị lớp ẩn từ bước thời gian trước, vẫn thực hiện phép tính feedforward chuẩn.
- Thay đổi đáng kể nhất nằm ở tập trọng số mới U, kết nối lớp ẩn từ bước thời gian trước với lớp ẩn hiện tại.



Suy luận trong RNN



- Với W, U, V là các ma trân trọng số.
- Cập nhật trạng thái ẩn dựa trên đầu vào và trạng thái ẩn trước đó.

$$h_t = g(Uh_{t-1} + Wx_t)$$

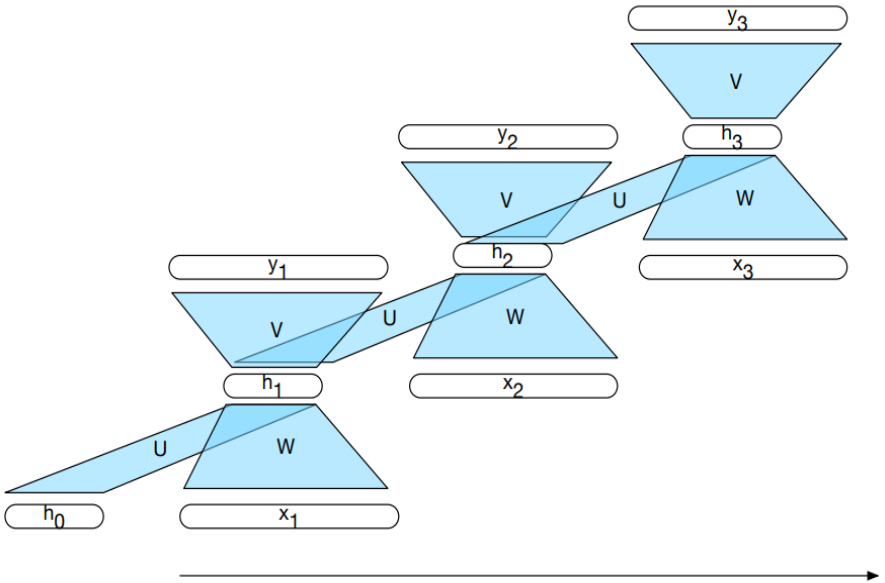
- Tính toán đầu ra từ trạng thái ẩn.

$$y_t = f(Vh_t)$$

- Nêu muốn lấy phân phối xác xuất trong trường hợp phân loại.

$$y_t = \text{softmax}(Vh_t)$$

Huấn luyện



- Sử dụng thuật toán lan truyền ngược (backpropagation) để huấn luyện các trọng số trong RNN.
- Các lớp mạng được sao chép cho mỗi bước thời gian, các trọng số U, V và W được chia sẻ chung trong tất cả các bước thời gian.

Mô hình ngôn ngữ RNN

Phân phối xác suất đầu ra

$$y_t = \text{softmax}(Vh_t)$$

Trạng thái ẩn

$$h_t = g(Uh_{t-1} + We_t)$$

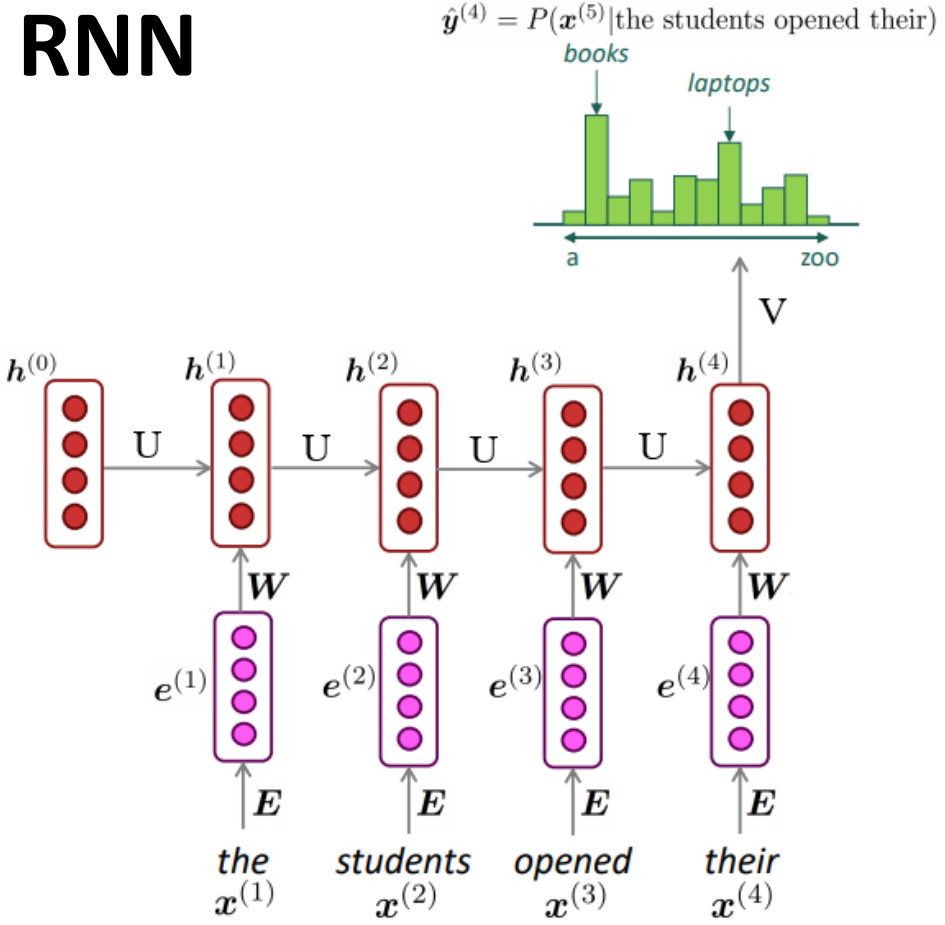
h_0 *là trạng thái ẩn đầu tiên*

word embeddings

$$e_t = Ex_t$$

Từ/one-hot-vector

$$x_t$$



Chú ý: Chuỗi đầu vào có thể dài hơn nhiều

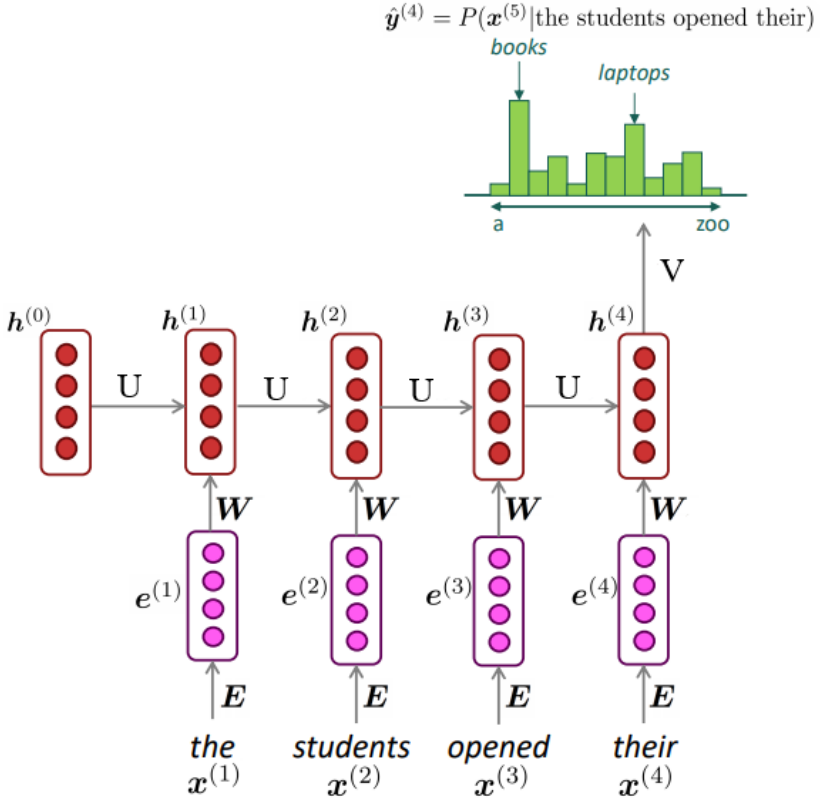
Mô hình ngôn ngữ RNN

Ưu điêm RNN:

- Có thể xử lý đầu vào với bất kỳ độ dài nào
- Tính toán cho bước t có thể sử dụng thông tin từ nhiều bước trước
- Kích thước mô hình không tăng đối với ngữ cảnh đầu vào dài hơn
- Các trọng số giống nhau được áp dụng trên mọi bước thời gian, do đó, có sự đối xứng trong cách xử lý đầu vào.

Nhược điểm RNN

- Tính toán lặp lại chậm
- Trong thực tế, khó truy cập thông tin từ nhiều bước xa.



Huấn luyện ngôn ngữ RNN

- Nhận một kho văn bản lớn là một chuỗi các từ $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Cung cấp giá trị của các thông số cho RNN-LM, tính toán phân phối đầu ra $\hat{y}^{(t)}$ tại mỗi bước t
- Hàm mất mát ở bước t là cross-entropy giữa phân phối xác suất được dự đoán $\hat{y}^{(t)}$ và từ đúng tiếp theo $y^{(t)}$

$$\begin{aligned} \mathbf{e}_t &= \mathbf{E}\mathbf{x}_t \\ \mathbf{h}_t &= g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{e}_t) \\ \mathbf{y}_t &= \text{softmax}(\mathbf{V}\mathbf{h}_t) \end{aligned}$$

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

- Tính trung bình giá trị này để có được tổn thất chung cho toàn bộ tập huấn luyện

$$\frac{1}{T} \sum_{t=1}^{T} L_{CE}$$

Huấn luyện ngôn ngữ RNN

Teacher forcing

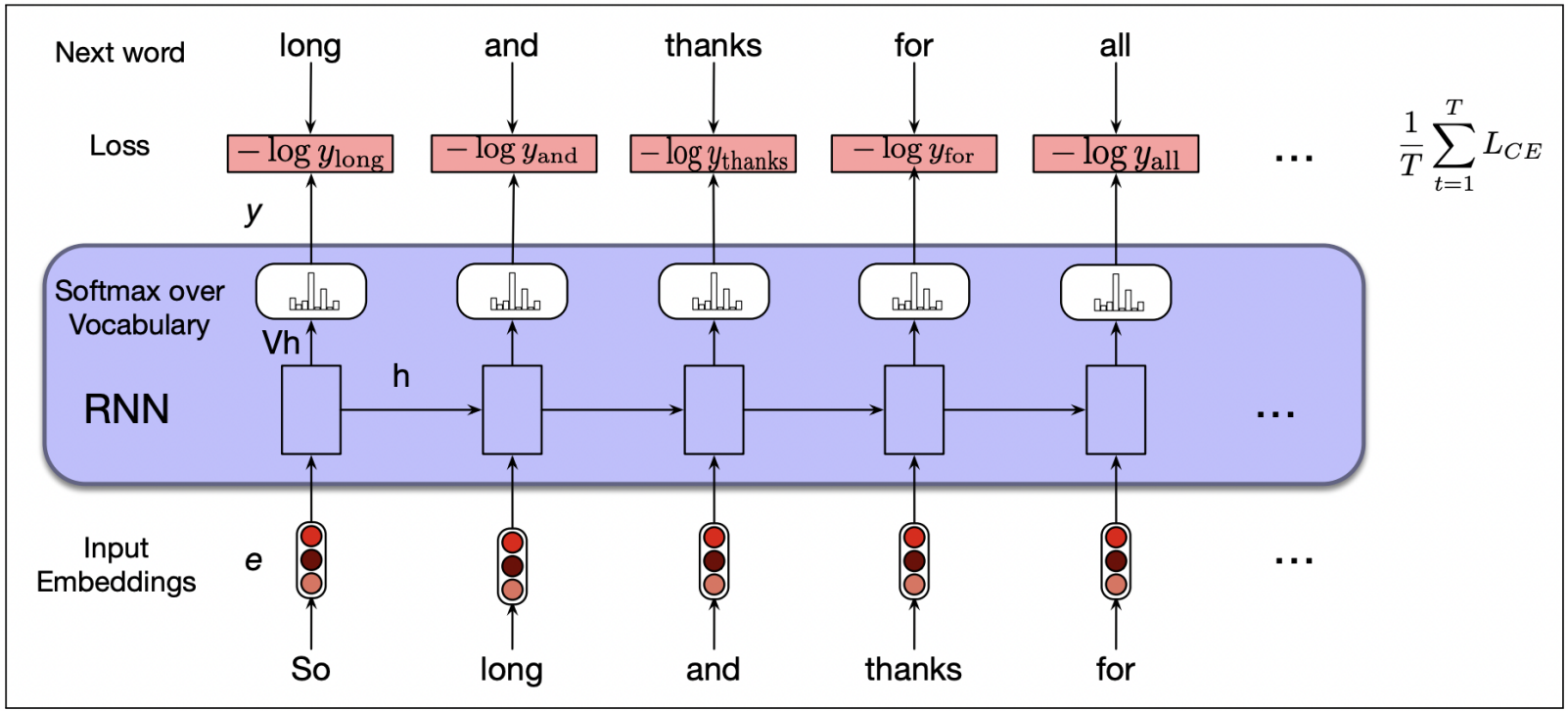


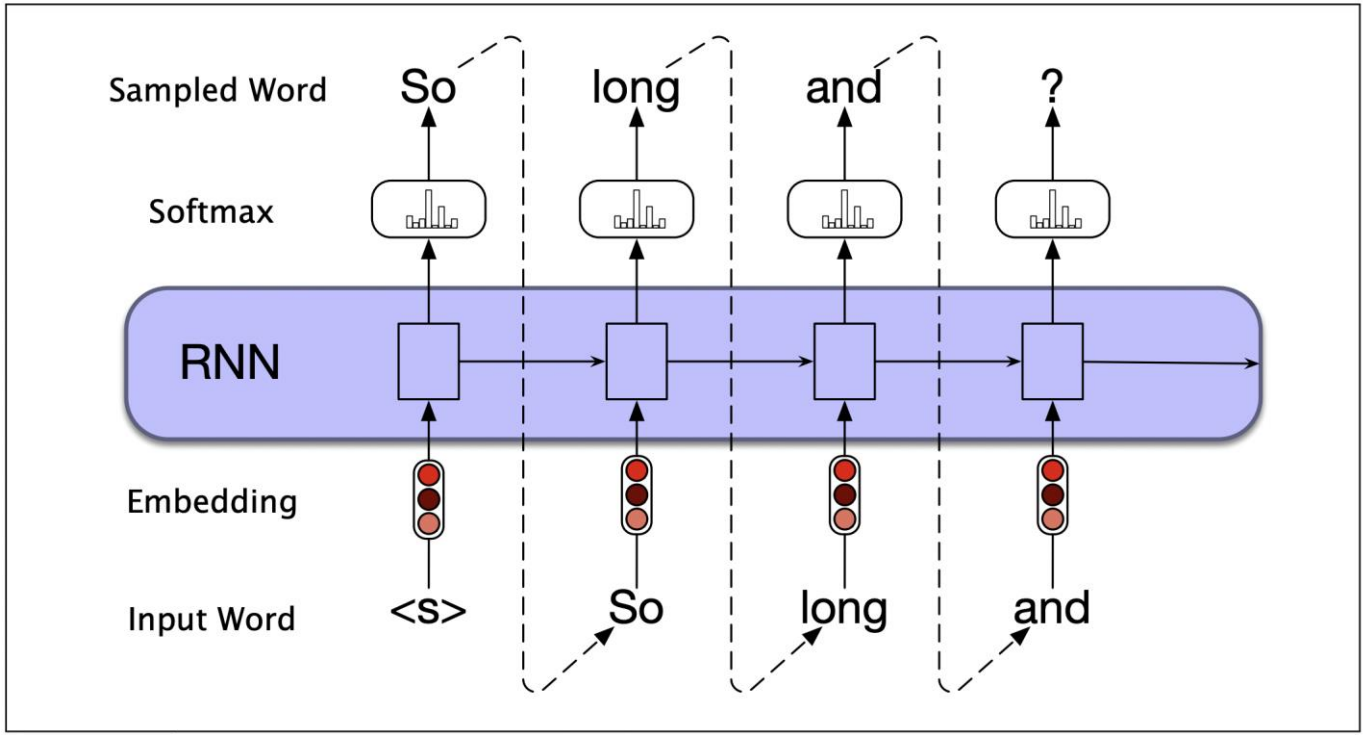
Figure 9.6 Training RNNs as language models.

Tóm tắt

- Mô hình ngôn ngữ (ML): Một hệ thống dự đoán từ tiếp theo
- Recurrent Neural Network: Một họ các mạng thần kinh:
  - Lấy đầu vào tuần tự có độ dài bất kỳ
  - Áp dụng các trọng số giống nhau cho mỗi bước
  - Có thể trả về đầu ra một cách tùy chọn ở mõi bước
- Recurrent Neural Network ≠ mô hình ngôn ngữ
- Đã chứng minh rằng RNN là một cách tuyệt vời để xây dựng LM.
- Nhưng RNN hữu ích nhiều hơn thế nữa.

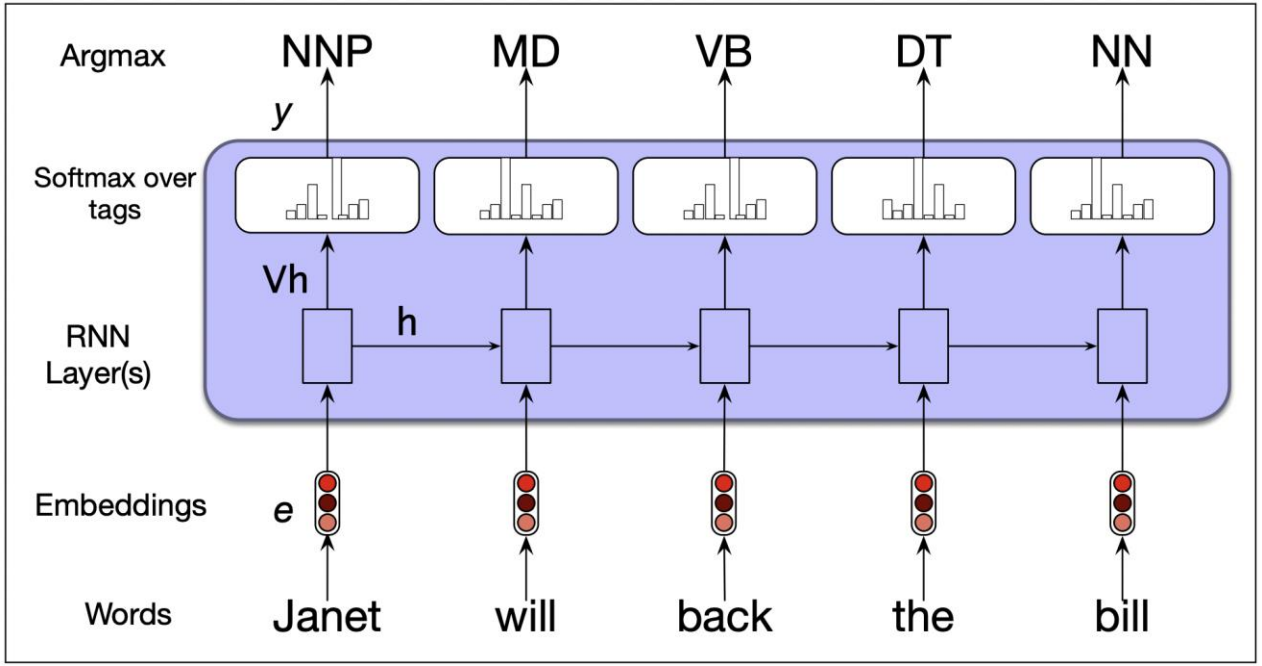
Autoregressive generation với LM dựa trên RNN

- Ví dụ: Sử dụng một mô hình ngôn ngữ được đào tạo để tạo ra các câu ngẫu nhiên



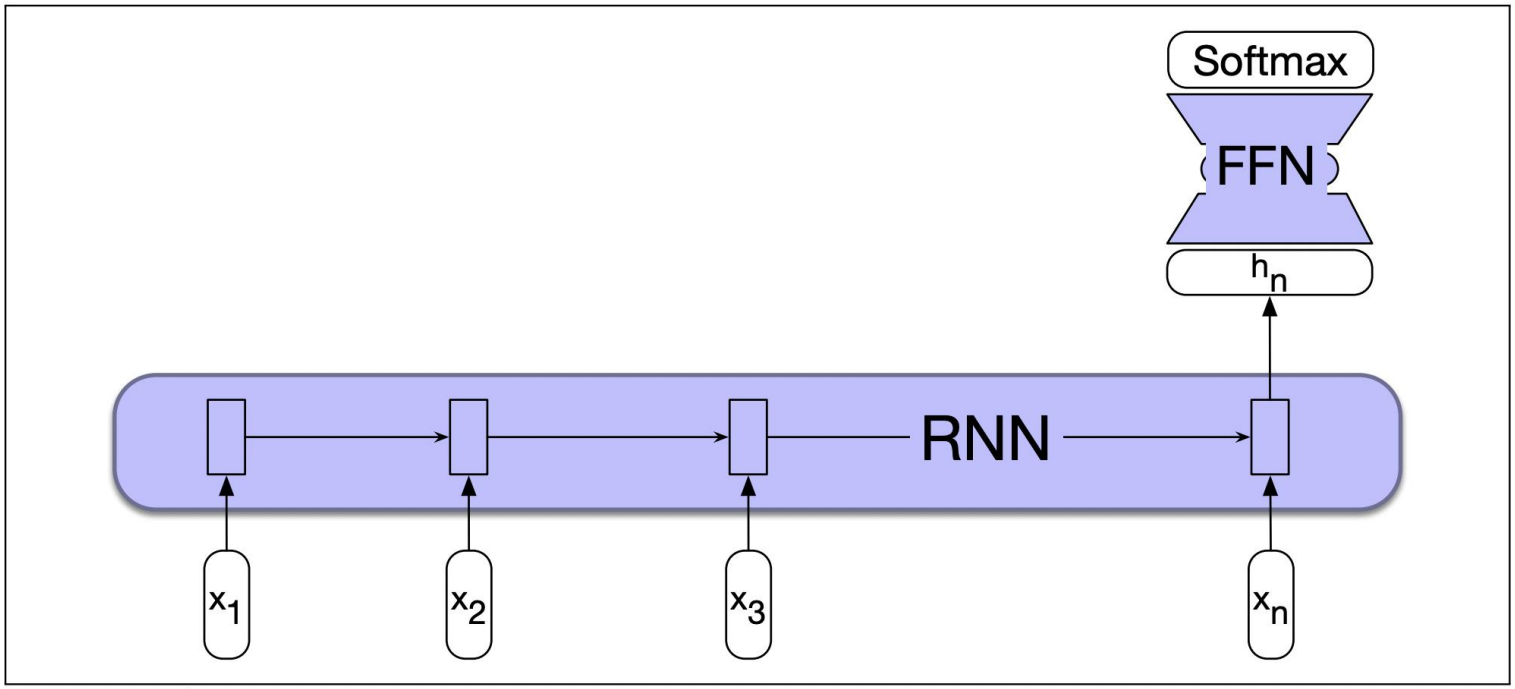
RNN có thể được sử dụng để gắn thẻ từ loại

- Ví dụ: Gán nhãn từ loại (POS tagging), nhận dạng thực thể tên



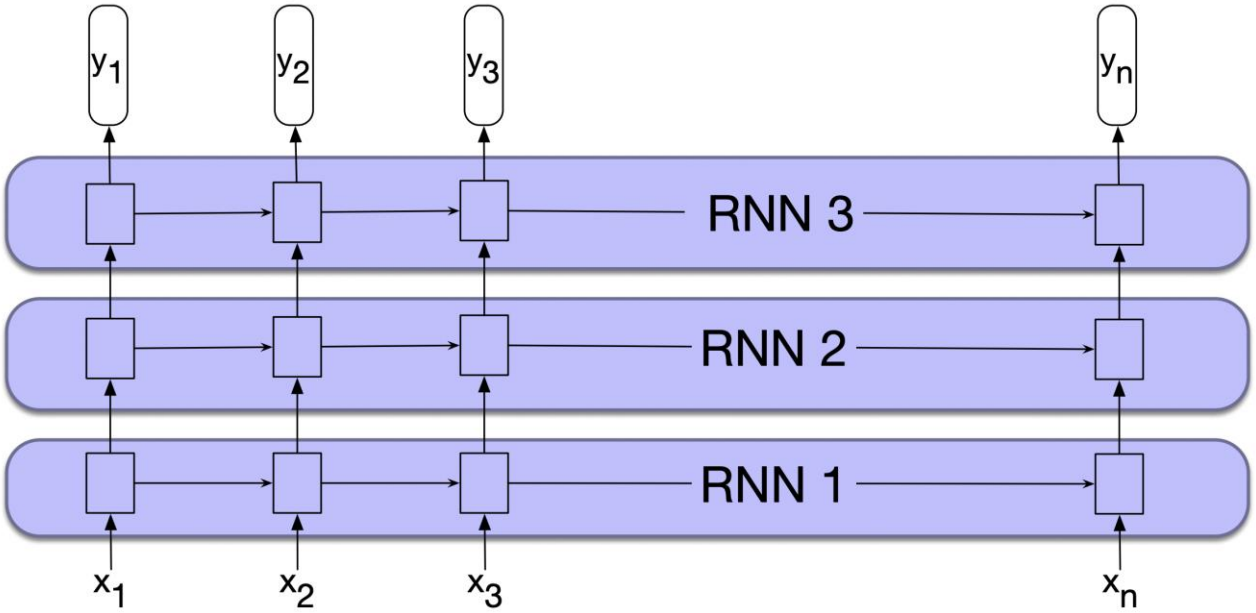
RNN có thể được sử dụng để phân loại câu

- Ví dụ: Phân loại cảm xúc, phân loại chủ đề, phát hiện thư spam…



RNN xếp chồng

- Đầu ra của mức thấp hơn đóng vai trò là đầu vào cho mức cao hơn.
- Đầu ra của mạng cuối cùng đóng vai trò là đầu ra cuối cùng.



RNN hai chiều (Bidirectional RNNs)

- Bối cảnh của mạng bên trái của thời điểm hiện tại t. (forward network)

$$h_t^f = RNN_{forward}(x_1^t)$$

- Bối cảnh của mạng bên phải của thời điểm hiện tại t. (backward network)

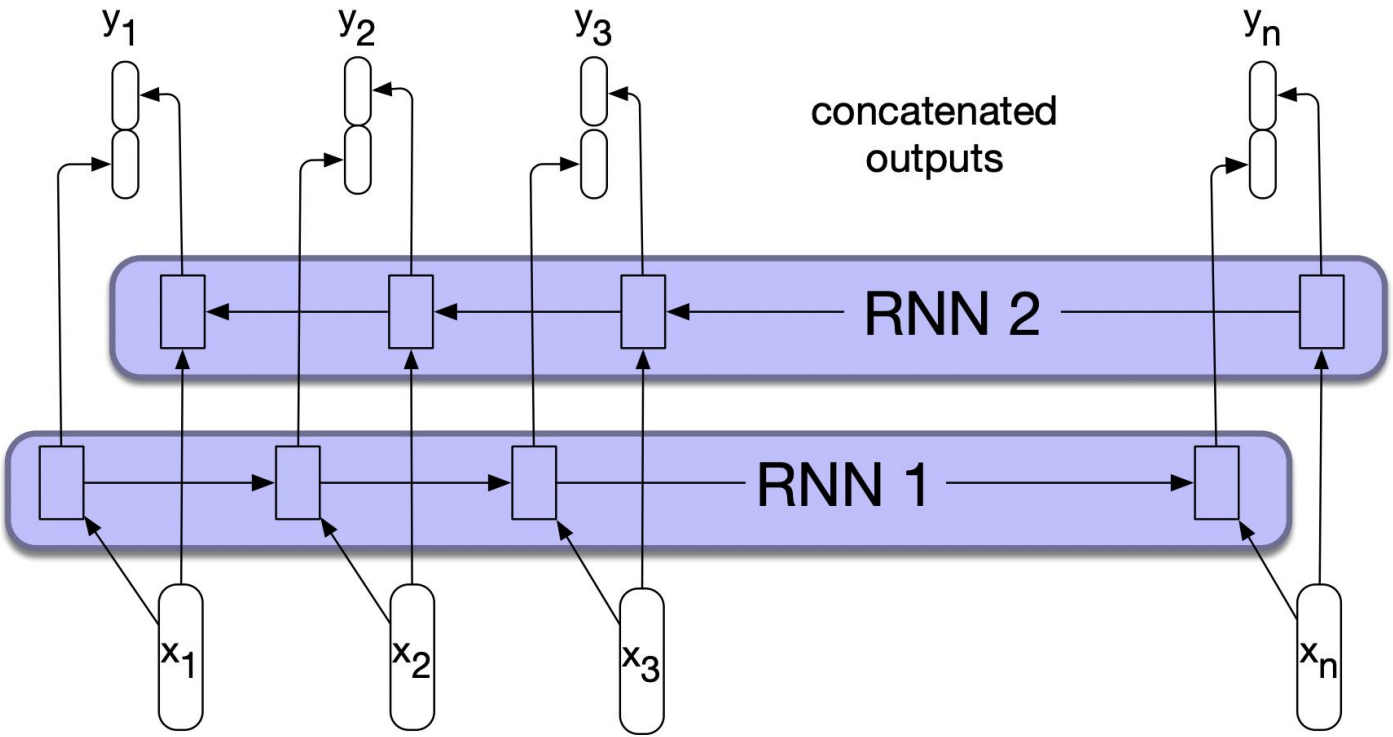
$$h_t^b = RNN_{backward}(x_t^n)$$

- Bi-RNN kết hợp hai RNN độc lập ở trên

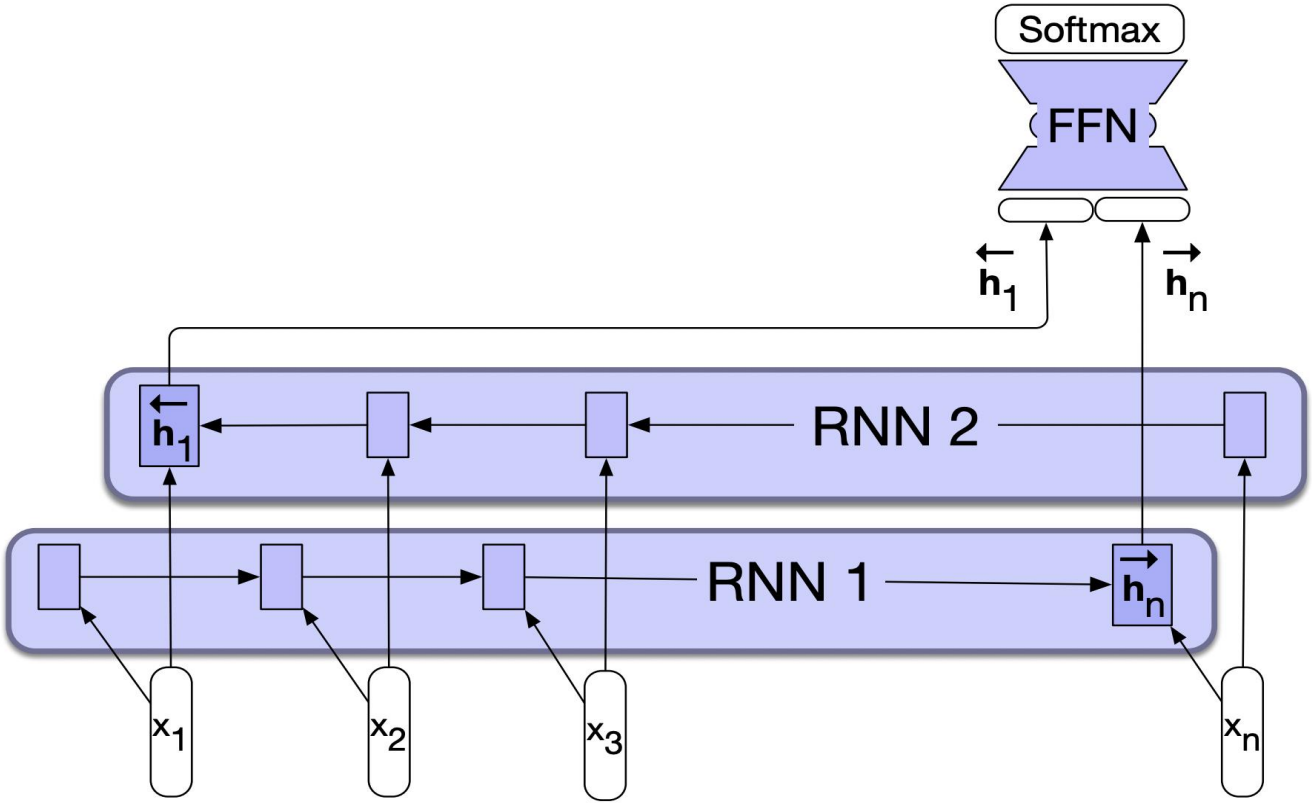
$$h_t = h_t^f \oplus h_t^b$$

RNN hai chiều (Bidirectional RNNs)

$$h_t = h_t^f \oplus h_t^b$$



RNN hai chiều RNN để phân loại chuỗi



Vấn đề phụ thuộc xa

- Xem ví dụ sau:
  - Tôi sinh ra và lớn ở Việt Nam … tôi có thể nói lưu loát tiếng Việt
  - The flights the airline was cancelling were full

Vấn đề của RNN

- Giá trị lớp ẩn tại thời điểm t chỉ ảnh hưởng tới lớp ẩn hiện tại và lớp kế tiếp.
- Vấn đề Vanishing gradient: Vì lỗi được lan truyền trở lại qua các lớp ẩn và phải chịu các phép tính nhân lặp, chúng có xu hướng về 0

Quản lý ngữ cảnh trong RNN

LSTM và GRU

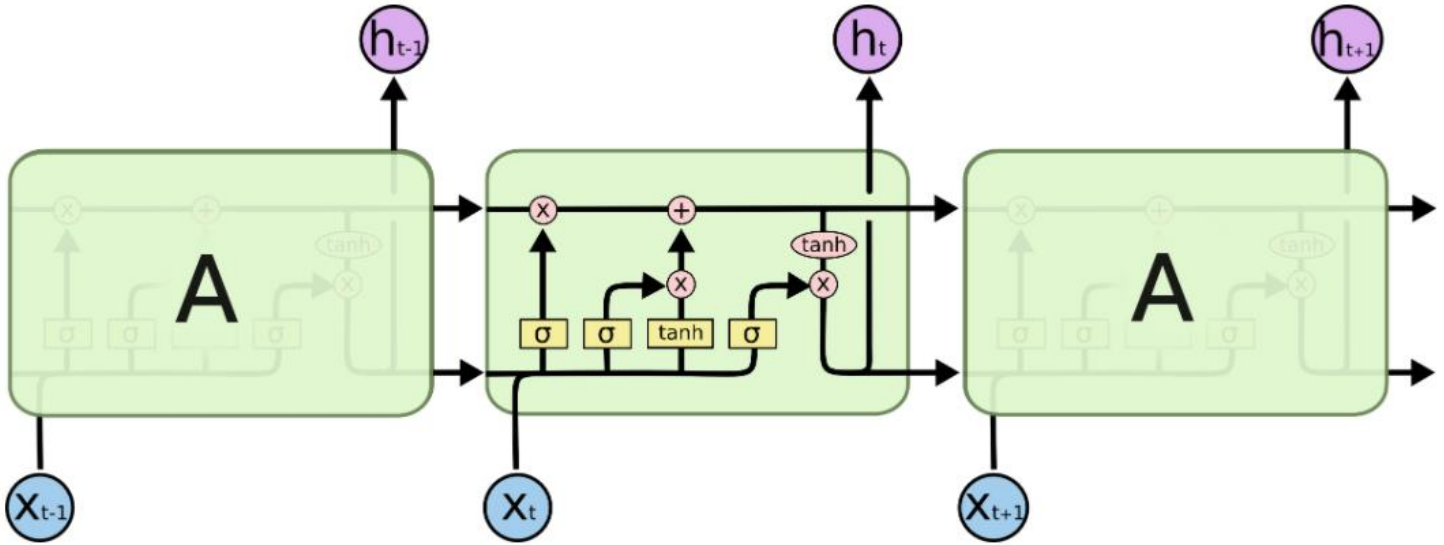
Long Short-Term Memory

(Hochreiter and Schmidhuber, 1997)

- Được thiết kế để giải quyết vấn đề Vanishing gradient.

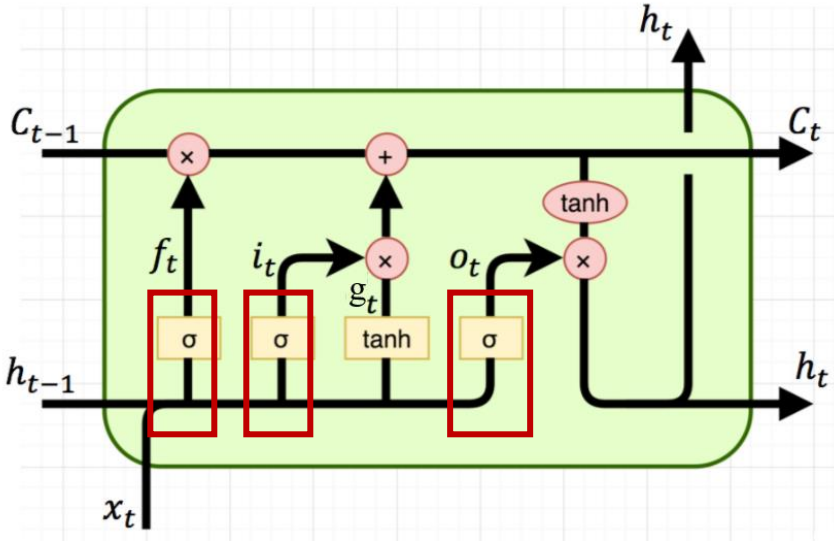
- Ý tưởng cơ bản: tách vector s được truyền giữa các bước thời gian thành một thành phần bộ nhớ (ngữ cảnh) và một thành phần trạng thái ẩn.

Long Short-Term Memory



Các cổng

- Các cổng LSTM kiểm soát luồng thông tin



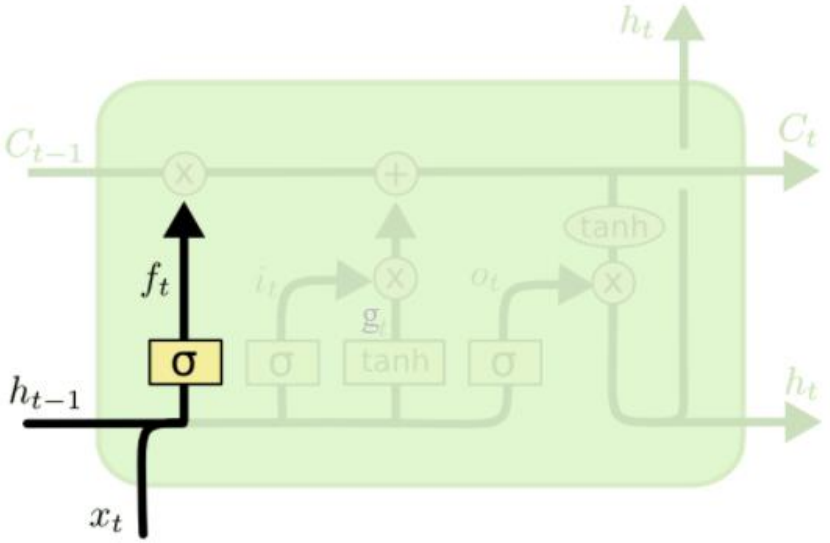
- Một sigmoid làm giảm giá trị đầu vào của nó thành từ 0 đến 1

- Bằng cách nhân đầu ra của một phần tử sigmoid với một vectơ khác, chúng ta quên thông tin trong vectơ (nếu nhân với 0) hoặc cho phép nó vượt qua (nếu nhân với 1)

Bước 1 - Cổng quên (forget gate)

- Mục đích của cổng này để xóa thông tin khỏi ngữ cảnh không còn cần thiết.

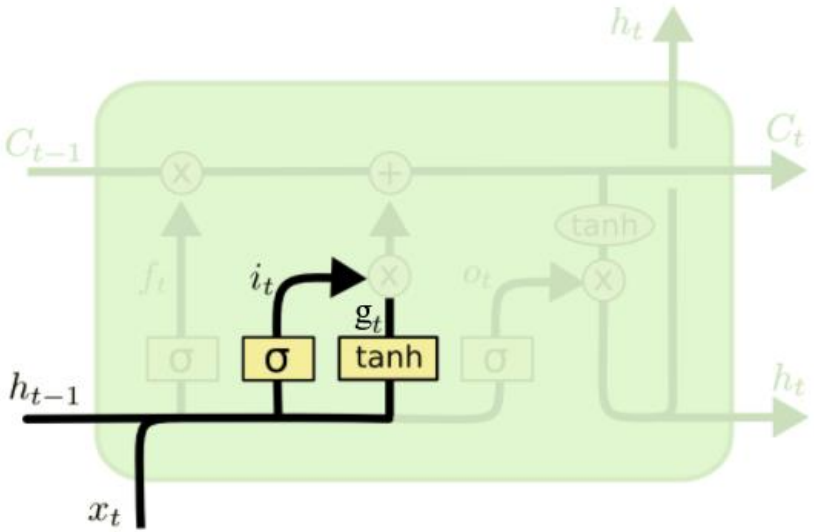
$$f_t = \sigma(U_f h_{t-1} + W_f x_t)$$



Bước 2 - Cổng đầu vào (input gate)

- Mục đích của cổng này để chọn thông tin để thêm vào ngữ cảnh hiện tại

$$i_t = \sigma(U_i h_{t-1} + W_i x_t)$$

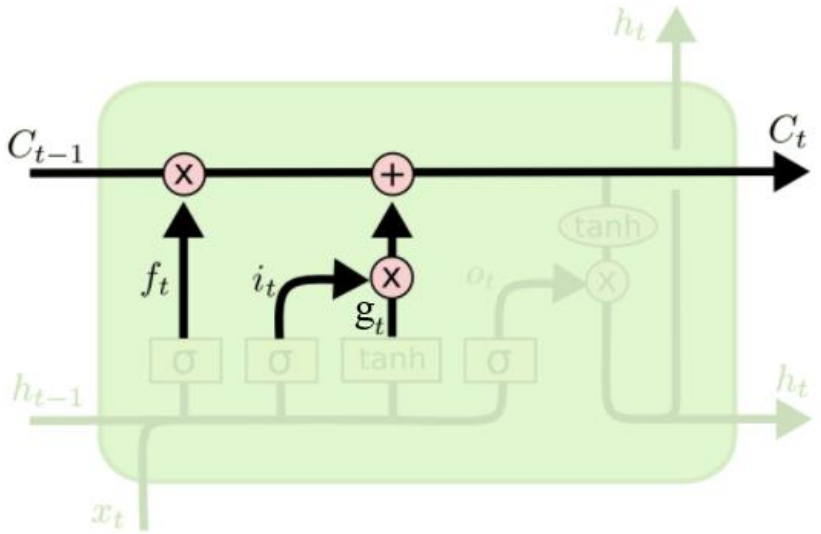


- Tính toán thông tin thực tế cần trích xuất từ trạng thái ẩn trước đo và đầu vào hiện tại

$$g_t = tanh(U_g h_{t-1} + W_g x_t)$$

Bước 3 - cập nhật C_{t-1} sang trạng thái mới C_t

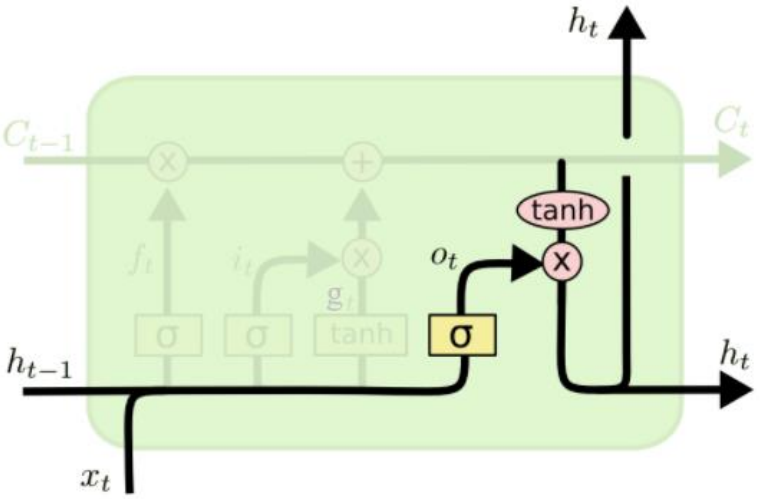
$$c_t = c_{t-1} \odot f_t + g_t \odot i_t$$



Bước 4 - Cổng đầu ra (output gate)

- Cổng đầu ra được sử dụng để quyết định thông tin nào là cần thiết cho trạng thái ẩn hiện tại.

$$o_t = \sigma(U_o h_{t-1} + W_o x_t)$$
$$h_t = o_t \odot tanh(c_t)$$



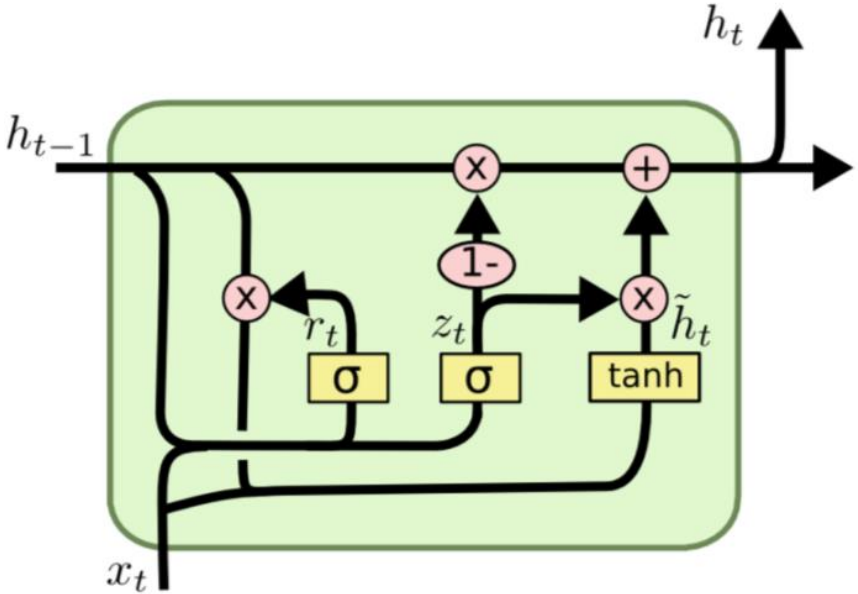
Gated Recurrent Units (GRU)

- Cổng reset **r**: Quyết định khía cạnh nào của lớp ẩn trước đó có liên quan đến bối cảnh hiện tại và những gì có thể bỏ qua.

$$r_t = \sigma(U_r h_{t-1} + W_r x_t)$$

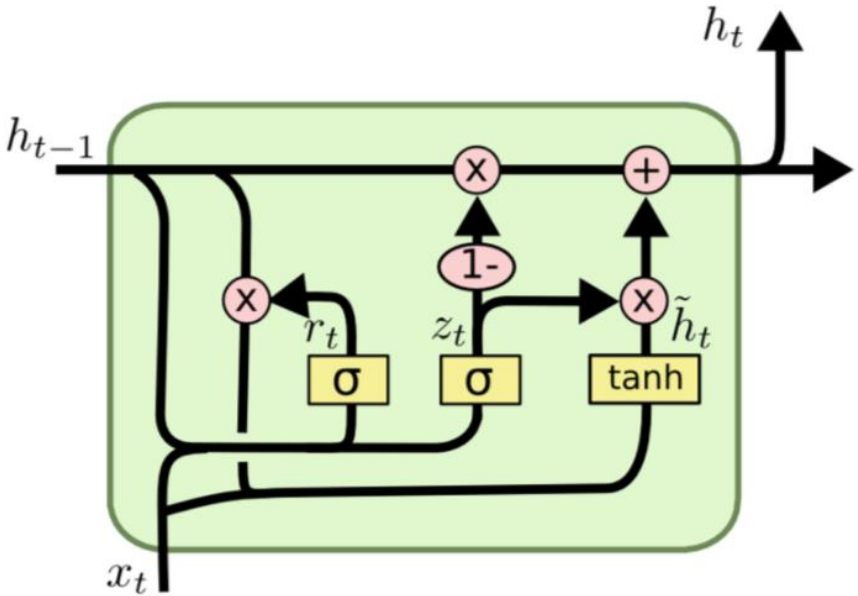
- Trạng thái lớp ẩn mới

$$\tilde{h}_t = tanh(U(r_t \odot h_{t-1}) + W x_t)$$



Gated Recurrent Units (GRU) (Cho et al., 2014)

- Cổng update **z**: xác định khía cạnh nào của trạng thái mới này $\tilde{h}_t$ sẽ được sử dụng trực tiếp trong trạng thái ẩn mới.
- Xác khía cạnh nào của trạng thái trước đó cần được giữ lại sử dụng trong tương lai.



$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t$$

Self-Attention Networks:

Transformers

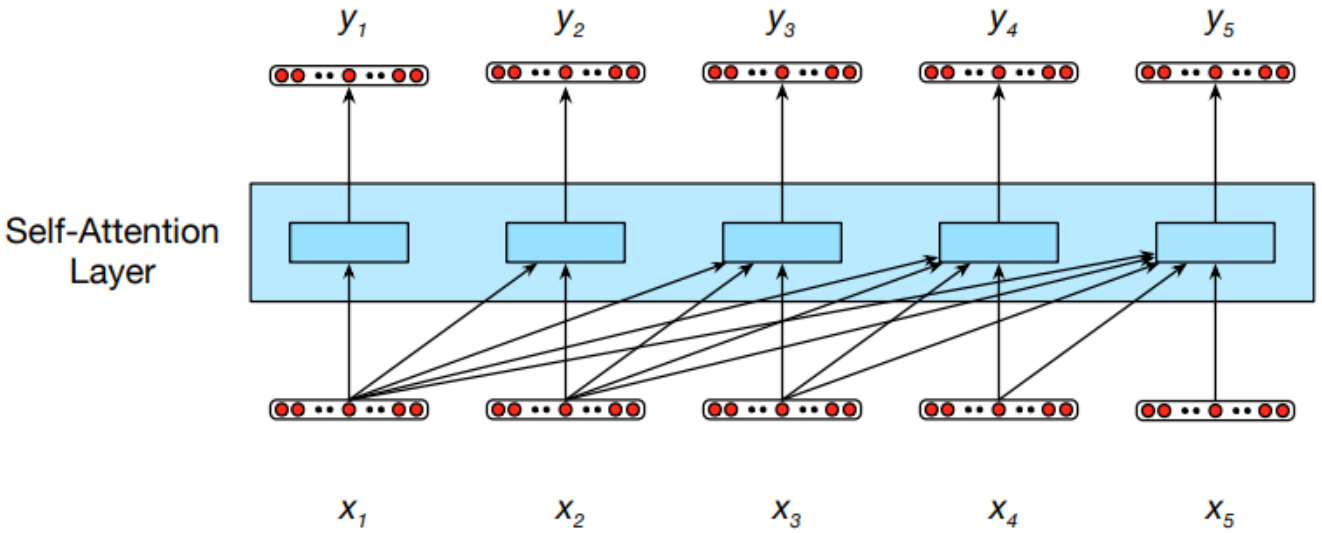
Tại sao Transformers xuất hiện?

- LSTM vẫn còn bị mất mát thông tin thông qua một loạt các kết nối lặp lại.

- Vì tính chất tuần tự nên không sử dụng được các nguồn tài nguyên song song.

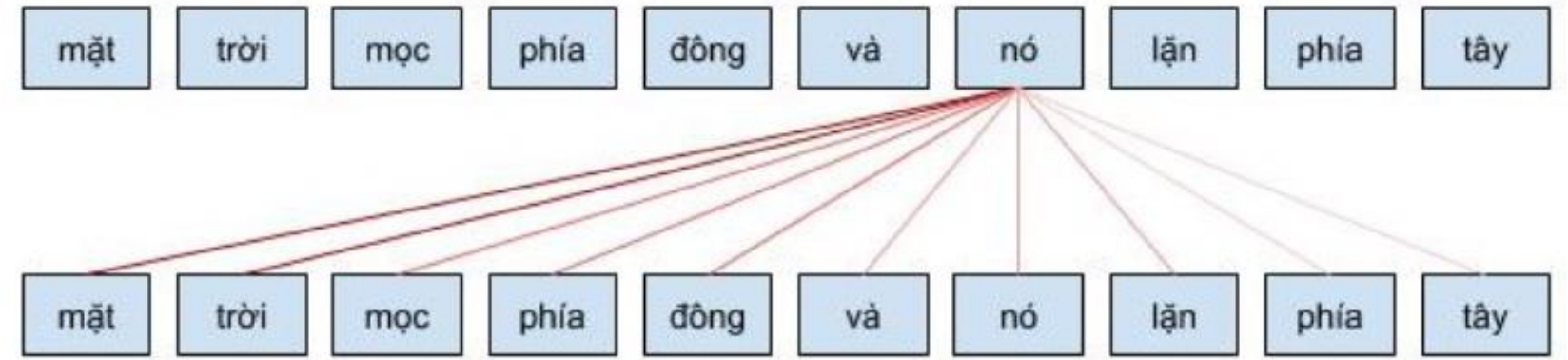
Self-Attention Layer (Vaswani et al., 2017)

- Luồng thông tin trong mô hình self-attention
- Khi xử lý từng phần tử, mô hình chú ý đến tất cả các đầu vào và bao gồm cả đầu vào hiện tại.



Self-Attention Layer

- Self Attention cho phép mô hình khi mã hóa một từ có thể sử dụng thông tin của những từ liên quan tới nó



Self-Attention Layer

$$y_i = \sum_{j \le i} \alpha_{ij} v_j$$

$$\alpha_{ij} = \text{softmax}(score(x_i, x_j)) \;\; \forall j \le i$$

$$= \frac{\exp(score(x_i, x_j))}{\sum_{k=1}^{i} \exp(score(x_i, x_k))} \;\; \forall j \le i$$

$$score(x_i, x_j) = q_i \cdot k_j$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

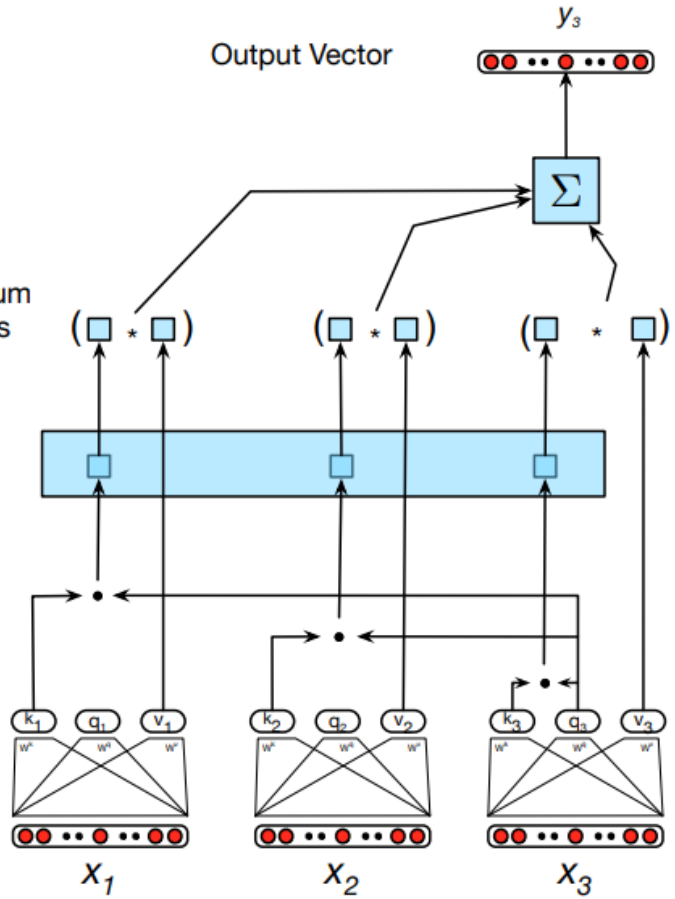
$$q_i = W^Q x_i; \;\; k_i = W^K x_i; \;\; v_i = W^V x_i$$

Weight and Sum value vectors

Softmax $\alpha_{i,j}$

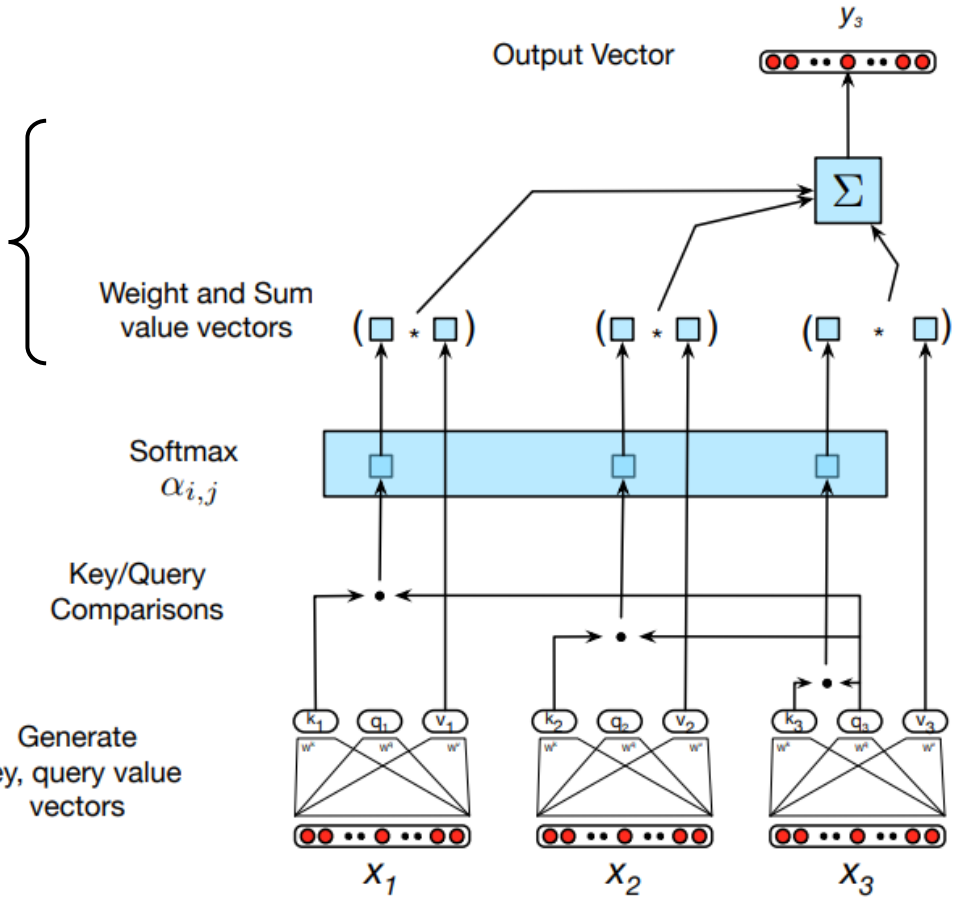
Key/Query Comparisons

Generate key, query value vectors



Self-Attention Layer

$$SelfAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}\right)\mathbf{V}$$



$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

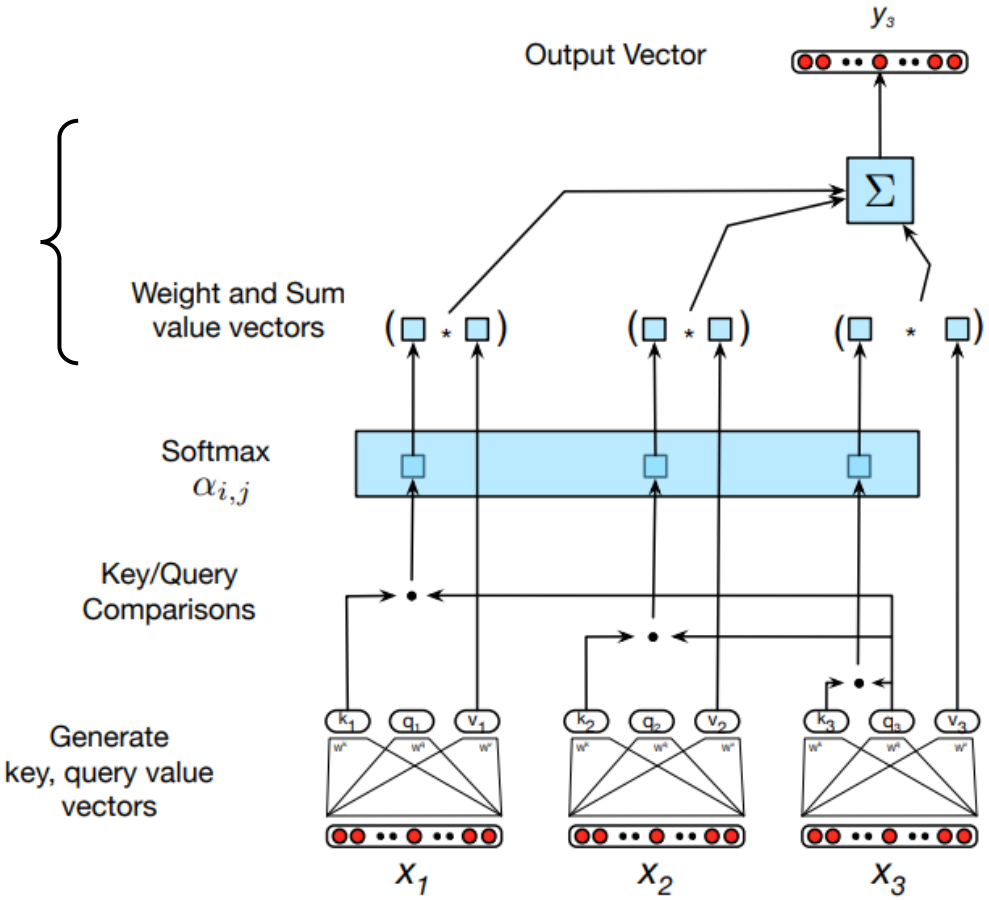
Generate key, query value vectors

Self-Attention Layer

N

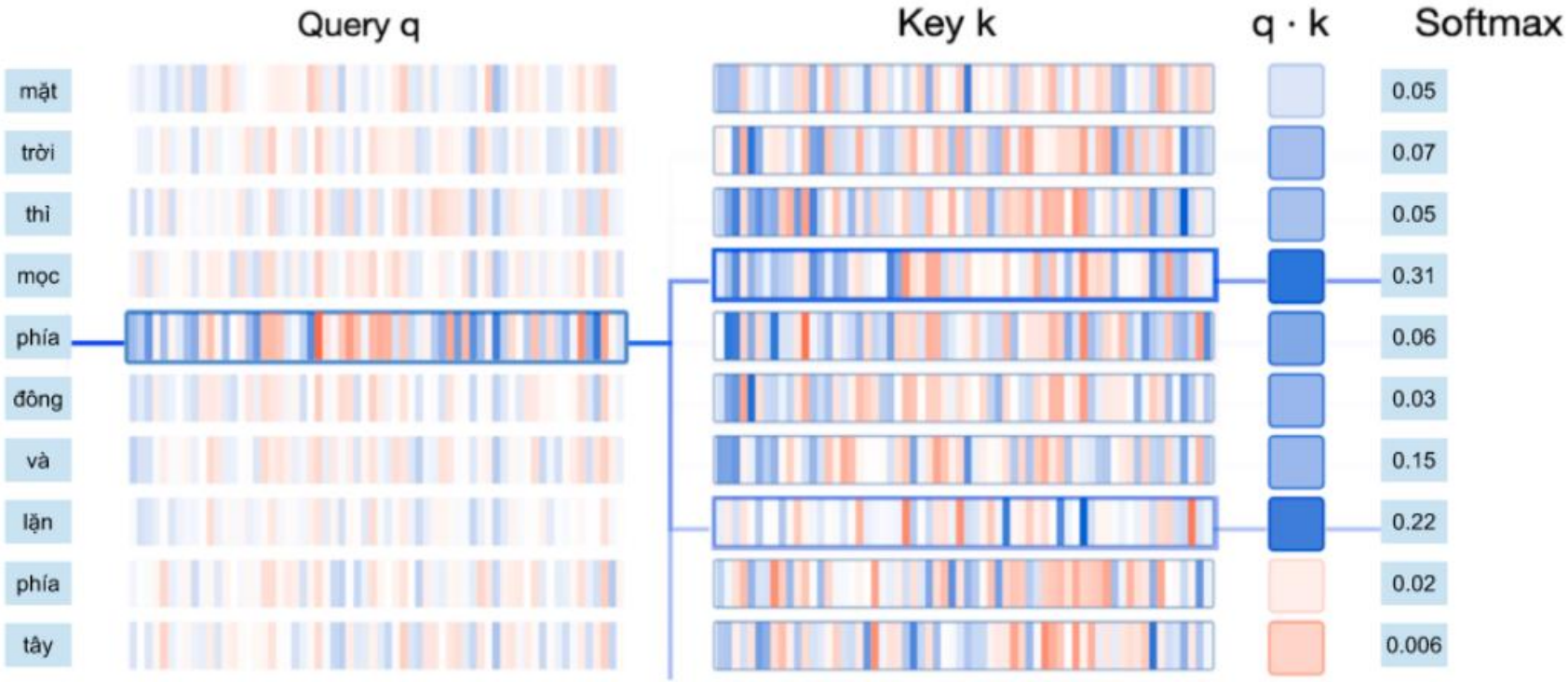
q1•k1	−∞	−∞	−∞	−∞
q2•k1	q2•k2	−∞	−∞	−∞
q3•k1	q3•k2	q3•k3	−∞	−∞
q4•k1	q4•k2	q4•k3	q4•k4	−∞
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N



Self-Attention Layer

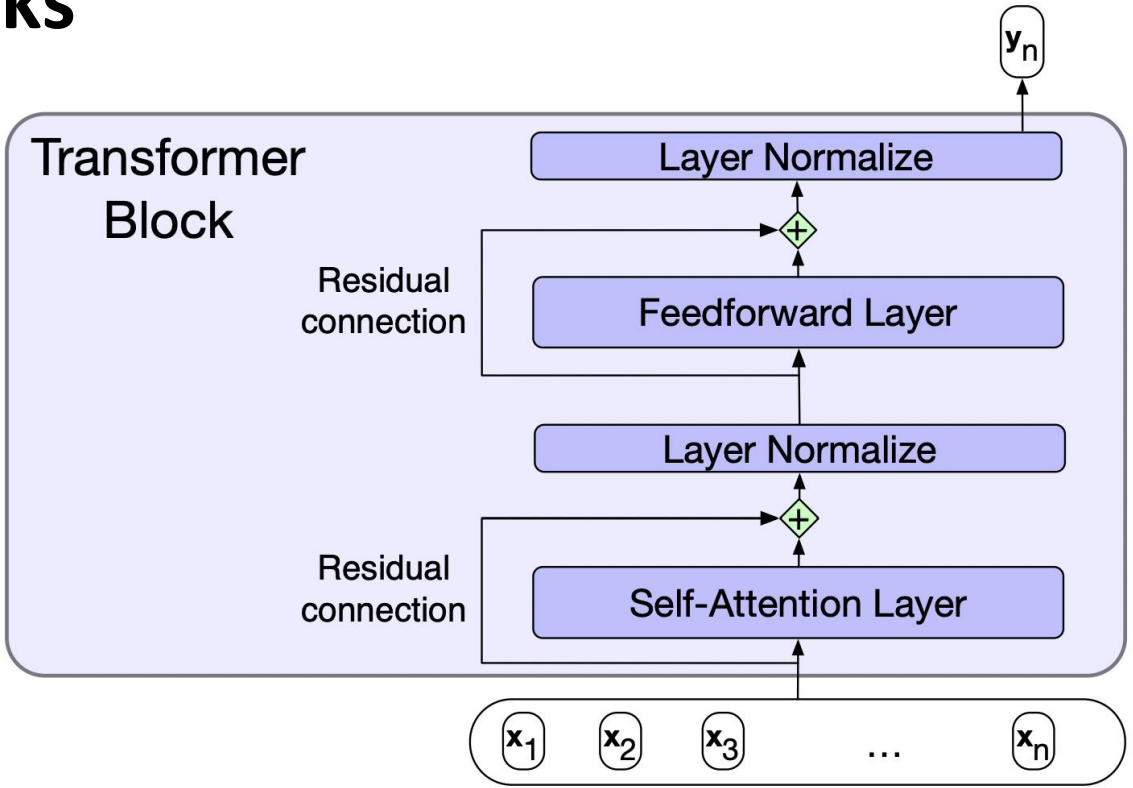
- Ví dụ:



Transformer Blocks

$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFNN}(\mathbf{z}))$$

$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{SelfAttn}(\mathbf{x}))$$



Multihead Attention

Giảm nó xuống kích thước đầu ra ban đầu.

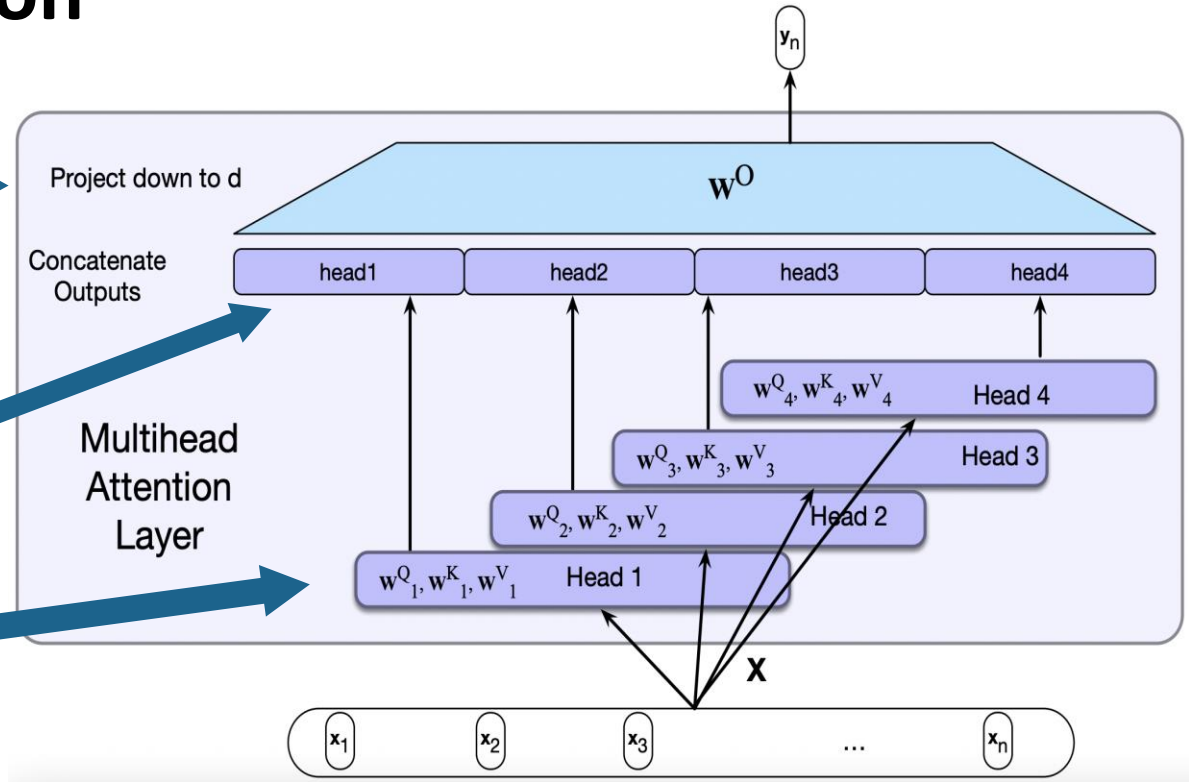
$$MultiHeadAttn(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q \; ; \; \mathbf{K} = \mathbf{X}\mathbf{W}_i^K \; ; \; \mathbf{V} = \mathbf{X}\mathbf{W}_i^V$$

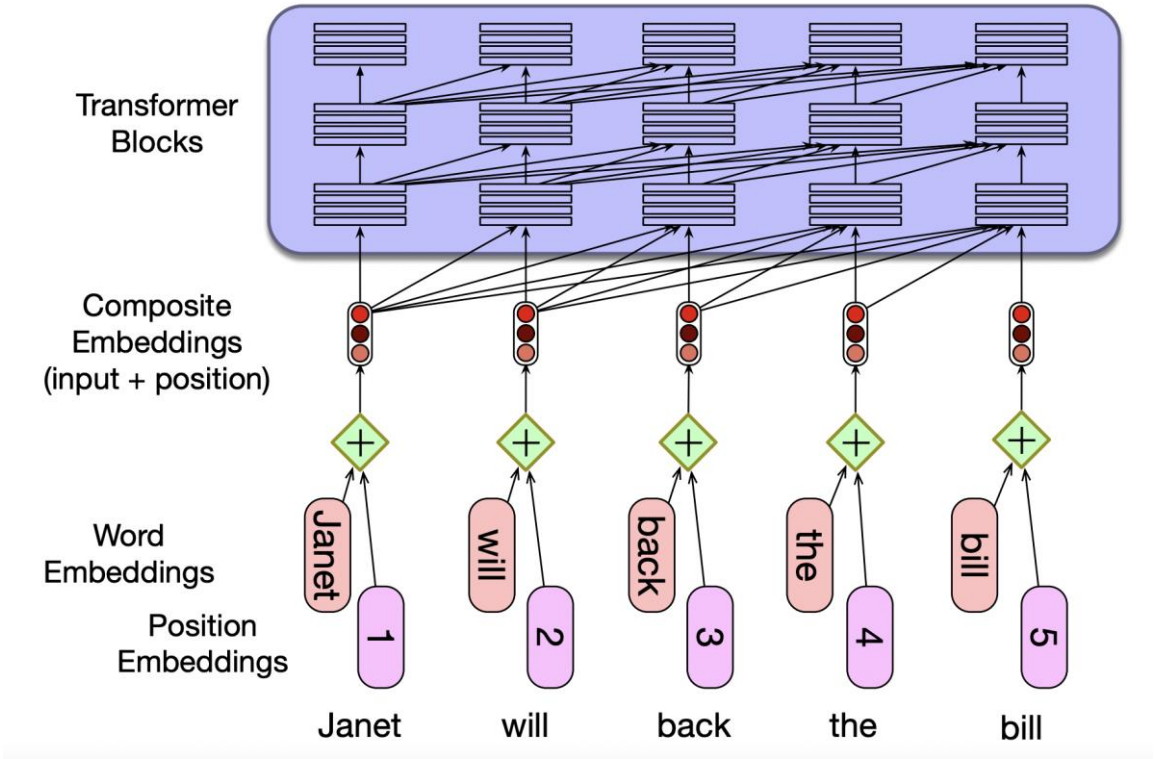
$$\mathbf{head}_i = SelfAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$

Nối các đầu ra từ mỗi self-attention

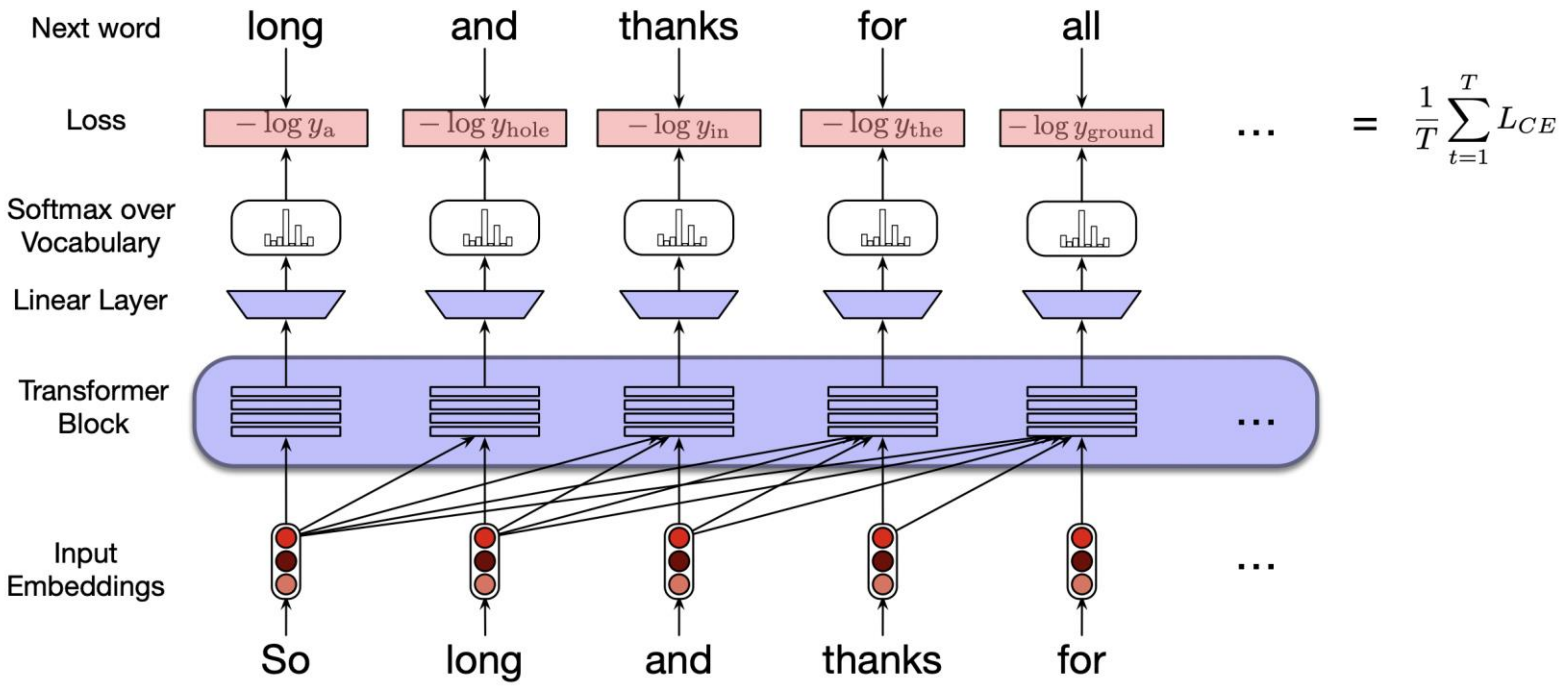
Với mỗi self-attention được cung cấp một bộ ma trận query, key, value riêng



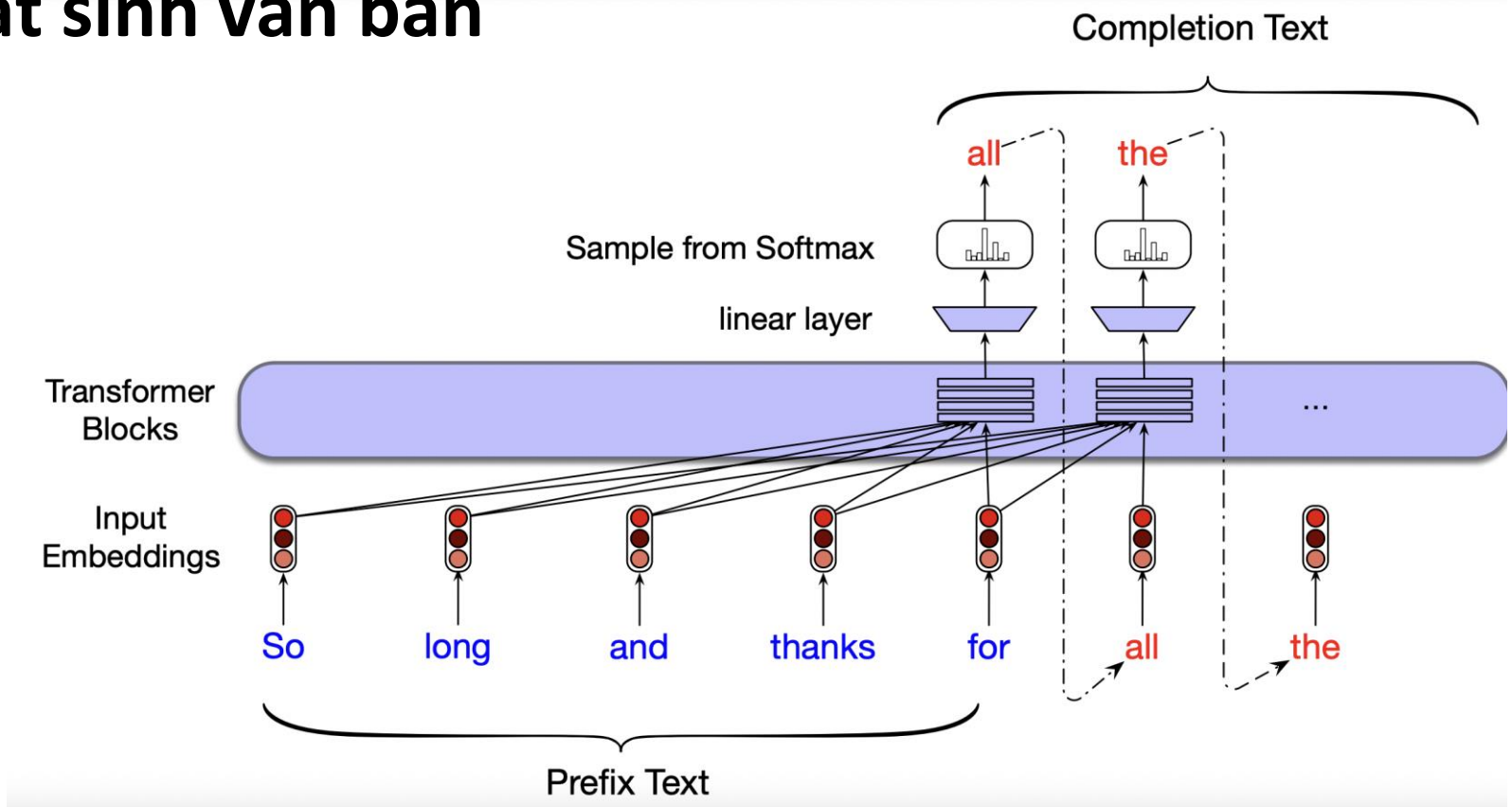
Positional Embeddings



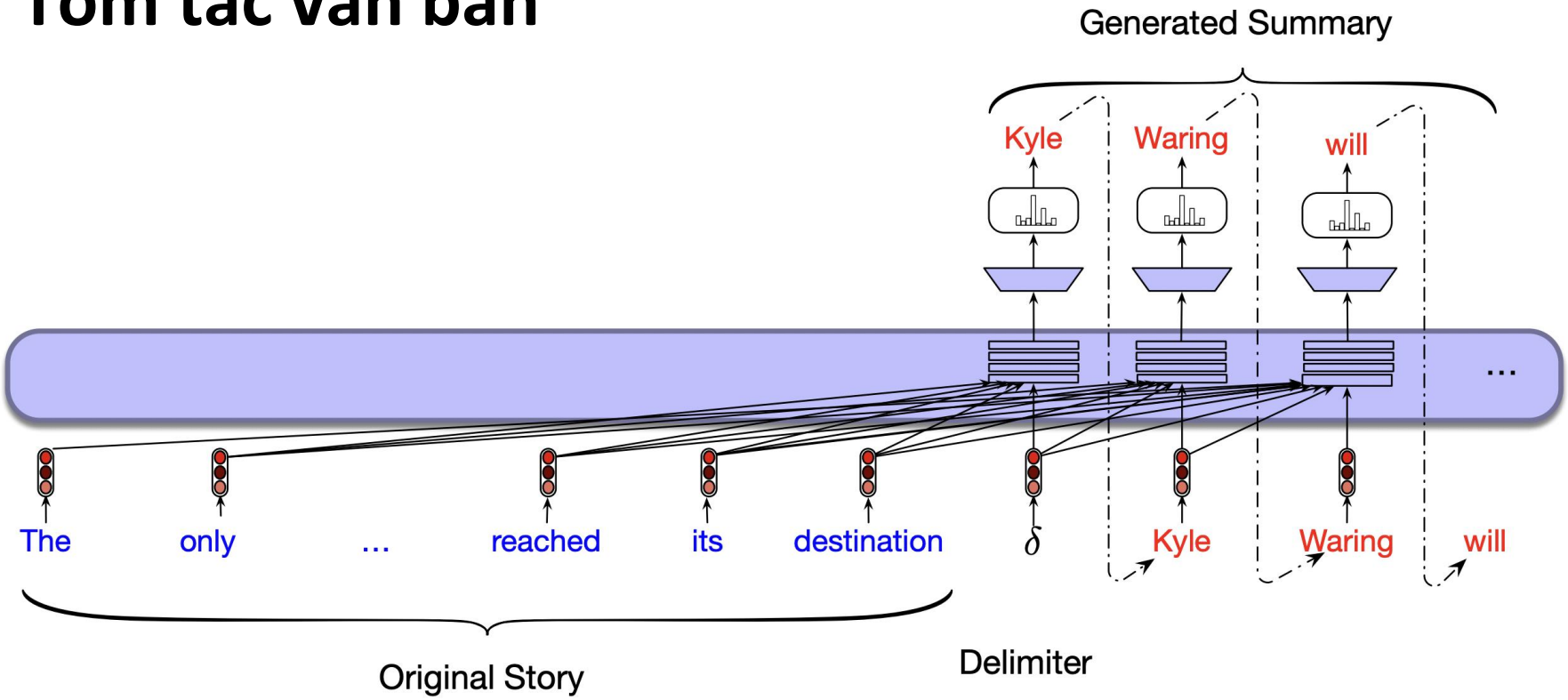
Transformers as Autoregressive Language Models



Phát sinh văn bản



Tóm tắt văn bản



Tác hại từ các mô hình ngôn ngữ

- Các mô hình ngôn ngữ có thể tạo ra ngôn ngữ độc hại
- Các mô hình ngôn ngữ có thể là một công cụ để tạo ra văn bản có thông tin sai lệch, lừa đảo, cực đoan hóa và các hoạt động có hại cho xã hội khác.
- Những vấn đề quan trọng về quyền riêng tư

Tổng kết

- Khái niệm về RNN và các cách áp dụng vào vấn đề ngôn ngữ.
  - Đầu ra của một neural unit tại một thời điểm cụ thể dựa trên cả giá trị đầu vào hiện tại và giá trị của lớp ẩn từ bước thời gian trước đó.
    - Mô hình ngôn ngữ xác suất
    - Auto-regressive generation
    - Gán nhãn chuỗi (Sequence labeling)
    - Phân loại văn bản
- Kiến trúc mạng LSTM và GRU.
- Mô hình Transformers với lớp Self-Attention.