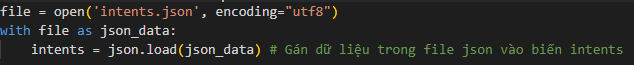
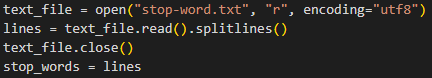
Đọc dữ liệu trong file json (File dataset)

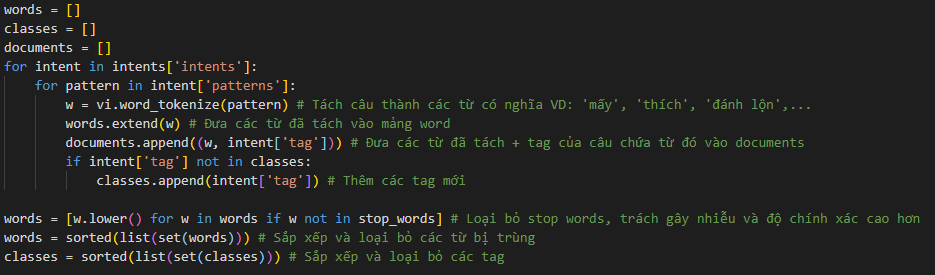


Lấy ra stop word trong file stop-word.txt

STOP WORD: hiểu đơn giản là các từ có tần số xuất hiện nhiều như à, à này, ... các từ này thường mang ít giá trị ý nghĩa và không khác nhau nhiều trong các văn bản khác nhau



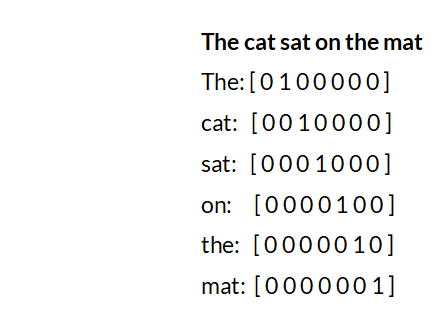
Xử lý dữ liệu train



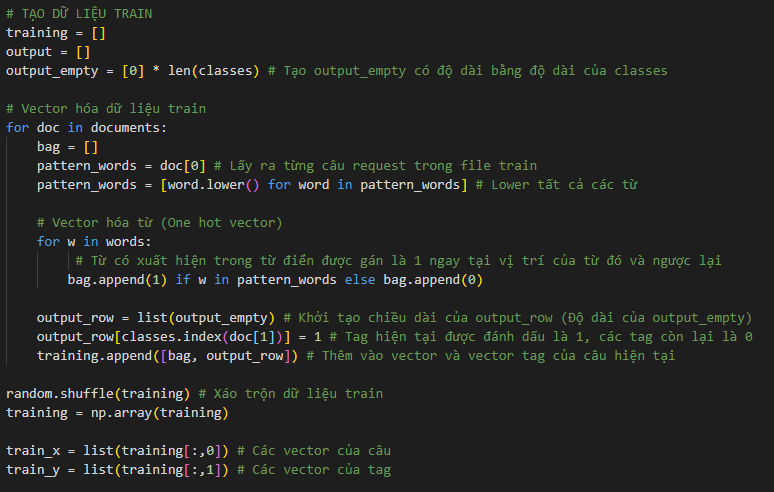
Tạo dữ liệu train

Nếu chúng ta sử dụng dự liệu dạng word thì sẽ không thể hoạt động được với tensorflow, vì vậy việc cần thiết bây giờ là chuyển dữ liệu này sang dạng tensor number

One hot vector: mỗi giá trị hạng mục sẽ được mã hóa bằng một vector nhị phân với toàn bộ các phần tử bằng 0 trừ một phần tử bằng 1 tương ứng với vị trí của giá trị hạng mục đó trong từ điển.

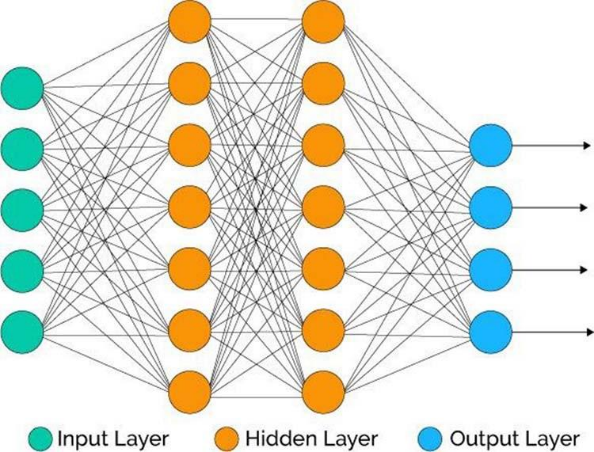


One hot vector: đầu ra là '0' cho mỗi thẻ và '1' cho thẻ hiện tại, độ dài của one-hot-vector sẽ bằng số lượng từ trong một từ điển



Build neural network

Ở đây chúng ta xây dựng một mạng nơ-ron bao gồm 1 lớp đầu vào (input layer), 1 lớp đầu ra (output layer), 4 lớp ẩn (hidden layer)

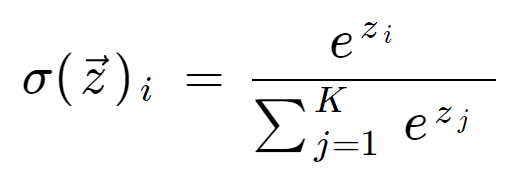


Khởi tạo giá trị bias bằng 0 để phần xử lý có thể đơn giản nhất và dễ hiểu nhất

SOFTMAX: Hàm softmax tính toán xác suất xảy ra của một sự kiện. Hàm softmax sẽ tính khả năng xuất hiện của một class trong tổng số tất cả các class có thể xuất hiện. Sau đó, xác suất này sẽ được sử dụng để xác định class mục tiêu cho các input.

SOFTMAX: Xác suất sẽ luôn nằm trong khoảng (0:1].

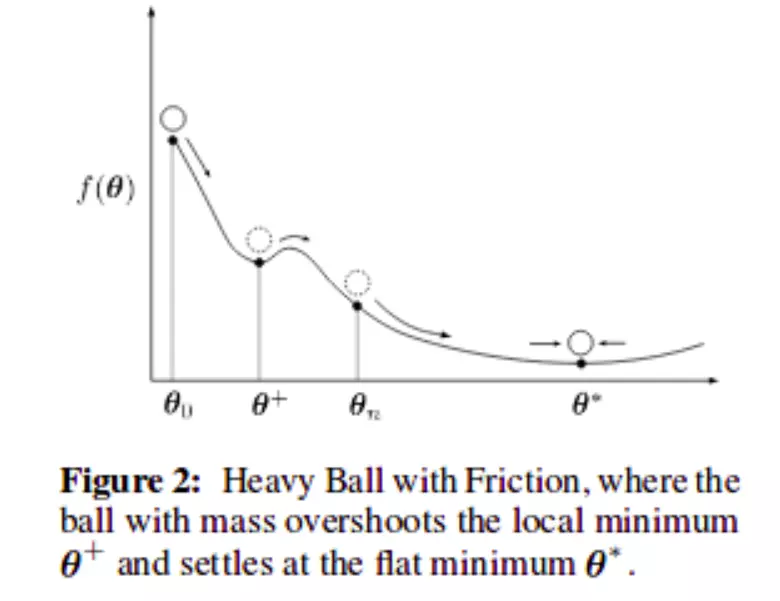
SOFTMAX: Chuẩn hóa softmax là một cách để giảm thiểu ảnh hưởng của những giá trị cực trị hay dữ liệu ngoại lai trong dữ liệu mà không phải chỉnh sửa dữ liệu ban đầu.



Optimizer: Tìm 1 cặp weights và bias phù hợp để tối ưu hóa model. Nhưng vấn đề là "học" như thế nào? Cụ thể là weights và bias được tìm như thế nào! Đâu phải chỉ cần random (weights, bias) 1 số lần hữu hạn và hy vọng ở 1 bước nào đó ta có thể tìm được lời giải. Rõ ràng là không khả thi và lãng phí tài nguyên! Chúng ta phải tìm 1 thuật toán để cải thiện weight và bias theo từng bước, và đó là lý do các thuật toán optimizer ra đời. Ở đây chúng ta sử dụng optimizer là Adam

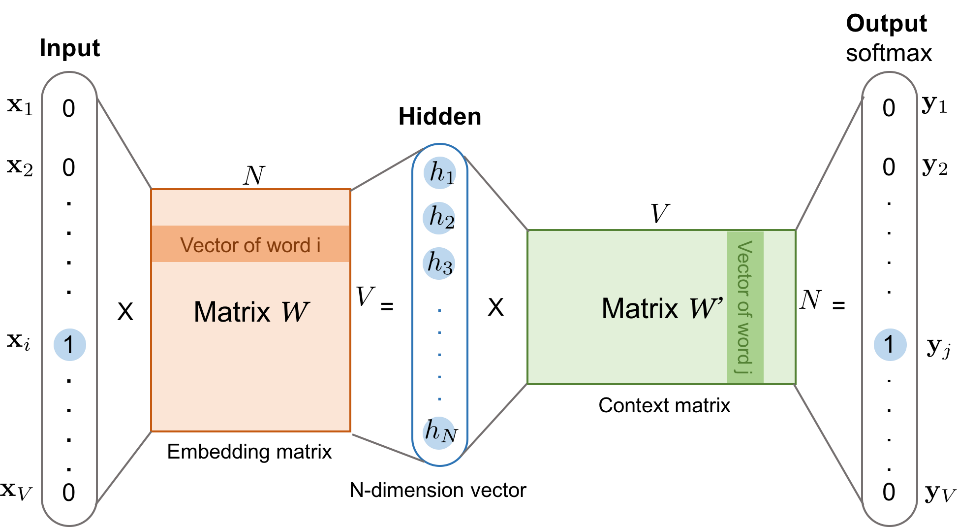
ADAM: là sự kết hợp của Momentum và RMSprop. Adam như 1 quả cầu rất nặng có ma sát, vì vậy nó dễ dàng vượt qua local minimum tới global minimum và khi tới global minimum nó không mất nhiều thời gian dao động qua lại quanh đích vì nó có ma sát nên dễ dừng lại hơn (Mục tiêu là tìm ra giá trị của loss function nhỏ nhất và nhanh nhất. Vì khi loss function càng gần bằng 0 thì định nghĩa càng được khẳng định)

ADAM: Yêu cầu rất ít bộ nhớ, Thích hợp với các bài toán có độ biến thiên không ổn định và dữ liệu traning phân mảnh, yêu cầu ít điều chỉnh



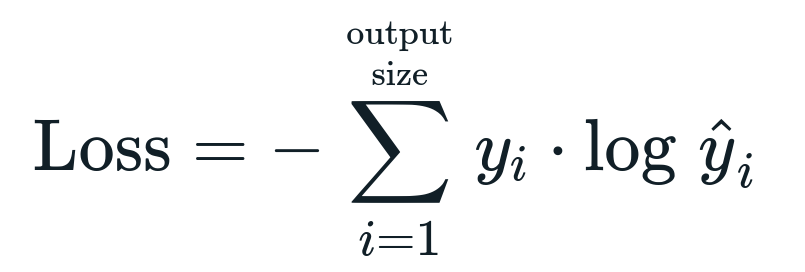
Embedding là một kỹ thuật đưa một vector có số chiều lớn, thường ở dạng thưa, về một vector có số chiều nhỏ, thường ở dạng dày đặc.

Ma trận embedding có thể coi là một ma trận trọng số (weight) trong một mạng neural nhân tạo.



Loss Function: để tối ưu model của mình sao cho tốt nhất, hay cũng dùng để đánh giá độ tốt của model (Giá trị của loss function càng về 0 càng tốt)

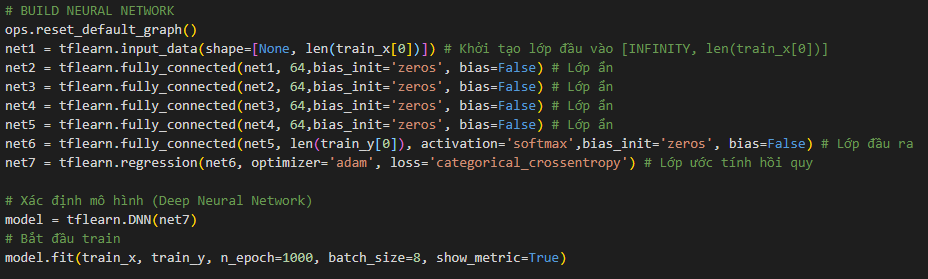
Crossentropy: là độ đo đánh giá độ dài trung bình số lượng bit cần thiết để mã hoá thông điệp với phân bố xác suất là q(x)



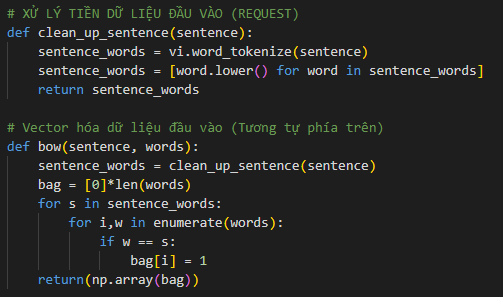
N\_epoch: Lặp qua dataset 1000 lần

Batch\_size: chia tập dữ liệu thành các phần, mỗi phần có kích thước là batch size.

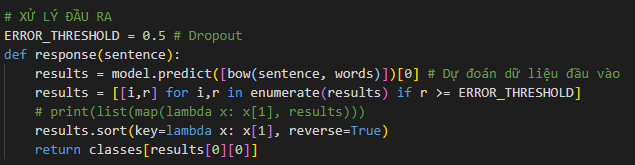
Show\_metric: Show độ chính xác (accuracy)



Xử lý tiền dữ liệu đầu vào



Xử lý đầu ra



Kết quả

