Đại học Khoa học tự nhiên

Môn học: Khoa Học Dữ Liệu

Báo cáo đồ án cuối kỳ

Word2Vec

Giảng viên
Nguyễn Ngọc Đức

Sinh viên
20424008 - Dương Mạnh Cường

Ngày 11 tháng 2 năm 2022

Mục lục

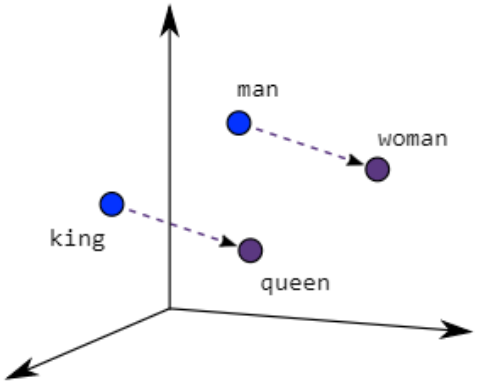
Neural network yêu cầu đầu vào ở dạng **numeric**. Cho nên, khi ta có dữ liệu dạng **text**, ta cần chuyển đổi chúng thành dữ liệu dạng numeric.

Có nhiều phương pháp khác nhau để chuyển đổi dữ liệu dạng text sang numeric mà phổ biến nhất là:

- Term frequency-inverse document frequency (TF-IDF).
- Bag of words (BOW).

Tuy nhiên, điểm yếu của hai phương pháp trên là chúng **không nắm bắt được ngữ nghĩa của từ**, nói cách khác là chúng không hiểu được ý nghĩa của từ.

Có nhiều cách khắc phục nhược điểm này, mà một trong những cách đó là sử dụng **Word2Vec** bằng cách đại diện cho từng từ bằng một **vector** trong không gian m chiều. Lúc này, các từ có nghĩa tương đồng nhau sẽ nằm gần nhau.



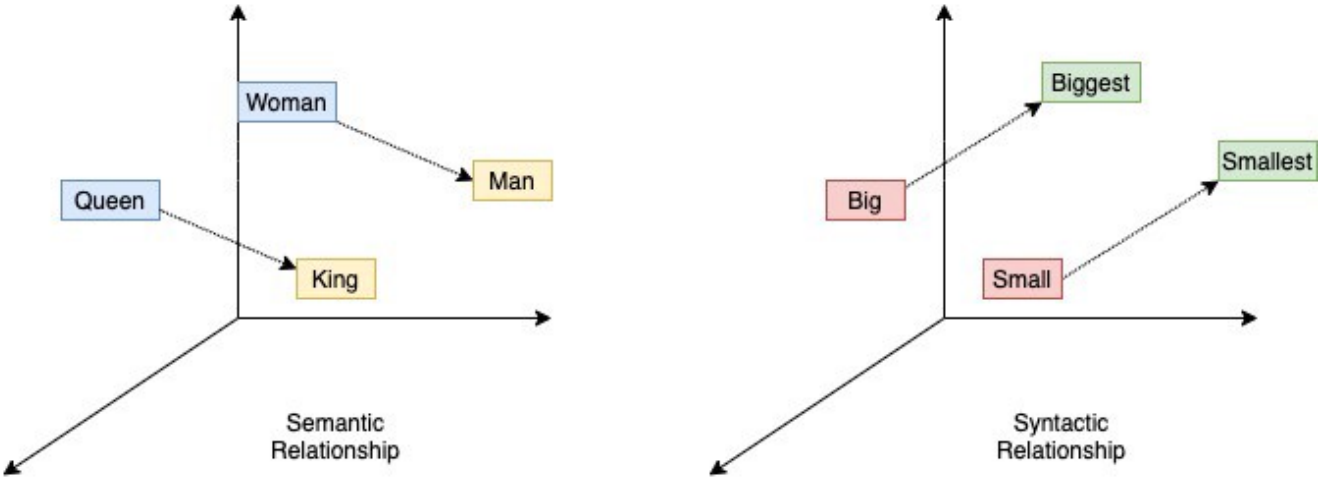
Hình 1: Các từ có ý nghĩa tương đồng nhau nằm gần nhau.

1 Word2Vec model

Word2Vec là một trong những phương pháp **word embedding** được sử dụng phổ biến.

Word embedding là cách ta biểu diễn các **word vector** trong không gian vector.

Các word vector được tạo ra bởi Word2Vec model có khả năng nắm bắt được các **semantic** *(ngữ nghĩa)* và **syntactic** *(ý nghĩa cú pháp)* của từ.



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

Ví dụ có câu: *"Archie used to live in New York, he then moved to Santa Clara. He loves apples and strawberries."*.

Word2Vec model sẽ phát sinh các vector cho từng từ trong văn bản. Nếu chúng ta trực quan các vector này trong không gian vector tương ứng, chúng ta có thể thấy các từ tương tự nhau sẽ nằm gần nhau.

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

Nhận xét

- Ở đây các cặp từ như *apples - strawberries*, *New York - Santa Clara* có ý nghĩa tương đồng nhau nên nằm gần nhau.

Do đó, với Word2Vec model có thể học cách biểu diễn các vector giúp cho neural network có thể hiểu được ý nghĩa của từ tương ứng với vector đó.

Và vì chúng ta có thể hiểu được semantic và syntactic của từ điều này giúp ta tận dụng các vector này 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:

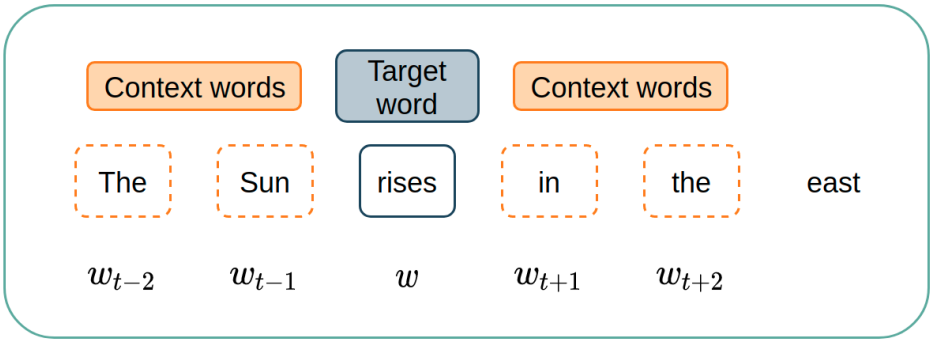
1. **CBOW model**.
2. **Skip-gram model**.

1.1 CBOW model

Giả sử chúng ta có một neural network bao gồm: **một input layer**, **một hidden layer** và **một output layer**. Mục đích của network này là dự đoán ra **một từ** dựa vào **các từ xung quanh nó**. Từ mà chúng ta cố gắng dự đoán được gọi là **target word** và các từ xung quanh nó được gọi là **context word**.

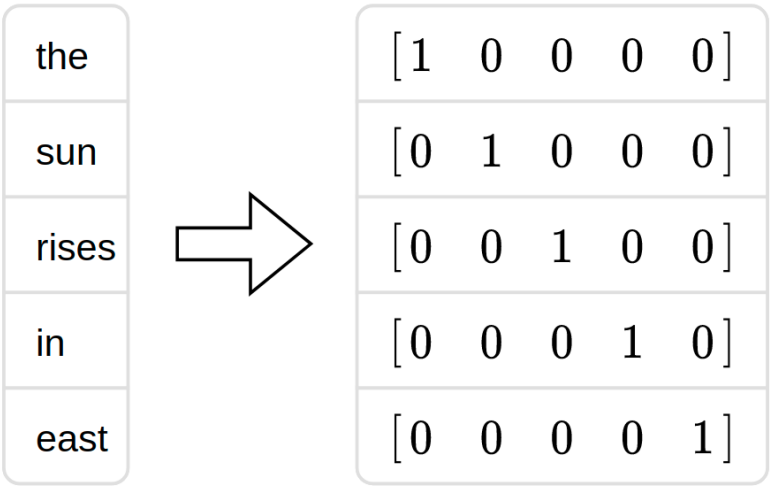
Vậy chúng ta cần bao nhiêu context word để dự đoán ra target word? Chúng ta sẽ sử dụng một **window** *(cửa sổ)* có kích thước là n để chọn các context word. Nếu $n = 2$ thì chúng ta sẽ sử dụng hai từ **phía trước** và **phía sau** của target word làm các context word.

Xem xét câu sau: *"The Sun rises in the east."* với *rises* là target word. Nếu chúng ta xét kích thước của window là 2 thì chúng ta sẽ lấy hai từ phía trước là *The*, *Sun* và hai từ phía sau là *in*, *the* của target word làm các context word.



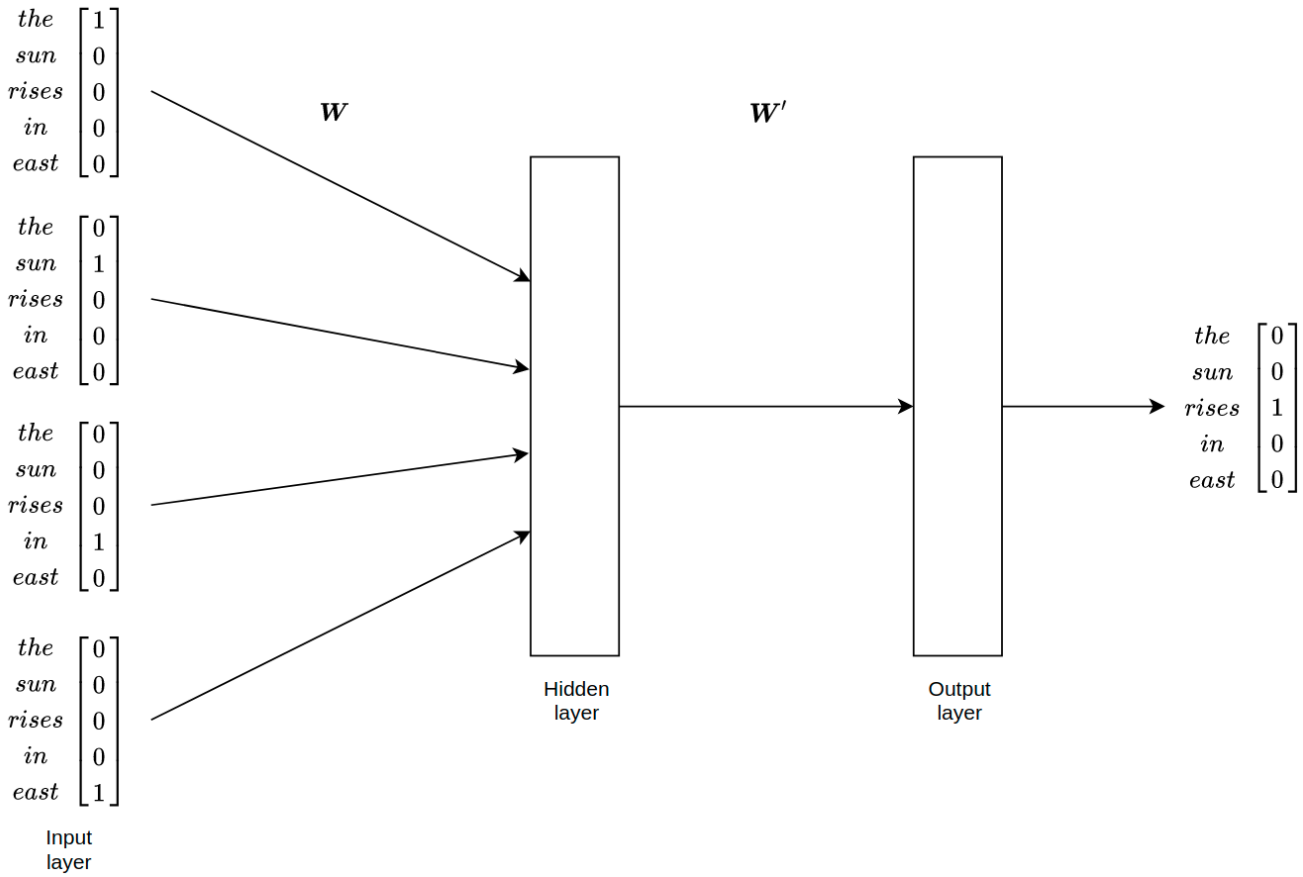
Hình 4: Target word và context word trong CBOW model.

Lúc này input của network là các context word và output của network là target word. Và vì neural network chỉ chấp nhận input là numeric data nên chúng ta sẽ sử dụng kỹ thuật **one-hot encoding** để chuyển đổi các text data thành numeric data.



Hình 5: One-hot encoding cho text data.

Kiến trúc của CBOW model được thể hiện dưới hình sau. Ở đây các context word là: *the*, *sun*, *in* và *east* được dùng làm đầu vào cho network và dự đoán ra target word là *rises* ở đầu ra.



Hình 6: Kiến trúc của CBOW model 1.

Ở vài lần lặp đầu tiên, network không thể dự đoán target word một cách chính xác. Nhưng sau một loạt các vòng lặp bằng cách sử dụng **gradient descent**, các **weight** *(trọng số)* của network được cập nhật và tìm ra được **optimal weight** *(trọng số thích hợp)* để dự đoán ra target word một cách chính xác.

Vì chúng ta có một input layer, một hidden layer và một output layer, nên chúng ta sẽ có hai weight:

1. Weight từ input layer đến hidden layer - $\boldsymbol{W}$.
2. Weight từ hidden layer đến output layer - $\boldsymbol{W'}$.

Trong quá trình đào tạo network, các weight sẽ được cập nhật trong quá trình back propagation nhằm tìm ra optimal weight cho hai bộ $\boldsymbol{W}$ và $\boldsymbol{W'}$.

Về sau, weight set giữa input layer và hidden layer $\boldsymbol{W}$ được cập nhật và tối ưu tạo thành các vector đại diện cho các từ của input layer.

Sau khi kết thúc quá trình đào tạo, chúng ta chỉ cần loại bỏ output layer và lấy ra weight set giữa input layer và hidden layer và gán chúng cho các từ tương ứng.

Dưới đây là các vector tương ứng cho các từ của $\boldsymbol{W}$. Word embedding tương ứng cho từ *sun* là $\begin{bmatrix}0.0 & 0.3 & 0.3 & 0.6 & 0.1\end{bmatrix}$.

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

1.1.1 CBOW model với một context word

CBOW model cần một số lượng context word C nhất định để dự đoán target word. Ở phần này chúng ta sẽ xem 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.

Trước tiên, chúng ta cần làm quen với một vài khái niệm. Tất cả các từ duy nhất nằm trong **corpus** *(kho ngữ liệu)* của chúng ta được gọi là các **vocabulary** *(từ vựng)*. Trong ví dụ trước đó của chúng ta, chúng ta có năm từ là: *the*, *sun*, *rises*, *in* và *east* - toàn bộ năm từ này chính là vocabulary trong corpus của chúng ta.

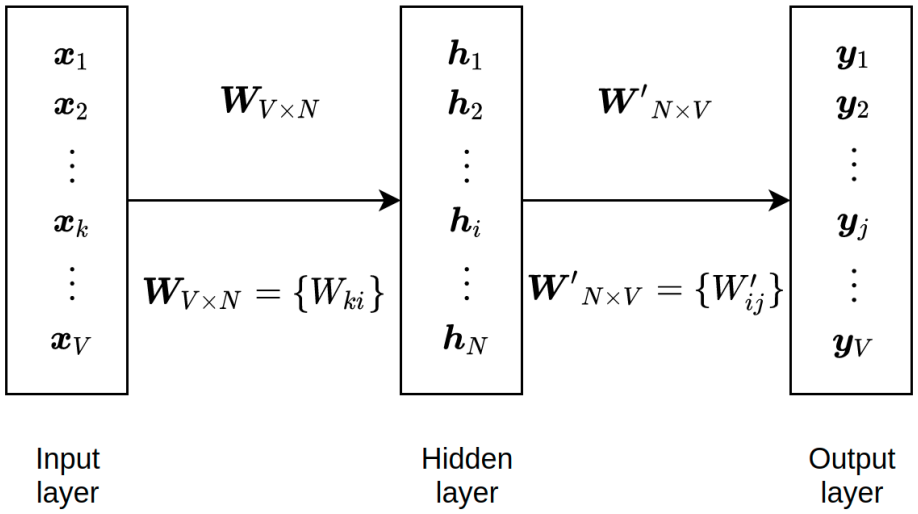
Đặt V là số lượng vocabulary và N là số neuron của hidden layer. Vì neural network của chúng ta có ba layer, cụ thể:

- 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.

1.1.1.1 Forward propagation

Để dự đoán target word từ các context word, chúng ta sẽ thực hiện quá trình forward propagation. Trước tiên, chúng ta nhân $\boldsymbol{X}$ cho $\boldsymbol{W}$.

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

Chúng ta biết rằng input là các one-hot vector, nên khi ta nhân $\boldsymbol{X}$ với $\boldsymbol{W}$ thì $\boldsymbol{H}$ lúc này đơn giản là:

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

$\boldsymbol{W}_{(k,.)}$ về cơ bản 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.

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

Bây giờ chúng ta đang ở hidden layer, tiếp theo chúng ta cần tính xác suất cho từng từ trong corpus để nó là một target word.

Đặt $\boldsymbol{u}_j$ là điểm số để từ thứ j trong corpus là target word - được tính bằng cách nhân $\boldsymbol{H}$ cho $\boldsymbol{W'}$. Và vì chúng ta đang tính toán điểm số cho từ j trong corpus, nên ta chỉ nhân cột j trong $\boldsymbol{W'}$ cho $\boldsymbol{H}$.

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

Lúc này, cột j của $\boldsymbol{W'}_{ij}$ đại diện cho vector của từ j trong corpus. Đặt vector đại diện cho từ thứ j là $Z'_{w_j^T}$. Phương trình trên có thể được viết lại thành:

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

Thế phương trình (1) vào phương trình (2), ta được:

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

Việc tính toán dot product giữa $Z'_{w_j^T}$ và Z_{w_I} nói cho chúng ta biết độ tương đồng giữa target word với các context word đầu vào. Cho nên nếu điểm số của từ j trong corpus cao thì có nghĩa có sự tương đồng cao giữa các context word và target word j này và ngược lại.

Cuối cùng, ta sẽ áp dụng softmax activation để chuyển đổi các điểm số này sang xác suất:

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

Lúc này, $\boldsymbol{y}_j$ cho chúng ta biết xác suất của từ j là target word. Chúng ta sẽ tính xác suất cho tất cả các từ trong corpus và chọn từ mà có xác suất cao nhất là target word.

Tiếp theo, chúng ta sẽ tính toán hàm loss. Đặt $\boldsymbol{y}_j^*$ là xác suất để từ j chính xác là target word. Chúng ta cần tối đa xác suất này:

$$\max \boldsymbol{y}_j^*$$

Và để giảm phạm vị khi tính hàm loss ta sẽ tính tối đa hàm log của xác suất.

$$\max \log\left(\boldsymbol{y}_j^*\right)$$

Nhưng vì chúng ta sẽ áp dụng gradient descent để tìm ra optimal weight nên thay vì tối đa hóa ta sẽ tìm tối thiểu hóa công thức trên như sau:

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

Vậy lúc này loss function của chúng ta sẽ thành:

$$\mathfrak{L} = -\log\left(\boldsymbol{y}_j^*\right) \tag{4}$$

Thế phương trình (3) vào phương trình (4) ta thu được:

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

1.1.1.2 Backward propagation

Để tối thiểu hóa loss function, chúng ta sử dụng gradient descent. Chúng ta sẽ tính gradient của loss function để cập nhật weight.

Chúng ta có hai weight set là $\boldsymbol{W}$ và $\boldsymbol{W'}$. Trong quá trình backward propagration, chúng ta có thể tiến hành cập nhật weight cho chúng. Cụ thể như sau:

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

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

Trước tiên, trong quá trình forward propagation chúng ta có các phương trình sau:

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

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

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

Trước tiên, chúng ta sẽ tính gradient của loss function cho $\boldsymbol{W'}$. Chúng ta không thể tính gradient của loss function đối với $\boldsymbol{W'}$ trực tiếp. Cho nên chúng ta phải áp dụng **chain rule** như sau:

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

Ta có đạo hàm của $\dfrac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j}$ như sau:

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{u}_j} = \boldsymbol{e}_j \tag{5}$$

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

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

Tiếp theo, chúng ta sẽ tính đạo hàm cho $\dfrac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{W'}_{ij}}$, và bởi vì chúng ta biết $\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$ nên:

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

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

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

Tiếp theo, chúng ta cần tính gradient của loss function đối với $\boldsymbol{W}$, và chúng ta cũng không thể tính gradient trực tiếp nên chúng ta phải áp dụng chain rule.

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

Để tính đạo hàm của $\frac{\partial \mathfrak{L}}{\partial \boldsymbol{h}_i}$, chúng ta áp dụng chain rule:

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

Thế phương trình (5) vào phương trình trên, ta được:

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

Và vì chúng ta biết $\boldsymbol{u}_j = \boldsymbol{W'}_{ij}^T \cdot \boldsymbol{H}$, tiếp tục thế vào phương trình trên chúng ta được:

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