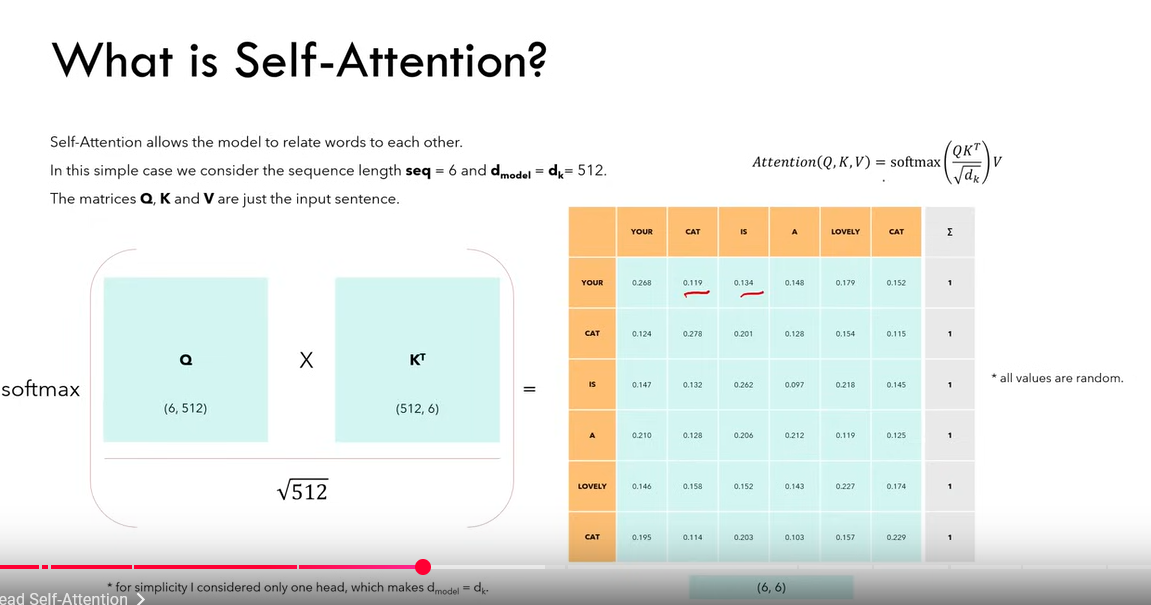
# Self Attention

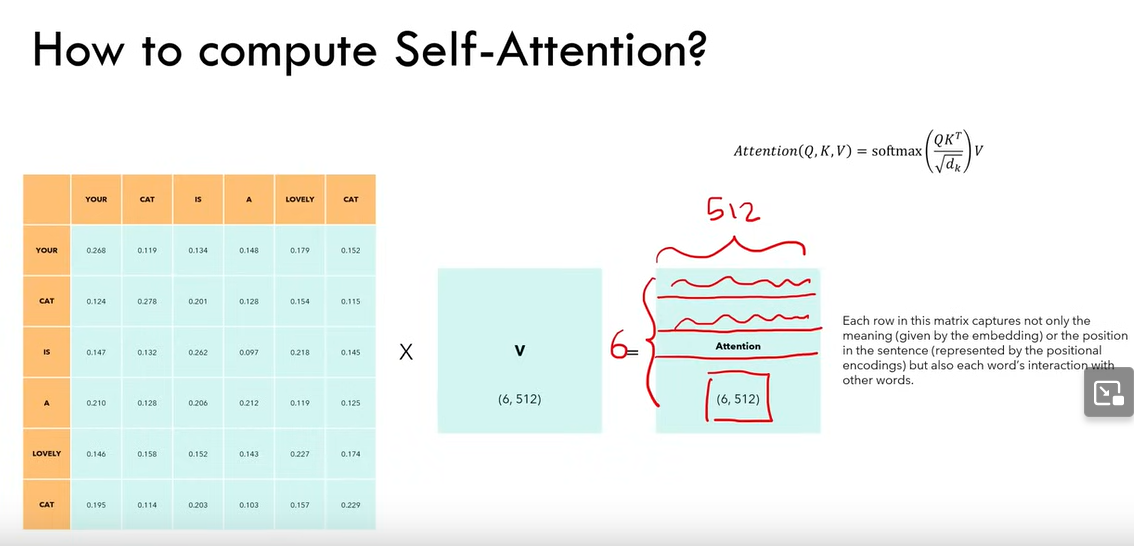
## Mối quan hệ từ - từ



* Là cơ chế đế biểu diễn mối liên quan giữa từ này, với các từ còn lại.
* Q,K,T ban đầu sẽ là input sentence biểu diễn của từ đó.
* Trong ví dụ, một câu có 6 từ, với mỗi từ biểu diễn thành một vector 512 chiều, thì ta có được vector [6x512]
* Ta dùng QxK.T, thì ta được vector 6x6.
* Nếu nhân ma trận, ta sẽ thấy, chính là hàng đầu Q, nhân với cột (hàng đảo ngược) K, chính là vector biểu diễn hai từ nhân với nhau.
* Thành ra sau khi ra được ma trận [6x6], chính là ma trận biểu diễn mối quan hệ giữa **một** từ này với **một** từ kia.

## Mối quan hệ từ - tất cả từ

* Sau đó ta nhân với V (Lưu ý chia \sqrt{d\_k} để chuẩn hóa, áp dụng softmax):



* Ta dùng [6x6] nhân với [6x512] thì được [6x512] giống với shape input ban đầu.
* Ở đây, một hàng, không chỉ biễu diễn ý nghĩa, hay vị trí, mà là ý nghĩa của mỗi từ với toàn bộ từ khác.
* Vì ta thấy, ví dụ hàng 1 R (reuslt) nhân với cột 1 của V, thì chính là:
  + Hàng 1: Mối quan hệ từ Your, với Your, Cat, Is….
  + Cột 1: Biểu diễn số 1 của tất cả các từ.
* Ta nhân lại, được mối liên hệ, **một** từ, với tất cả các từ.

## Chi tiết Self-Attention

### Permutaion invariant (Bất biến hoán v

* (Nếu không xem sự đóng góp của positional encoding).
* A red line drawing of a ladder

  Description automatically generated with medium confidence
* Vì ta thay đổi vị trí B,C, thì kết quả cũng thay đổi theo sự thay đổi này luôn.

### Không có parameter

* Chỉ có nhân, chia, softmax.
* Tất nhiên, encoding của input thì cần được train.
* Trong **multi-head** cần train.

### Giá trị của đường chéo sẽ lớn nhất

* Vì, từ đó có mối quan hệ với chính nó, nên giá trị đó phải là lớn nhất.

### Có thể tùy chỉnh mối quan hệ

* Ví dụ nếu ta muốn một vài vị trí không interact với nhau, mình có thể chỉnh giá trị thành -inf trước khi áp dụng softmax 🡪 sẽ thành 0 coi như là không có mối quan hệ với nhau.
* Sử dụng decode thì ok.
* 

# Multi-head Attention

## A screenshot of a computer Description automatically generated

## Giải thích kiến trúc

* Đầu tiên, tương tự bên kia thì ta có input (đã cộng với positional encoding), ta chia làm 4 phần, một phần đi thẳng qua layer normalization.
* 3 phần còn lại đặt tên là Q,K,V (Sẽ giải thích sau).
* Sau đó, 3 Q,K,V này không tính trực tiếp, mà đi qua lớp FCN Wq, Wk, Wv (d\_model, dmodel) này, ra shape giống ban đầu.
* Bây giờ mới là ý của multi head attention, không phải như kia mà nhân thẳng Q’, K’, rồi V’.
* MÀ ta chia thành nhiều head (H), với H=4 (4 phần ở trên), theo chiều d\_model
* Tức là mỗi phần đều đủ từ, nhưng là một ¼ biểu diễn của 512.
* Sau đó mỗi phần nhỏ này tính tương tự như self-attention. Gọi từng kết quả là từng head\_i.
* Thì sau đó, ta sẽ concat các head lại, tạo thành shape ban đầu (seq,dmodel).
* Ở đây ta tiếp tục đưa qua W0 một FCN để train tiếp. 🡪 MH-A.
* Thay vì tính attention, thẳng Q’, K’, V’. tính các đầu nhỏ, một ý hay: “**Mỗi head xem mỗi khía cạnh khác nhau của một từ”.**
* Ví dụ như ngôn ngữ Trung thì, hay bất cứ nào, một word có thể là noun, adj, adv,… 🡪 mỗi head học một cái riêng.
* Attention can be visualized (ảnh trên – cuối của paper).
* Sau khi áp dụng softmax các Q,K nhỏ. Thì cũng ra mối quan hệ giữa hai từ.
* A close-up of a diagram

  Description automatically generated
* Tức là từ making này, có mỗi quan hệ với từ ví dụ như difficult, là theo đầu đỏ, xanh, vàng,…
* Và, ví dụ mỗi cái quan hệ này, có thể thấy một phần mà các đầu khác không thấy?

## Tại sao Q, K, V? (Query, Keys, Values)

* Ví dụ database: A diagram of a diagram

  Description automatically generated with medium confidence
* User make query 🡪 love 🡪 embedding
* Tính toán dot product, query, và toàn bộ các keys 🡪 softmax.
* Amplify, nào liên quan nhất!!!

## Mask matrix

* Trước khi nhân với V, ta có áp dụng softmax, và nếu ta không muốn hai từ liên quan với nhau thì ta sẽ áp -inf vào, thì đây chính là công việc của mask.
* Để có thể từ phía sau không ảnh hưởng từ phía trước (tại vì chưa thấy mà!!).