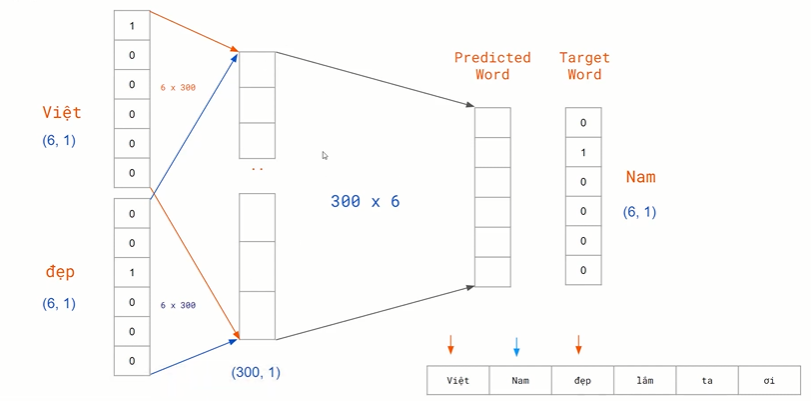
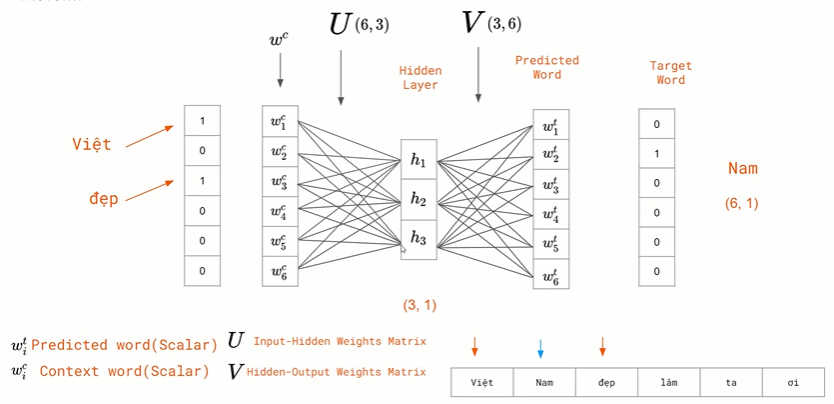
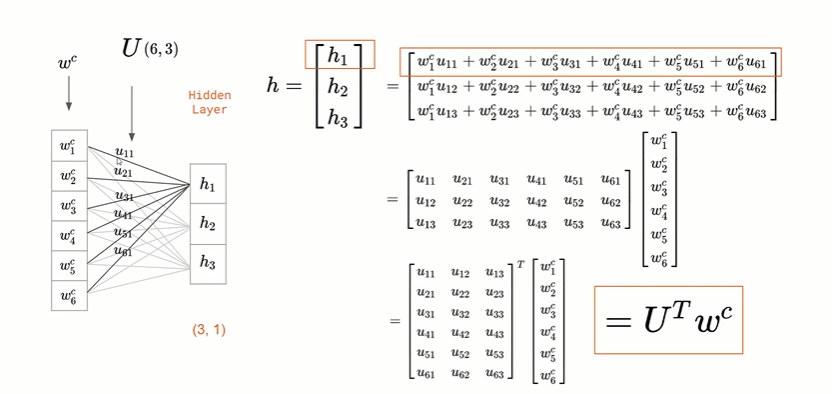
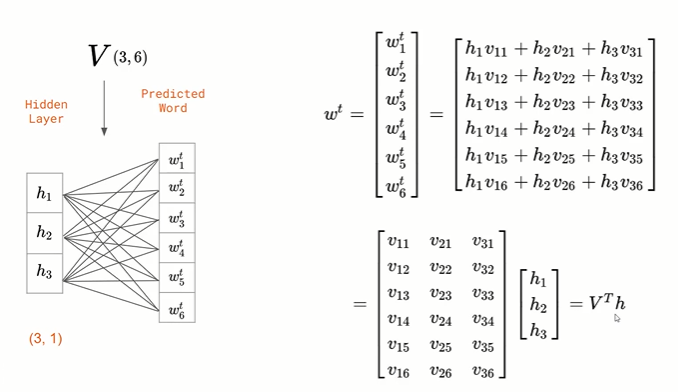
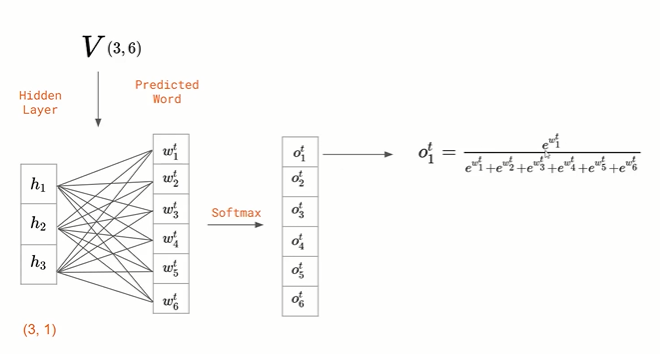
Đi sơ lược qua phần lý thuyết:





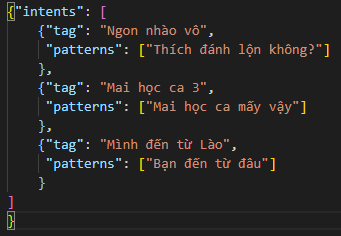






Đến phần thực hành:

Dataset:



“Thích đánh lộn không”

“Mai học ca mấy vậy”

“Bạn đến từ đâu”

Bộ từ điển sau khi phân tích các câu trên:



Vector từ của các câu trên theo thứ tự:

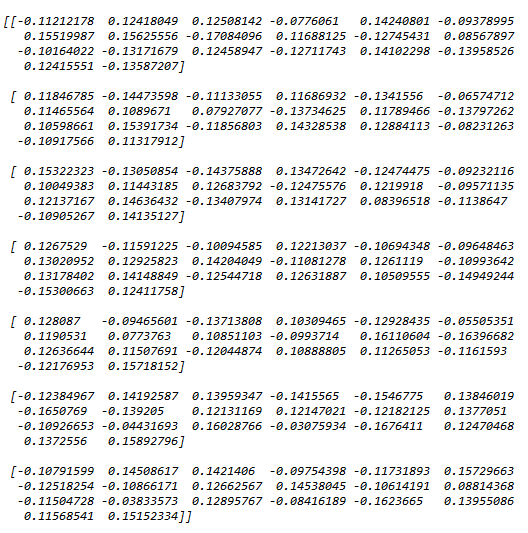


Vector của các tag:

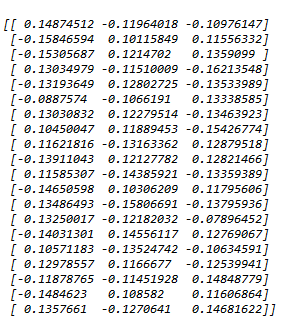


Các vector này sẽ được xáo trộn để thể hiện rõ bản chất học máy

Vector embedding: (Đây là ma trận (1)) là một ma trận trọng số (weight) Có thể hiểu nôm na nó là chỉ số thông minh của mạng nơ-ron



Context vector: (Đây là ma trận (2))



Sau mỗi n\_epoch tensorflow sẽ hỗ trợ tính toán embedding vector và context vector và điều chỉnh thông số này sao cho kết quả thu được là chính xác nhất

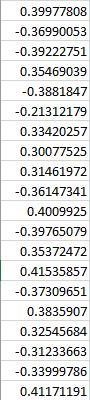
Sau khi train xong và thu được vector embedding và vector context thì chúng ta tiến hành dự đoán đầu ra (Đưa vào một input chưa từng có trong dataset):

Ở đây ta sẽ lấy input để test là “Học ca mấy”. Đầu ra mong đợi của chúng ta sẽ là “Mai học ca 3” trong dataset

Bắt đầu tính tay!

Sau khi tính toán, ta thu được vector của câu “Học ca mấy” là: [0, 1, 1, 0, 1, 0, 0] (Đây là ma trận (3))

Lấy chuyển vị của (1) nhân với (3) ta thu được:

 (Đây là ma trận (4))

Lấy chuyển vị của (2) nhân với (4) ta thu được:

 (Đây là ma trận (5))

Áp dụng exponential function (y = e^x) cho từng hàng của ma trận trên ta thu được kết quả:



Tính tổng các phần tử trên là: 3.695996922

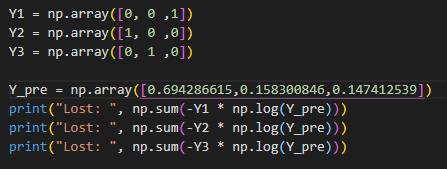
Áp dụng công thức softmax ta thu được softmax của từng thành phần ma trận (5) thu được kết quả lần lược là:

 (Đây là kết quả tính tay)

(Đây là kết quả của việc chạy hàm model.predict())



Áp dụng loss function crossentropy:



Ta thu được kết quả:



Ta thấy, lost function của Y2 (tức, ma trận [1, 0, 0]) có giá trị gần 0 nhất

Vì sau khi lệnh này được thực thi:



Các tag sẽ được sắp sếp lại ntn:



* Kết quả thu được sẽ là “Mai học ca 3”

# Thực ra chúng ta cũng đã có thể kết luận khi thu được giá trị của hàm softmax (69% ở phía trên)