Word2Vec

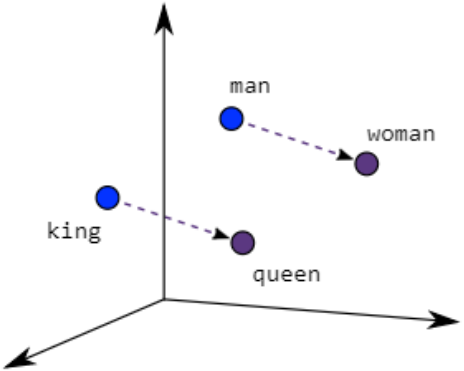
20424008 - Dương Mạnh Cường

Đại học Khoa Học Tự Nhiên
ĐHQG Thành phố Hồ Chí Minh

14/02/2022

Nội dung

Word2Vec model (1)

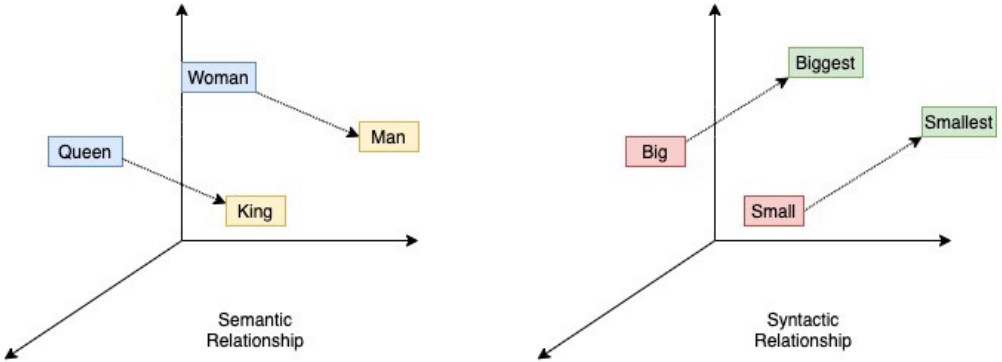


Hình 1: Các từ có ý nghĩa tương đồng nhau nằm gần nhau.

- Word2Vec là một phương pháp biến đổi dữ liệu dạng **text** sang **numeric**.
- Các từ được đại diện bằng các **vector**.
- Word2Vec model sử dụng **neural network** nên yêu cầu input data phải là numeric.
- Khắc phục được nhược điểm của các phương pháp truyền thống như TF-IDF, Bag of Words vì chúng không hiểu được **syntactic** và **semantic** của từ.

Word2Vec model (2)

- Word2Vec là một phương pháp **word embedding** được sử dụng phổ biến.
- Các **word vector** được tạo ra bởi Word2Vec model có khả năng nắm bắt được **semantic** và **syntactic** của từ.



Hình 2: Ví dụ về **semantic** và **syntactic**.

Word2Vec model (3)

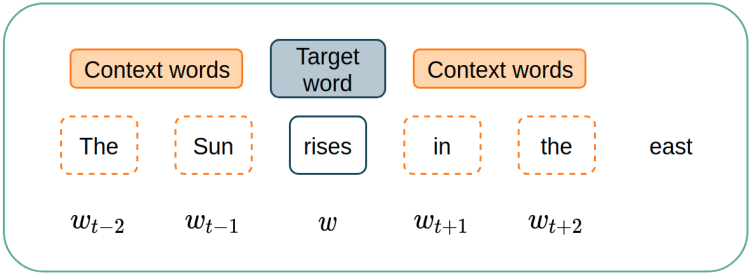
- Word2Vec biểu diễn các word vector trong không gian m chiều.
- Ví dụ có câu: *"Archie used to live in New York, he then moved to Santa Clara. He loves apples and strawberries."*.

Hình 3: Các từ trong câu ví dụ được biểu diễn trong không gian vector.

Word2Vec model (4)

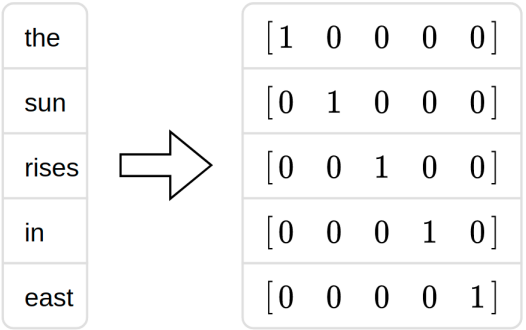
- Vì hiểu được semantic và syntactic của từ điều này giúp tận dụng các vector vào các bài toán như **text summarization** *(tóm tắt văn bản)*, **sentiment analysis** *(phân tích tình cảm)*, **text generation** *(tạo văn bản)*,...
- Có hai cách để xây dựng Word2Vec model:
  1. **CBOW model**.
  2. **Skip-gram model**.

- Giả sử có một neural network bao gồm: **một input layer**, **một hidden layer** và **một output layer**. Cần dự đoán ra **một từ** dựa vào **các từ xung quanh nó**. Từ được dự đoán được gọi là **target word** và các từ xung quanh nó được gọi là **context word**.
- Đặt n là số context word cần thiết để dự đoán target word. Xét $n = 2$ với câu: *"The Sun rises in the east."*.



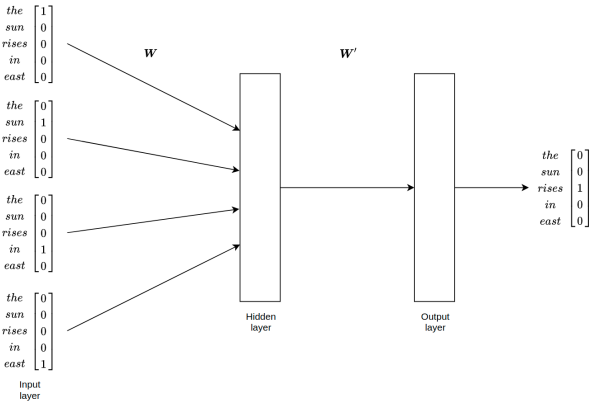
Hình 4: Target word và context word trong CBOW model.

- Input của network là các context word và output của network là target word.
- Cần sử dụng kỹ thuật **one-hot encoding** để chuyển đổi các text data thành numeric data trước khi đưa vào network.



Hình 5: One-hot encoding cho text data.

- Kiến trúc của CBOW model. Các context word là: *the*, *sun*, *in* và *east* được dùng làm đầu vào cho network và target word là *rises* được dự đoán ở đầu ra.



Hình 6: Kiến trúc của CBOW model 1.

Word2Vec model (8)

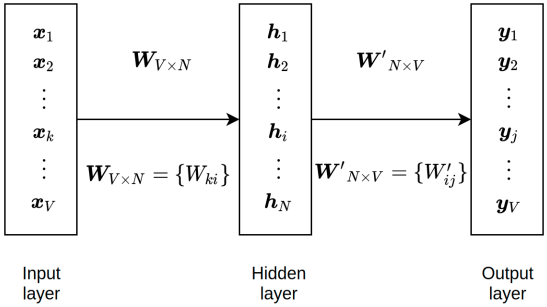
CBOW model

- Kết thúc quá trình đào tạo, lấy weight $\boldsymbol{W}$ ra làm các word vector cho các vocabulary.
- Dưới đây là các vector tương ứng cho các từ của $\boldsymbol{W}$. Word embedding tương ứng cho từ *sun* là $\begin{bmatrix}0.0 & 0.3 & 0.3 & 0.6 & 0.1\end{bmatrix}$.

$$\boldsymbol{W} = \begin{array}{c} \\ the \\ sun \\ rises \\ in \\ east \end{array} \begin{bmatrix} \\ 0.01 & 0.02 & 0.1 & 0.5 & 0.37 \\ 0.0 & 0.3 & 0.3 & 0.6 & 0.1 \\ 0.4 & 0.34 & 0.11 & 0.61 & 0.43 \\ 0.1 & 0.11 & 0.1 & 0.17 & 0.369 \\ 0.33 & 0.4 & 0.3 & 0.17 & 0.1 \end{bmatrix}$$

- Xét trường hợp chỉ sử dụng duy nhất một context word, tức $C = 1$. Lúc này network nhận vào một context word ở đầu vào và trả về một target word ở đầu ra.
- Xét câu: *"The Sun rises in the east."*.
- Đặt V là số lượng vocabulary và N là số neuron của hidden layer. Vì neural network của chúng ta có ba layer, cụ thể:
  - Input layer được đại diện bởi $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_k, ..., \boldsymbol{x}_V\}$. Khi chúng ta nói $\boldsymbol{x}_k$, tức ta đang đề cập đến từ thứ k trong corpus của chúng ta.
  - Hidden layer được đại diện bởi $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, ..., \boldsymbol{h}_i, ..., \boldsymbol{h}_N\}$. Khi chúng ta nói $\boldsymbol{h}_i$, tức ta đang đề cập đến neuron thứ i trong hidden layer.
  - Output layer được đại diện bởi $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, ..., \boldsymbol{y}_j, ..., \boldsymbol{y}_V\}$. Khi chúng ta nói $\boldsymbol{y}_j$, tức ta đang đề cập đến từ thứ j trong output layer.

- Số chiều của $\boldsymbol{W}$ là $V \times N$, tức *số vocabulary* $\times$ *số neuron trong hidden layer*. W_{ki} đại diện cho một phần tử trong ma trận $\boldsymbol{W}$ giữa $\boldsymbol{x}_k$ trong input layer và $\boldsymbol{h}_i$ trong hidden layer.
- Số chiều của $\boldsymbol{W'}$ là $N \times V$, tức *số neuron trong hidden layer* $\times$ *số vocabulary*. W'_{ij} đại diện cho một phần tử trong ma trận $\boldsymbol{W'}$ giữa $\boldsymbol{h}_i$ trong hidden layer và $\boldsymbol{y}_j$ trong output layer.



Hình 7: Kiến trúc của CBOW model 2.

Forward propagation

- **Bước 1**:

$$\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}^T = \boldsymbol{W}_{(k,.)}$$

$\boldsymbol{W}_{(k,.)}$ là biểu diễn vector của input word. Đặt vector đại diện cho input word w_I bằng Z_{w_I}. Lúc này phương trình trên có thể được viết thành.

$$\boldsymbol{H} = Z_{w_I} \tag{1}$$

Forward propagation

- **Bước 2**: Đặt $\boldsymbol{u}_j$ là điểm số để từ thứ j trong corpus là target word - được tính bằng cách nhân $\boldsymbol{H}$ cho $\boldsymbol{W'}$.

$$\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$$

- **Bước 3**: Cột j của $\boldsymbol{W'}_{ij}$ đại diện cho vector của từ j trong corpus. Đặt vector đại diện cho từ thứ j là $Z'_{w_j^T}$

$$\boldsymbol{u}_j = Z'_{w_j^T} \cdot \boldsymbol{H} \tag{2}$$

Forward propagation

- **Bước 4**: Thế phương trình (1) vào phương trình (2), ta được:

$$\boldsymbol{u}_j = Z'^{T}_{w_j} \cdot Z_{w_I}$$

- **Bước 5**: Áp dụng softmax activation để chuyển đổi $\boldsymbol{u}_j$ sang xác suất:

$$\boldsymbol{y}_j = \frac{\exp(\boldsymbol{u}_j)}{\sum_{j'=1}^{V} \exp(\boldsymbol{u}'_j)} \tag{3}$$

Forward propagation

- **Bước 6**: Đặt $\boldsymbol{y}_j^*$ là xác suất để từ j chính xác là target word. Và vì chúng ta áp dụng gradient descent để tìm ra optimal weight nên thay vì tối đa hóa ta sẽ tìm tối thiểu hóa của $\log(\boldsymbol{y}_j^*)$

$$\min(-\log(\boldsymbol{y}_j^*))$$

- **Bước 7**: Loss function của chúng ta lúc này thành:

$$\begin{aligned}\mathfrak{L} &= -\log(\boldsymbol{y}_j^*) \\ &= -\log\left(\frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_j')}\right)\end{aligned}$$

Forward propagation *(tiếp theo)*

$$\begin{aligned}\mathfrak{L} &= -\left(\log\left(\exp\left(u_j\right)\right) - \log\left(\sum_{j'=1}^{V}\exp\left(u'_j\right)\right)\right) \\ &= -\log\left(\exp\left(u_j\right)\right) + \log\left(\sum_{j'=1}^{V}\exp\left(u'_j\right)\right) \\ &= -u_j + \log\left(\sum_{j'=1}^{V}\exp\left(u'_j\right)\right)\end{aligned}$$

Backward propagation

- **Bước 1**: Trong quá trình này, chúng ta sẽ cập nhật cho W và W' như sau:

$$W = W - \alpha \frac{\partial \mathfrak{L}}{\partial W}$$

$$W' = W' - \alpha \frac{\partial \mathfrak{L}}{\partial W'}$$

Backward propagation

- **Bước 2**: Nhắc lại các công thức trong quá trình forward propagation:

$$\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}^T$$

$$\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$$

$$\mathfrak{L} = -\boldsymbol{u}_j + \log\left(\sum_{j'=1}^{V} \exp\left(\boldsymbol{u'}_j\right)\right)$$

Backward propagation

- **Bước 3**: Tính gradient của loss function đối với $\boldsymbol{W'}$ theo quy tắc **chain rule**.

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W'}_{ij}} = \frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} \cdot \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}}$$

Ta có đạo hàm của $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j}$ như sau: $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} = \boldsymbol{e}_j$ (5)

với $\boldsymbol{e}_j$ là sự khác biệt giữa actual word và predicted word.

Backward propagation

- **Bước 4**: Tính đạo hàm cho $\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}}$, và bởi vì chúng ta biết $\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$ nên:

$$\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}} = \boldsymbol{H}$$

- **Bước 5**: Như vậy, gradient của loss function đối với $\boldsymbol{W'}$ bằng:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W'}_{ij}} = \boldsymbol{e}_j \cdot \boldsymbol{H}$$

Backward propagation

- **Bước 6**: Tính gradient của loss function đối với $\boldsymbol{W}$ bằng quy tắc chain rule:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}_{ki}} = \frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i} \cdot \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{W}_{ki}}$$

- **Bước 7**: Tính đạo hàm của $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i}$, chúng ta áp dụng chain rule:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i} = \sum_{j=1}^{V} \frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} \cdot \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{h}_i}$$

Backward propagation

- **Bước 8**: Thế phương trình (5) vào phương trình trên, ta được:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i} = \sum_{j=1}^{V} \boldsymbol{e}_j \cdot \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{h}_i}$$

- **Bước 9**: Vì $\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$, tiếp tục thế vào phương trình trên được:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i} = \sum_{j=1}^{V} \boldsymbol{e}_j \cdot \boldsymbol{W'}_{ij} = \mathfrak{L}\boldsymbol{H}^T$$

Backward propagation

- **Bước 10**: Tính đạo hàm cho $\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{W}_{ki}}$. Biết $\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}^T$ nên:

$$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{W}_{ki}} = \boldsymbol{X}$$

- **Bước 11**: Cuối cùng, gradient của loss function đối với $\boldsymbol{W}$ là:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}_{ki}} = \boldsymbol{L}\boldsymbol{H}^T \cdot \boldsymbol{X}$$

Backward propagation

- **Bước 12**: Như vậy, các set weight $\boldsymbol{W}$ và $\boldsymbol{W'}$ được cập nhật trong quá trình backward propagation như sau:

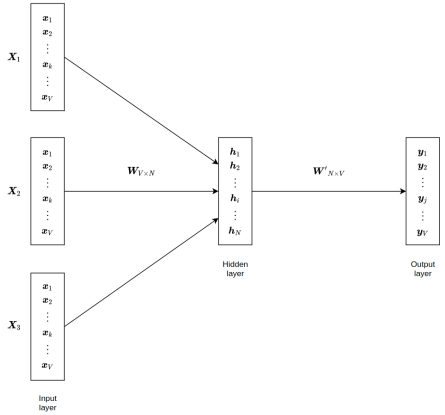
$$\boldsymbol{W} = \boldsymbol{W} - \alpha \boldsymbol{L}\boldsymbol{H}^T \cdot \boldsymbol{X}$$

$$\boldsymbol{W'} = \boldsymbol{W'} - \alpha \boldsymbol{e}_j \cdot \boldsymbol{H}$$

Mã nguồn

```
def single_context_CBOW(x, label, W1, W2, loss):
  # forward propagation
  h = np.dot(W1.T, x)
  u = np.dot(W2.T, h)
  y_pred = softmax(u)

  # error giữa actual value và predicted value
  dW2 = np.outer(h, e)
  dW1 = np.outer(x, np.dot(W2.T, e))

  # cập nhật các weight set
  W1 = W1 - lr * dW1
  W2 = W2 - lr * dW2

  # loss function
  loss += -float(u[label] == 1) + np.log(np.sum(np.exp(u)))

  return W1, W2, loss
```

- Dưới đây là kiến trúc của CBOW model với multiple context words:



Hình 8: Kiến trúc của CBOW model với multiple context words.

- Với multiple context words, chúng ta lấy giá trị trung bình của tất cả các context word đầu vào. Cụ thể với C context word $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_C$, chúng ta thực hiện:

$$\begin{aligned}\boldsymbol{H} &= \frac{(\boldsymbol{X}_1 + \boldsymbol{X}_2 + \ldots + \boldsymbol{X}_C)}{C}\boldsymbol{W}^T \\ &= \frac{1}{C}(\boldsymbol{X}_1\boldsymbol{W}^T + \boldsymbol{X}_1\boldsymbol{W}^T + \ldots + \boldsymbol{X}_C\boldsymbol{W}^T)\end{aligned}$$

- Tương tự như CBOW model với single context word, $\boldsymbol{X}_1\boldsymbol{W}^T$ đại diện cho vector của input context word w_1. Đại diện Z_{w_1} cho input context word w_1, cho nên:

$$\boldsymbol{H} = \frac{1}{C}(Z_{w_1} + Z_{w_2} + \ldots + Z_{w_C}) \tag{6}$$

- Ở đây, C đại diện cho số lượng context word, để tính toán giá trị của $\boldsymbol{u}_j$ sẽ tương tự như cách ta tính toán cho single context word.

$$\boldsymbol{u}_j = Z'^{T}_{w_j} \cdot \boldsymbol{H} \tag{7}$$

  với $Z'^{T}_{w_j}$ là vector đại diện cho từ thứ j trong corpus.

- Thế phương trình (6) vào phương trình (7), ta được:

$$\boldsymbol{u}_j = Z'^{T}_{w_j} \cdot \frac{1}{C}(Z_{w_1} + Z_{w_2} + \ldots + Z_{w_C})$$

- Tính toán loss function:

$$\mathfrak{L} = -u_j + \log\left(\sum_{j'=1}^{V} \exp\left(u'_j\right)\right)$$

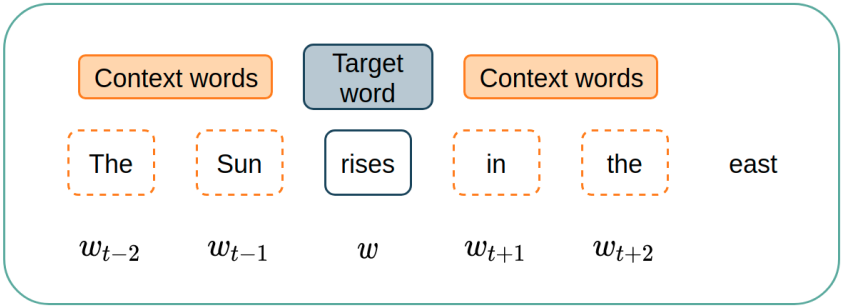
- Tiếp theo, là quá trình backward propagation. Cụ thể, quá trình cập nhật cho $\boldsymbol{W}$ sẽ như sau:

$$\boldsymbol{W} = \boldsymbol{W} - \alpha \mathfrak{L} \boldsymbol{H}^T \cdot \frac{(\boldsymbol{X}_1 + \boldsymbol{X}_2 + \ldots + \boldsymbol{X}_C)}{C}$$

- Đối với $\boldsymbol{W'}$ sẽ tương tự như single context word:

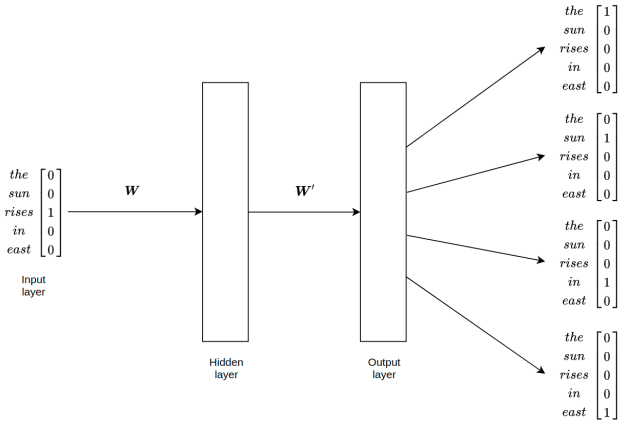
$$\boldsymbol{W'} = \boldsymbol{W'} - \alpha \boldsymbol{e}_j \cdot \boldsymbol{H}$$

- Skip-gram model về cơ bản như là việc làm ngược lại CBOW model. Với Skip-gram model, chúng ta sẽ cố gắng dự đoán các context word dựa vào target word được cung cấp ở đầu vào. Lấy lại ví dụ ban đầu của CBOW model, chúng ta sẽ dùng target word là *rises* để dự đoán các context word là *the*, *sun*, *in* và *east*.



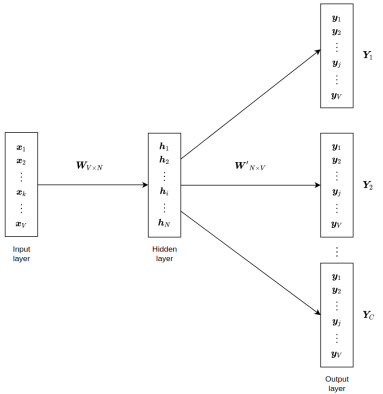
Hình 9: Target word và context word trong Skip-gram model.

- Dưới đây là kiến trúc của Skip-gram model:



Hình 10: Kiến trúc của Skip-gram model 1.

- Với một target word làm đầu vào $\boldsymbol{X}$ và trả về C context word $\boldsymbol{Y}$ ở đầu ra.



Hình 11: Kiến trúc của Skip-gram model 2.

Forward propagation

- **Bước 1**: Trước tiên chúng ta nhân $\boldsymbol{X}$ cho hidden layer weight $\boldsymbol{W}$.

$$\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}^T = Z_{w_I}$$

ở đây, Z_{w_I} là vector đại diện cho input word w_I.

- **Bước 2**: Tính $\boldsymbol{u}_j$, đây chính là điểm số dùng để đánh giá sự tương đồng giữa từ j trong corpus và target word.

$$\begin{aligned} \boldsymbol{u}_j &= \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H} \\ &= Z'^T_{w_j} \cdot \boldsymbol{H} \end{aligned}$$

với $Z'^T_{w_j}$ là vector đại diện cho từ j.

Forward propagation

- **Bước 3**: Không giống với CBOW model chỉ cần dự đoán duy nhất một target word. Ở đây, dự đoán ra C context word. Nên để cho tường minh chúng ta có thể ghi lại phương trình trên thành:

$$\boldsymbol{u}_{c,j} = Z'^{T}_{w_j} \cdot \boldsymbol{H} \text{ với } c = 1, 2, 3, \ldots, C$$

Vì $\boldsymbol{u}_{c,j}$ là điểm số đại diện cho từ thứ j là context word c. Cho nên:

- $\boldsymbol{u}_{1,j}$ là điểm số cho từ thứ j sẽ là context word đầu tiên.
- $\boldsymbol{u}_{2,j}$ là điểm số cho từ thứ j sẽ là context word thứ hai.
- $\boldsymbol{u}_{c,j}$ là điểm số cho từ thứ j sẽ là context word thứ c.

Forward propagation

- **Bước 4**: Chuyển toàn bộ các điểm số trên thành xác suất bằng hàm softmax và đại diện bởi $\boldsymbol{y}_{c,j}$.

$$\boldsymbol{y}_{c,j} = \frac{\exp\left(\boldsymbol{u}_{c,j}\right)}{\sum_{j'=1}^{V} \exp\left(\boldsymbol{u}_{j'}\right)} \tag{8}$$

$\boldsymbol{y}_{c,j}$ là xác suất cho từ thứ j trong corpus sẽ là context word thứ c.

Forward propagation

- **Bước 5**: Đặt $\boldsymbol{y}^*_{c,j}$ là xác suất để từ được chọn đúng là context word. Chúng ta cần tối thiểu hóa hàm $-\log$ xác suất này.

$$\min\left(-\log\left(\boldsymbol{y}^*_{c,j}\right)\right)$$

- **Bước 6**: Thế phương trình (8) vào phương trình trên ta được:

$$\mathfrak{L} = -\log\frac{\exp\left(\boldsymbol{u}_{c,j}\right)}{\sum_{j'=1}^{V}\exp\left(\boldsymbol{u}_{j'}\right)}$$

Forward propagation - *(tiếp theo)*

Và vì chúng ta có C context word, nên chúng ta sẽ lấy tích các xác suất này với nhau.

$$\begin{aligned}\mathfrak{L} &= -\log \prod_{c=1}^{C} \frac{\exp\left(\boldsymbol{u}_{c,j}\right)}{\sum_{j'=1}^{V} \exp\left(\boldsymbol{u}_{j'}\right)} \\ &= -\sum_{c=1}^{C} \boldsymbol{u}_{c,j} + C \cdot \log \sum_{j'=1}^{V} \exp\left(\boldsymbol{u}_{j'}\right)\end{aligned}$$

Backward propagation

- **Bước 1**: Áp dụng gradient descent, tính toán gradient cho loss function và cập nhật cho weight. Trước tiên, tính toán gradient cho $\boldsymbol{W'}$ bằng chain rule như sau:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W'}_{ij}} = \sum_{c=1}^{C} \frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_{c,j}} \cdot \frac{\partial \boldsymbol{u}_{c,j}}{\partial \boldsymbol{W'}_{ij}}$$

- **Bước 2**: Đạo hàm của $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_{c,j}}$ là sự khác biệt giữa actual value và predicted value: $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} = \boldsymbol{e}_{c,j}$

Backward propagation

- **Bước 3**: Do có nhiều context word, nên cộng tất cả các $\boldsymbol{e}_{c,j}$ này lại:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} = \sum_{c=1}^{C} \boldsymbol{e}_{c,j} = E$$

- **Bước 4**: Tính đạo hàm cho $\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}}$, và bởi vì chúng ta biết $\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$, cho nên:

$$\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}} = \boldsymbol{H}$$

Backward propagation

- **Bước 5**: Cho nên, gradient của loss function đối với $\boldsymbol{W'}$ như sau:

$$\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}} = E \cdot \boldsymbol{H}$$

- **Bước 6**: Tính gradient của loss function đối với $\boldsymbol{W}$.

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}_{ki}} = \frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i} \cdot \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{W}_{ki}}$$

Backward propagation

- **Bước 7**: Lúc này, gradient của loss function so với $\boldsymbol{W}$ và $\boldsymbol{W'}$ như sau:

$$\boldsymbol{W'} = \boldsymbol{W'} - \alpha \boldsymbol{E} \cdot \boldsymbol{H}$$

$$\boldsymbol{W} = \boldsymbol{W} - \alpha \boldsymbol{L} \boldsymbol{H}^T \cdot \boldsymbol{X}$$

- **Bước 8**: Kết thúc quá trình đào tạo, chúng ta sẽ tiến hành cập nhật cho các weight set trên network. Sau cùng $\boldsymbol{W}$ đạt trạng thái optimal weight và trở thành word vector cho vocabulary trong corpus.

Các chiến lược cải thiện hiệu suất (1)

Để tiến hành cải thiện hiệu suất của quá trình đào tạo model, người ta sẽ sử dụng kèm theo một số kỹ thuật như:

- Sử dụng **Hierarchical softmax** thay vì softmax truyền thống.
- **Negative sampling**.
- **Subsampling frequent words**.

Các chiến lược cải thiện hiệu suất (2)

Hierarchical softmax

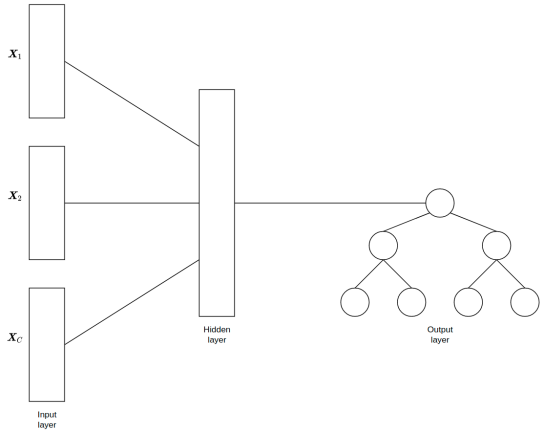
- Tính toán bằng softmax sẽ rất tốn kèm về mặt hiệu suất nên cần sử dụng một cách khác để tối ưu hóa hiệu năng tính toán đó là tính softmax trên cây - Hierarchical softmax:

$$\boldsymbol{y}_j = \frac{\exp\left(u_j\right)}{\sum_{j'=1}^{V} \exp\left(u'_j\right)}$$

  độ phức tạp $O(V)$.
- Hierarchical softmax sử dụng **Huffman binary search tree** *(cây tìm kiếm nhị phân Huffman)* để giảm độ phức tạp về mặt tính toán xuống $O(\log_2(V))$. Lúc này kiến trúc mạng có sự thay đổi đáng kể ở output layer - chúng ta sẽ thay output layer bằng một **binary search tree**.

Các chiến lược cải thiện hiệu suất (3)

Hierarchical softmax

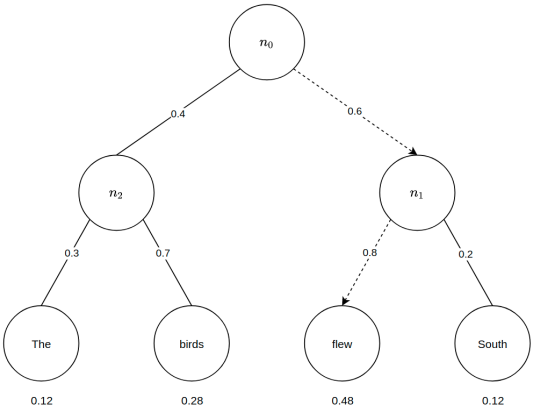


Hình 12: Thay thế output layer bằng Hierarchical softmax trong CBOW model.

- Đây là kiến trúc của Hierarchical softmax sau khi được nhúng vào output layer của network.
- Mỗi node lá bây giờ trong cây đại diện cho một từ trong corpus và tất cả các node trung gian đại diện cho **relative probability** *xác suất tương đối* của tất cả các node con của chúng.

Các chiến lược cải thiện hiệu suất (4)

Hierarchical softmax



Hình 13: Thay thế output layer bằng Hierarchical softmax trong CBOW model.

- Để tính xác suất của một target word khi được cung cấp context word. Chỉ cần duyệt cây và đưa ra quyết định nên duyệt cây bên trái hay bên phải tại một node nhất định. Giả sử cần tính xác suất của từ *flew* là target word dựa vào c context word. Chúng ta lấy tích của tất cả các xác suất dọc theo con đường đến target word.

$$\begin{aligned} p(\mathit{flew}|c) &= p_{n_0}(\mathit{left}|c) \times p_{n_1}(\mathit{right}|c) \\ &= 0.6 \times 0.8 \\ &= 0.12 \end{aligned}$$

Các chiến lược cải thiện hiệu suất (5)

Negative sampling

- Xét câu: *"Birds are flying in the sky."* với target word là *flying* và còn lại là các context word. Chúng ta cần cập nhật lại các weight trong network mỗi khi nó dự đoán một target word không chính xác. Vì vậy, ngoại trừ từ *flying*, nếu một từ khác được dự đoán là target word thì chúng ta sẽ cập nhật network.
- Như vậy, với corpus chứa hàng triệu vocabulary sẽ rất hao tốn tài nguyên về mặt tính toán.
- Để khắc phục điều này, đánh dấu correct target word là positive class và lấy mẫu một vài từ còn lại thuộc negative class. Về cơ bản, chúng ta đang chuyển về bài toán phân loại nhị phân. Lúc này, xác suất để một từ được chọn là negative word.

$$p(w_i) = \frac{frequency(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} frequency(w_j)^{\frac{3}{4}}}$$

Các chiến lược cải thiện hiệu suất (6)

Subsampling frequent words

- Trong corpus, sẽ có một số từ xuất hiện rất thường xuyên chẳng hạn như *the*, *is*,... và có một số từ lại xuất hiện không thường xuyên cho lắm. Để duy trì sự cân bằng giữa hai yếu tố, người ta sử dụng kỹ thuật **subsampling** *(lấy mẫu con)*. Vậy nên, chúng ta sẽ loại bỏ một số từ xuất hiện thường xuyên hơn một **threshold** nhất định với xác suất p được tính như sau:

$$p(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

ở đây t là threshold và $f(w_i)$ là frequency của tử i.

Chạy thử

Tài nguyên tham khảo

- **Hands-On Deep Learning Algorithms with Python** - *(chương 7)*, link sách tại https://www.amazon.com/Hands-Deep-Learning-Algorithms-Python/dp/1789344158.
- **Natural Language Processing in Action**, link sách tại https://www.amazon.com/Natural-Language-Processing-Action-Understanding/dp/1617294632.

Hỏi đáp và kết thúc

HẾT

Cảm ơn Thầy và các bạn đã chú ý lắng nghe.