Simply explain Attention Model

Introduction

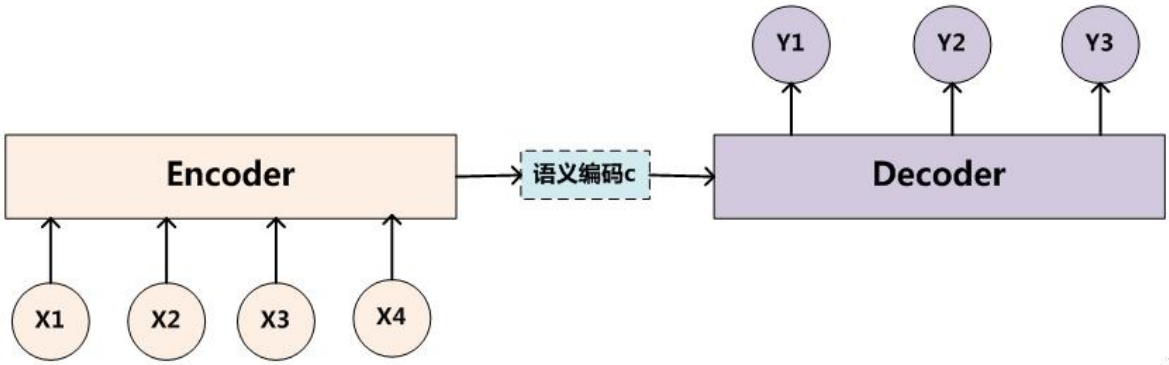
In the last two years, attention models have been widely used in various types of deep learning tasks such as natural language processing, image recognition and speech recognition, and are one of the core techniques in deep learning technologies that deserve the most attention and insight.

In other words, when we look elsewhere, our attention shifts with the movement of our gaze, which means that when people pay attention to a target or a scene, the distribution of attention is different within the target and at each spatial location within the scene.

From the perspective of the role of Attention, we can classify Attention types from two perspectives: **Spatial Attention** and **Temporal Attention**. For more practical applications, we can also classify Attention into **Soft Attention** and **Hard Attention**. Soft Attention is all data will be paid attention to, will calculate the corresponding attention weights, will not set the filtering conditions. hard Attention will generate attention weights after filtering out a part of the attention does not meet the conditions, so that its attention weight is 0, that is, can be understood as no longer pay attention to these do not meet the conditions of the part.

Background

Encoder Decoder framework



$$\mathbf{X} = \langle \mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_\mathbf{m} \rangle$$
$$\mathbf{Y} = \langle \mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_\mathbf{n} \rangle$$

$$\mathbf{C} = \mathcal{F}(\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_\mathbf{m})$$

$$\mathbf{y}_\mathbf{i} = \mathcal{G}(\mathbf{C}, \mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_{\mathbf{i}-1})$$

Simply put, the input X is used to summarize all the information, and then the decoder is used to analyze the information to obtain the result.

Problem?

The Encoder-Decoder model introduced above does not reflect the "attention model", so it can be regarded as a distraction model of inattention. Why do you say it is inattentive?

$$y_1 = f(C)$$
$$y_2 = f(C, y_1)$$
$$y_3 = f(C, y_1, y_2)$$

where f is the nonlinear transformation function of the decoder. From here, we can see that when generating the words of the target sentence, no matter which word is generated, y1,y2 or y3, they all use the same semantic encoding C of sentence X. There is no difference. This means that no matter which word is generated, y1, y2 or y3, any word in sentence X has the same impact on the generation of a target word yi without any difference.

Attention Model

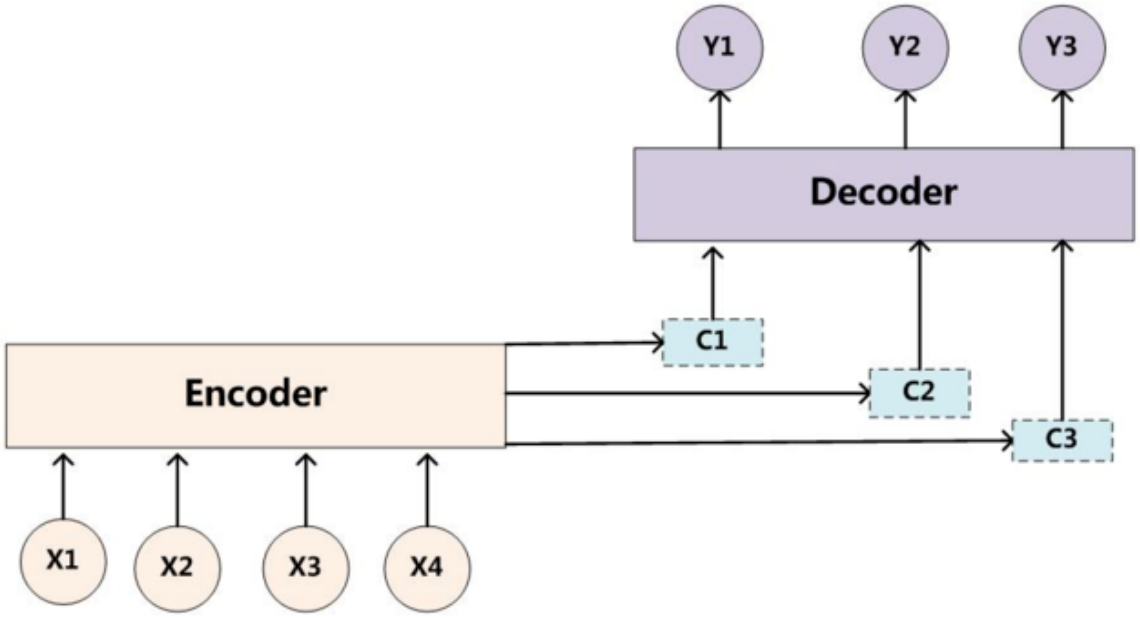
How Attention works

Take an example of translating an English sentence: Tom chase Jerry.

The translation of "Jerry" should reflect the different degree of influence of the English word on the translation of the current Chinese word, for example, by giving a probability distribution value similar to the following.

(Tom,0.3) (Chase,0.2) (Jerry,0.5)

The probability of each English word represents the amount of attention that the attention allocation model assigns to different English words when translating the current word "Jerry". This is certainly helpful for the correct translation of the target word, **because new information is introduced**. Similarly, each word in the target sentence should learn the attention allocation probability information of the word in its corresponding source sentence. This means that when generating each word Yi, the intermediate semantic representation C, which was originally the same, is replaced with a changing Ci depending on the currently generated word. the key to understanding the AM model is here, **the change from a fixed intermediate semantic representation C to a changing Ci that adjusts to the current output word to join the attention model.**



$$y_1 = f1(C_1)$$
$$y_2 = f1(C_2, y_1)$$
$$y_3 = f1(C_3, y_1, y_2)$$

$$C_{汤姆} = g(0.6 * f2("Tom"), 0.2 * f2(Chase), 0.2 * f2("Jerry"))$$

$$C_{追逐} = g(0.2 * f2("Tom"), 0.7 * f2(Chase), 0.1 * f2("Jerry"))$$

$$C_{杰瑞} = g(0.3 * f2("Tom"), 0.2 * f2(Chase), 0.5 * f2("Jerry"))$$

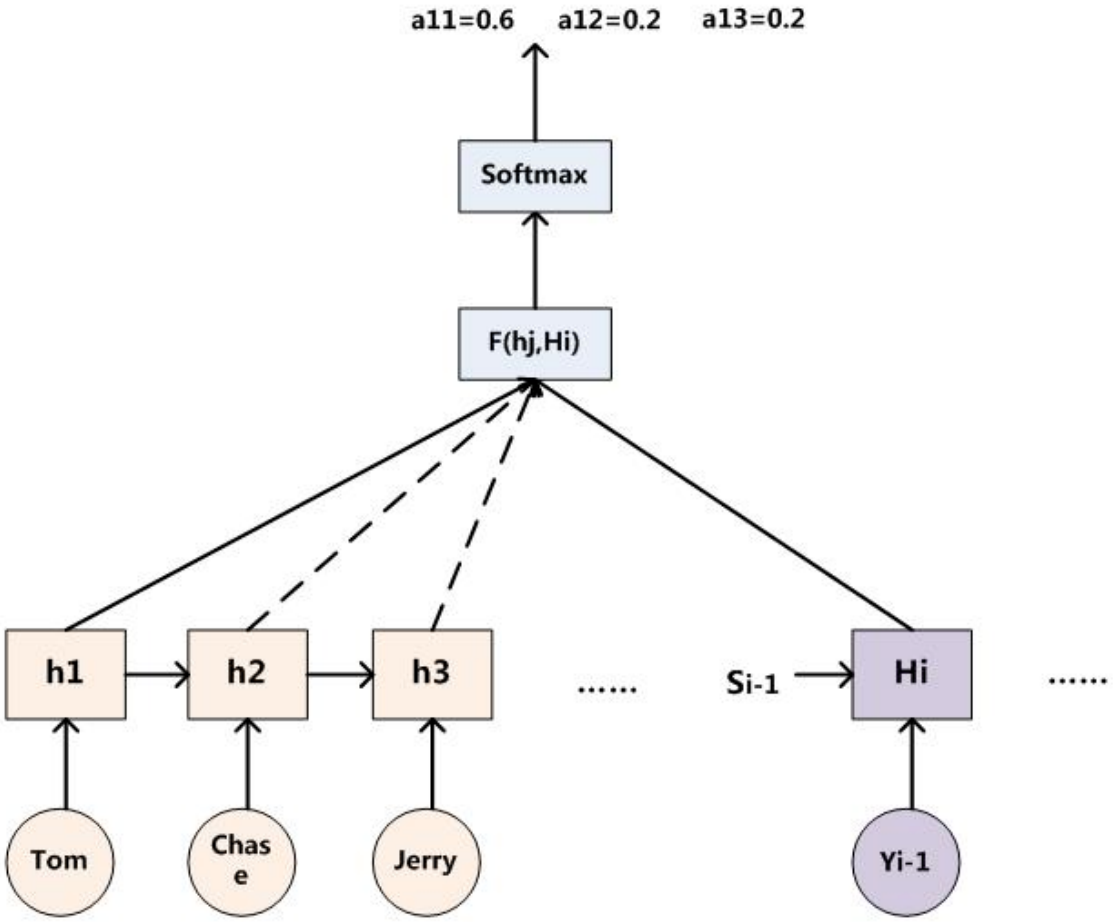
Among them, the f2 function represents some kind of transformation function of Encoder on the input English words, for example, if Encoder is using the RNN model, the result of this f2 function is often the state value of the hidden layer node after input xi at a certain moment; g represents the transformation function of Encoder to synthesize the middle semantic representation of the whole sentence according to the middle representation of the words, and in general practice, the g function is the weighted summation of the constituent elements, which is often seen in papers with the following formula.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Finally, there is only one problem: when generating a word for a target sentence, such as "Tom", how do you know the probability distribution of the input sentence word attention required by the AM model? That is, the probability distribution corresponding to "Tom".

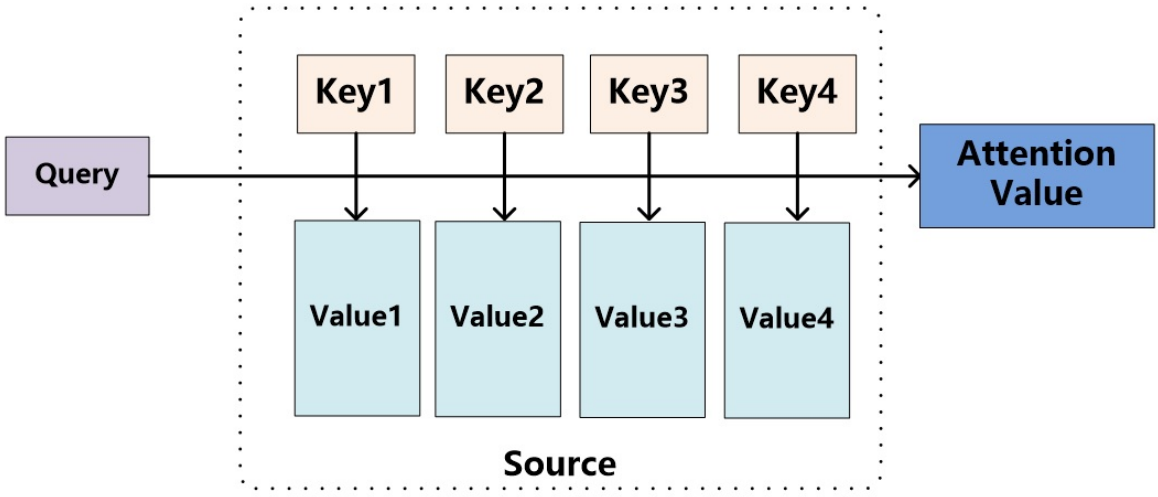
(Tom,0.3) (Chase,0.2) (Jerry,0.5) How do you get them?

For the Decoder using RNN, if we want to generate yi words, at moment i, we can know the output value Hi of the hidden layer node at moment i before generating Yi, and our purpose is to calculate the probability distribution of attention distribution of input sentence words "Tom", "Chase "Then we can use the state Hi of the hidden node at moment i to compare with the state hj of the RNN hidden node corresponding to each word in the input sentence one by one, i.e., we can use the function F(hj,Hi) to obtain the alignment probability of the target word Yi and each input word. possibility, this F function may take different approaches in different papers, and then the output of the function F is normalized by Softmax to obtain the value of the probability distribution of attention assignment that matches the probability distribution taking the value interval (this gives the attention weights).Picture shows the alignment probability of the words of the input sentence when the output word is "Tom". Most of the AM models adopt the above computational framework to calculate the information of the attention distribution probability, but the difference may only be in the definition of F.



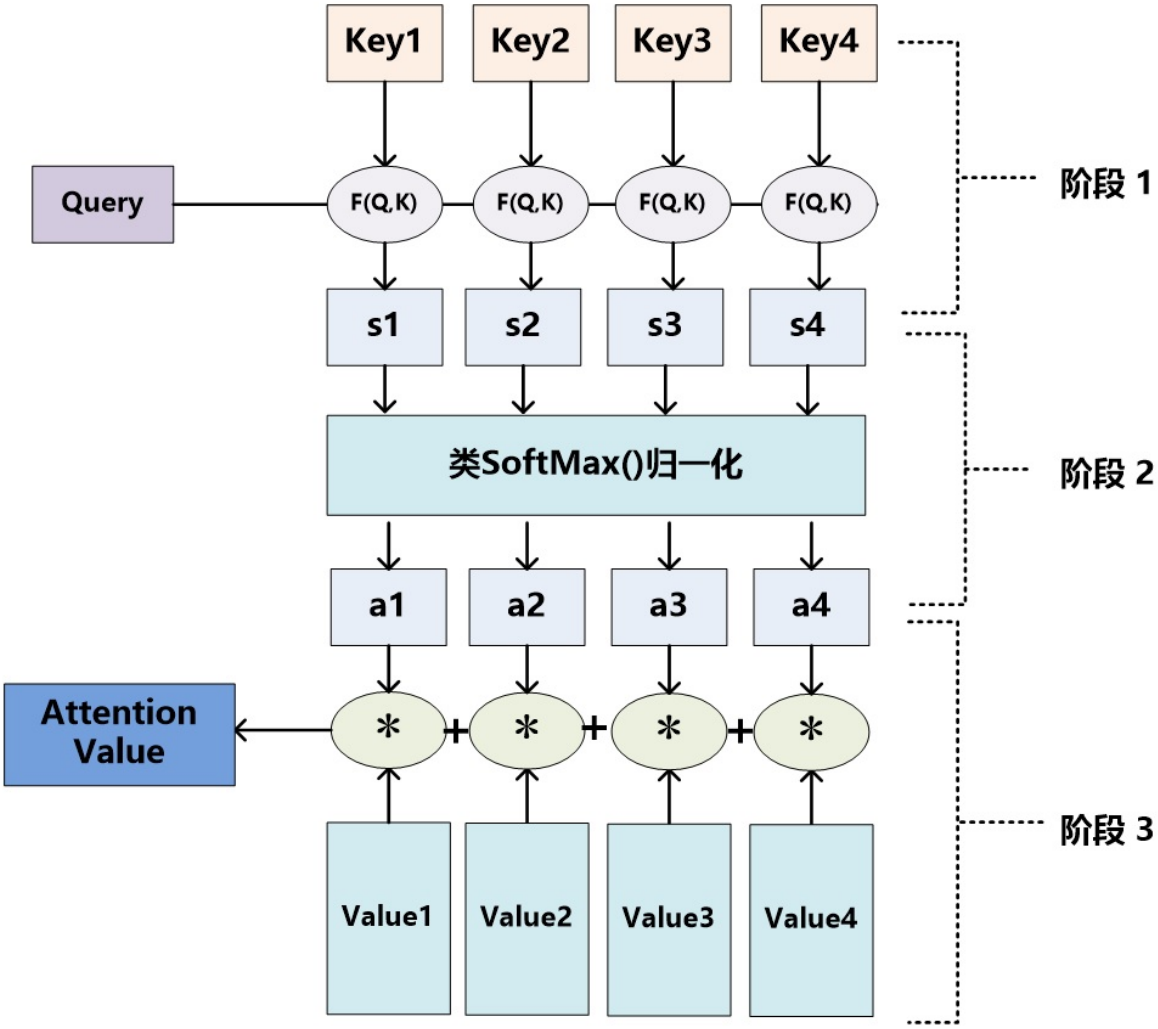
More and More Thinking

If the Attention mechanism is stripped from the Encoder-Decoder framework in the above example, and further abstracted, it is easier to understand the essence of the Attention mechanism.



We can look at the Attention mechanism in this way (refer to Figure): Imagine the constituent elements in the Source as a series of <Key,Value> data pairs, and given a certain element Query in the Target, the weight of each Key corresponding to Value is obtained by calculating the similarity or correlation between Query and each Key. The final Attention value is obtained by calculating the similarity or relevance of the Query and each Key, and then weighting and summing the Values. So essentially the Attention mechanism is a weighted sum of the Value values of the elements in the Source, and the Query and Key are used to calculate the weight coefficients of the corresponding Value. That is, the essence of the idea can be rewritten as the following formula.

$$\mathbf{Attention}(\mathbf{Query}, \mathbf{Source}) = \sum_{i=1}^{L_x} \boldsymbol{Similarity}(\boldsymbol{Query}, \boldsymbol{Key}_i) * \boldsymbol{Value}_i$$


Key1
Key2
Key3
Key4
Query
F(Q,K)
F(Q,K)
F(Q,K)
F(Q,K)
s1
s2
s3
s4
类SoftMax()归一化
a1
a2
a3
a4
Attention Value
*
*
*
*
Value1
Value2
Value3
Value4
阶段 1
阶段 2
阶段 3