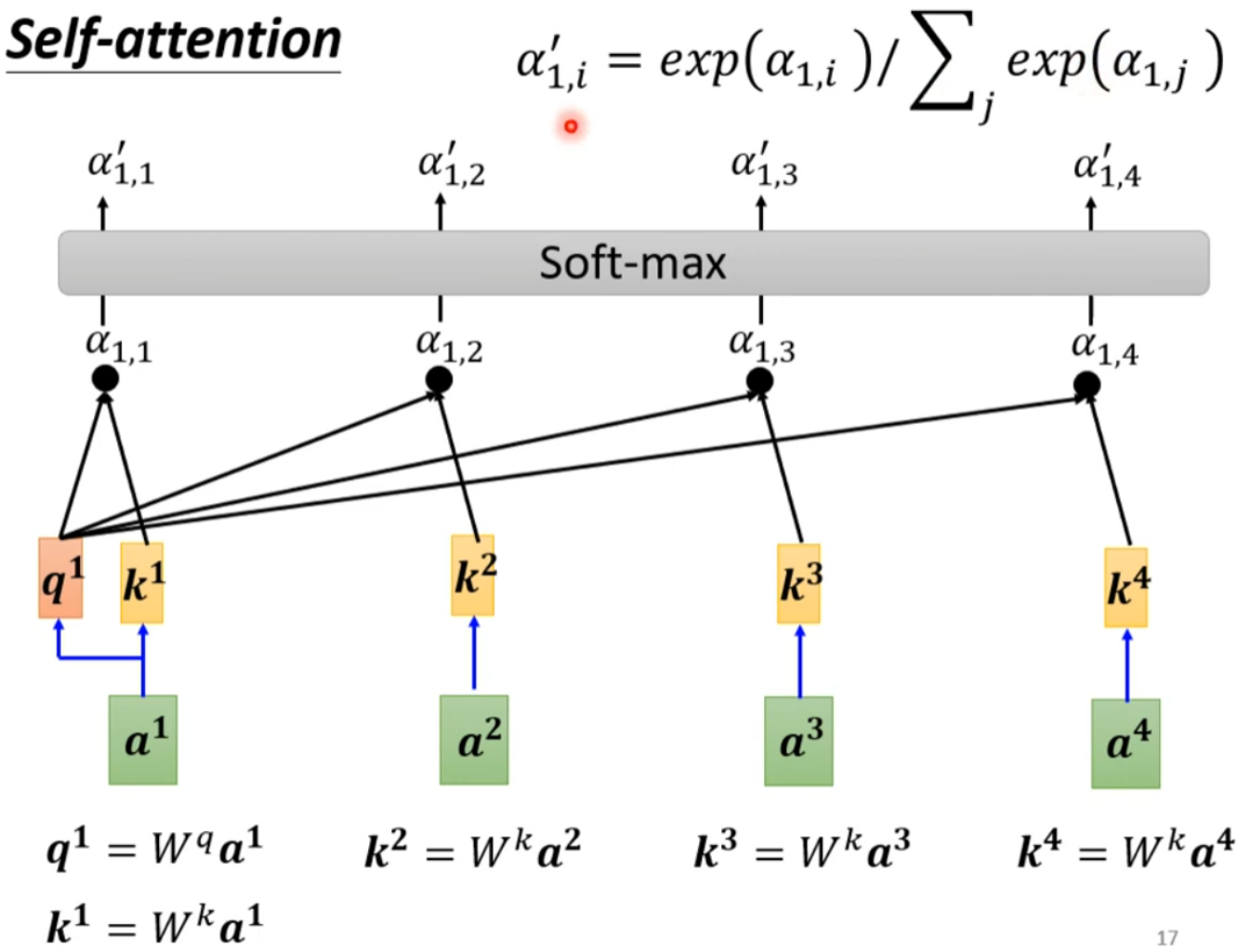
Self-attention

引入

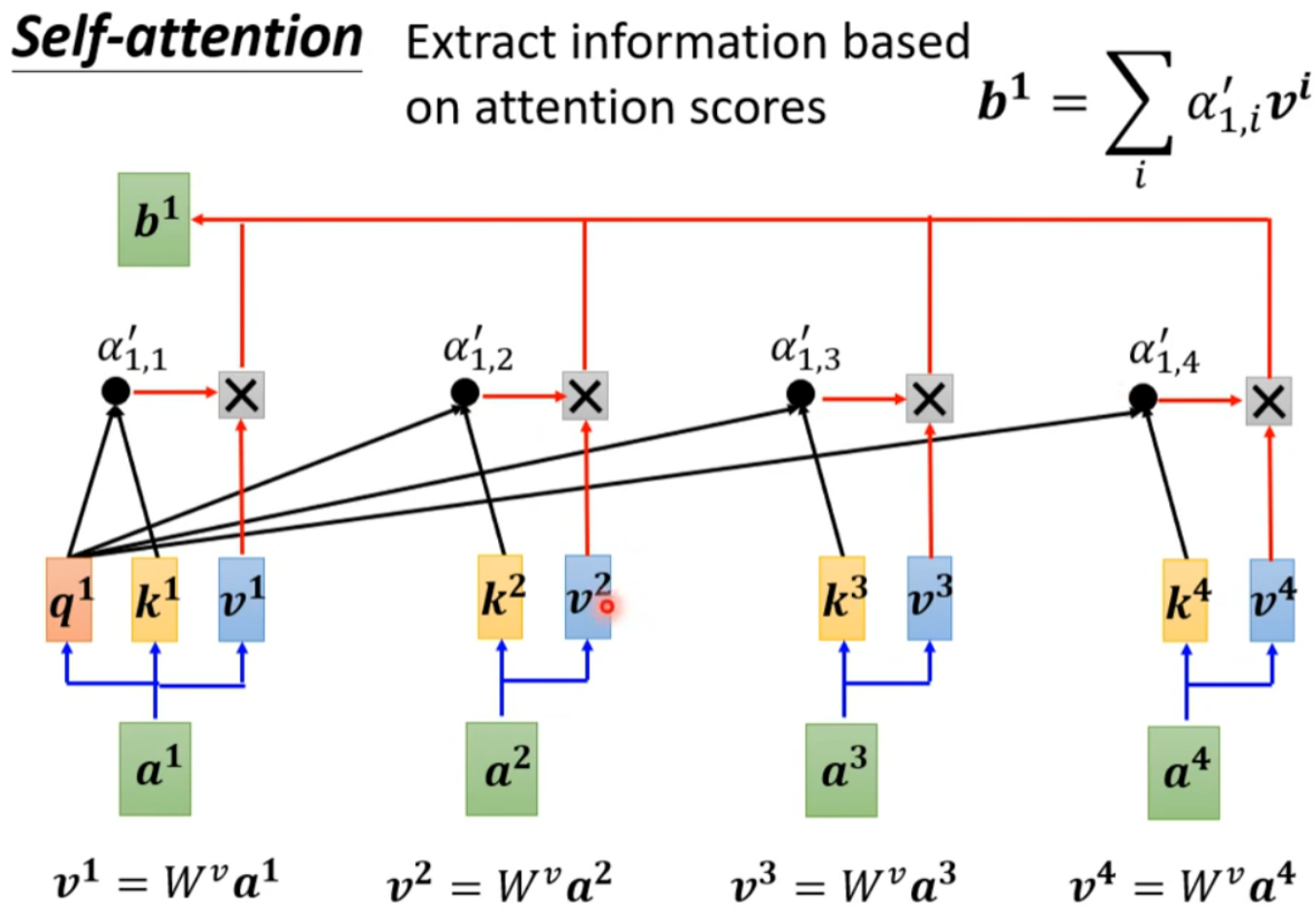
在Sequence Labeling中，只对一个token单独分析不合理，而是要上下文联系。如果只是用一个windows处理又不能满足变长的输入的需求。

自注意力（Self-attention）：考虑整个序列上下文后的输出

以第一个token（α^1）为例，首先要找出序列中其他输入和a^1的相关性α，首先算出a^1作为token的关键字q^1，把每一个向量和W^k相乘算出key，利用query和key的值相乘算出注意力分数（关联性）。



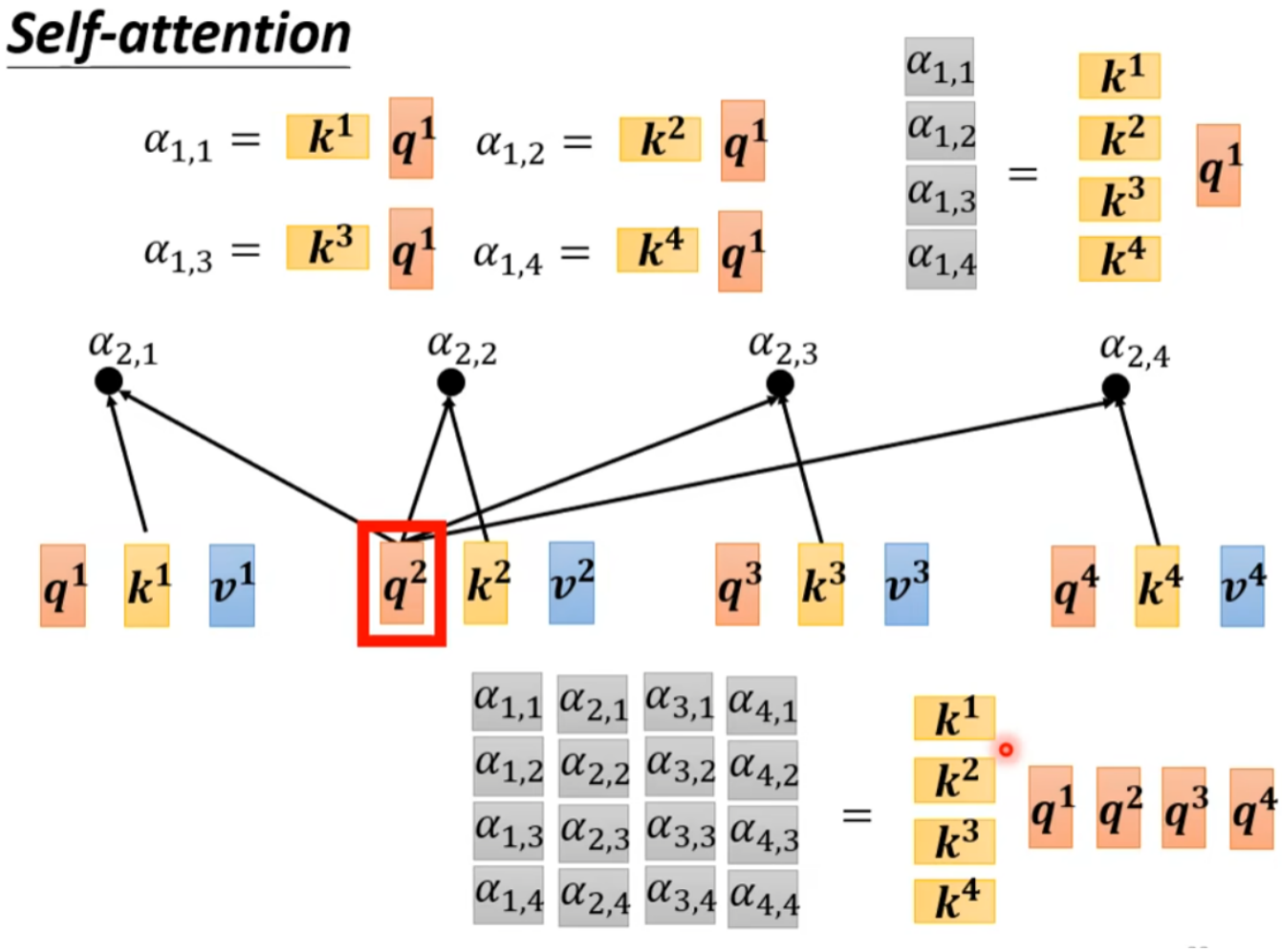
这样就知道那些向量和a^1有关，接下俩就要根据关联性抽取信息。把每个token和W^v相乘算出抽取信息，再和关联性α相乘相加就可以得到输出b。



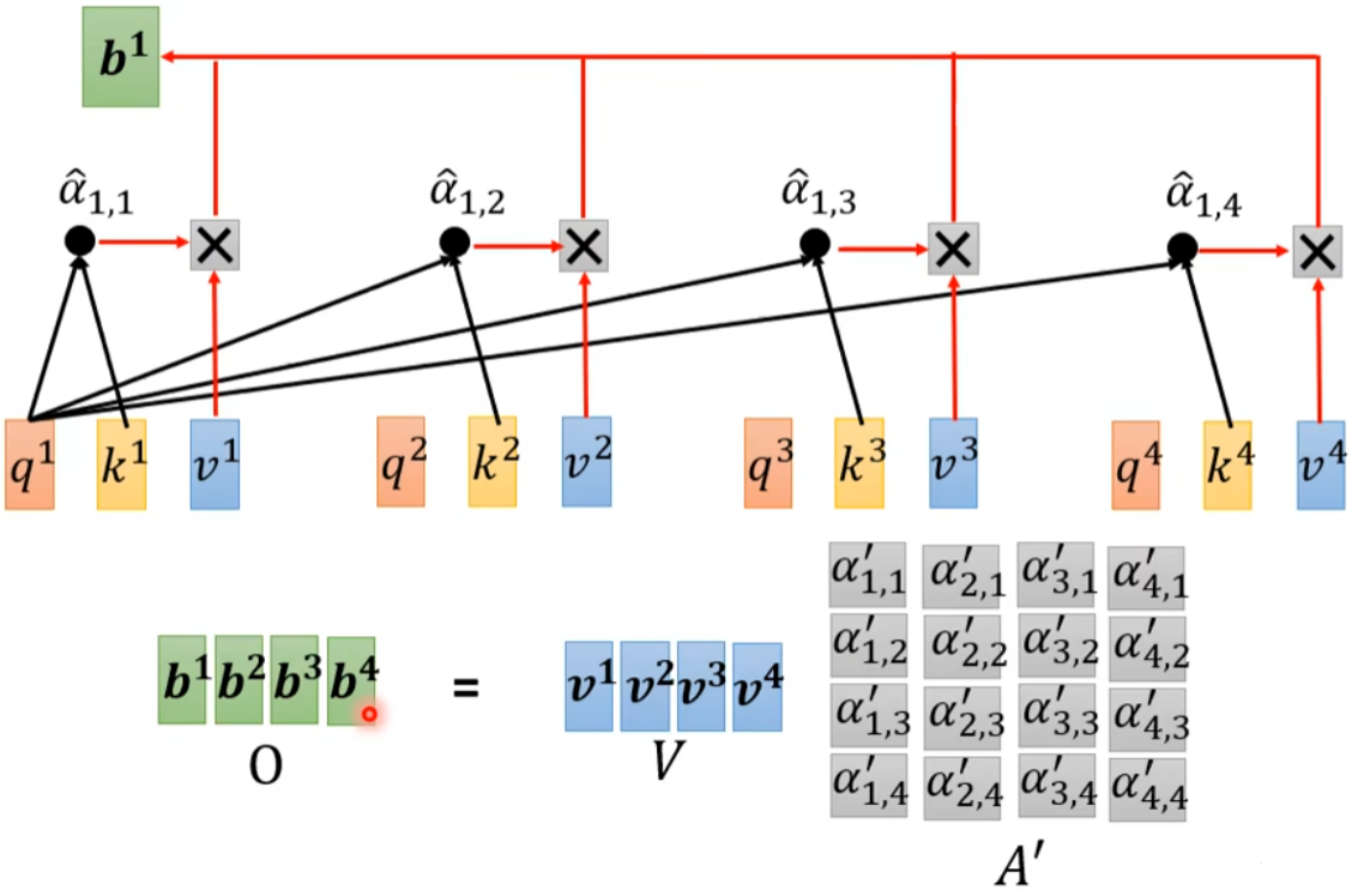
每一个token的b是并行同时产生的：

由于$q^i = W^q a^i, k^i = W^k a^i, v^i = W^v a^i$，那么把$a^i$拼接起来，就可以得到矩阵$I$。

那么我们就可以算出$Q = W^q \times I, K = W^k \times I, V = W^v \times I$。



那么就可以得到注意力矩阵$A' = softmax(K^T \times Q)$,此时再和$V$相乘就可以算出输出矩阵$O = V \times A'$



那么此时，我们可以看到未知参数只有W^q, W^k, W^v。

- 多头注意力机制：如果需要提取token不同的q关键字（观察不同的特征），那么就利用矩阵算出不同的Q以及对应K, V，最后利用输出权重W^O计算出输出O

位置编码（Positional Encoding）

以上的做法并没有对于位置顺序的考虑，因此需要编码位置信息。处理时直接把位置编码和相关性加起来$e^i + a^i$，目前没有最好的结论。

网络结构对比

- Self-attention和CNN

CNN是Self-attention的特例，CNN的弹性比较小，Self-attention的弹性比较大，因此Self-attention在数据量比较大的时候表现更好。

- Self-attention和RNN

RNN是单向，Self-attention是双向。RNN序列容易遗忘之前跨度较大的token内容，而Self-attention是利用矩阵计算的，不会忽略。且由于RNN是单项的，输出只能从头到尾，不能并行计算。

用于图的自注意力

只计算有边的权重，是一种特殊的GNN