1. 背景：为解决在基于 Seq2Seq 模型的机翻模型中输入和输出难以对齐以及应对长文本效果不佳等问题。
2. 基本思想：在生成译文的每个词项时，计算出源文本词项序列的权重分布，找出源文本序列中和当前词项**最相关**的元素，使模型在预测时更具有针对性。
3. 具体算法：先将编码器生成的源文本隐层序列$(h_1, h_2, ..., h_n)$和上一时间步的解码器隐层向量s_{t-1}进行匹配，计算隐层序列的权重分布$(a_{t1}, a_{t2}, ..., a_{tn})$；之后将$a_{ti}$和$h_i$加权求和得到带注意力的语义向量$c_t$；解码器在每个时间步根据动态变化的$c_t$逐个生成译文词项。该工作将注意力机制作为连接编码器和解码器的桥梁，实现了翻译过程中的词对齐，提高了模型生成译文的质量。
4. 注意力机制的通用形式：在上述任务中，包含已生成的译文信息的解码器隐层向量s_{t-1}代表下游任务，下游任务在不同任务中有不同的表达形式；将下游任务抽象成查询(Query, $Q = (q_1, q_2, ..., q_M)$)，将源文本看成键(Key, $K = (k_1, k_2, ..., k_N)$)-值(Value, $V = (v_1, v_2, ..., v_N)$)对序列。**针对q_t的注意力可以被描述成键值对序列在该查询上的映射。**

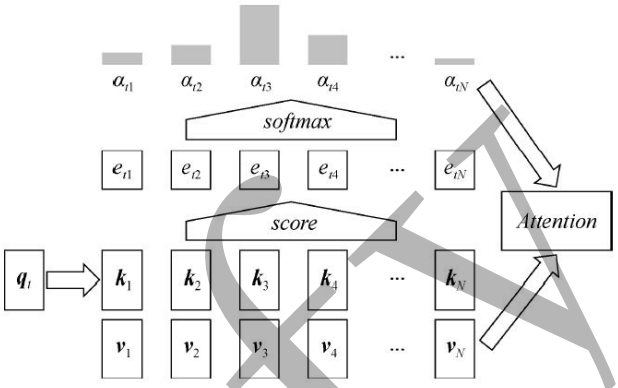

图 2　注意力机制的通用形式

Fig.2　The General Form of Attention Mechanism

计算过程：

（1）　计算查询q_t和每个键k_i的注意力得分e_{ti}，常用的计算方法包括点积、缩放点积、拼接以及相加等。

$$e_{ti} = score(\boldsymbol{q}_t, \boldsymbol{k}_i) = \begin{cases} \boldsymbol{q}_t^{\mathrm{T}} \boldsymbol{k}_i & \text{点积} \\ \boldsymbol{q}_t^{\mathrm{T}} \boldsymbol{W} \boldsymbol{k}_i & \text{通用} \\ \dfrac{\boldsymbol{q}_t^{\mathrm{T}} \boldsymbol{k}_i}{\sqrt{d_k}} & \text{缩放点积} \\ \boldsymbol{v}^{\mathrm{T}} \tanh(\boldsymbol{W}[\boldsymbol{q}_t; \boldsymbol{k}_i]) & \text{拼接} \\ \boldsymbol{v}^{\mathrm{T}} \tanh(\boldsymbol{W}\boldsymbol{q}_t + \boldsymbol{U}\boldsymbol{k}_i) & \text{相加} \end{cases} \quad (1)$$

（2）使用 Softmax 等函数对注意力得分做归一化处理，得到每个键的权重a_{ti}。

$$\alpha_{ti} = softmax(e_{ti}) = \frac{\exp(e_{ti})}{\sum_{n=1}^{N} \exp(e_{tn})} \quad (2)$$

（3）　将权重a_{ti}和其对应的值v_i加权求和作为注意力输出。

$$Attention(\boldsymbol{q}_t, \boldsymbol{K}, \boldsymbol{V}) = \sum_i \alpha_{ti} \boldsymbol{v}_i \quad (3)$$

5. 键值对是源文本的组成元素，可以是字符、词、短语、句子等，甚至是它们的组合，**通常情况下 K=V**。

6. 注意力机制的分类：

关注的范围：

表 1　注意力机制按照关注范围分类

Table 1　Classification of Attention Mechanism by Attention Range

注意力	关注范围
全局注意力	全部元素
局部注意力	以对齐位置为中心的窗口
硬注意力	一个元素
稀疏注意力	稀疏分布的部分元素
结构注意力	结构上相关的一系列元素

全局注意力(Global Attention)：对源文本序列中的每个元素都计算其针对查询的得分和权重。计算开销较大，且随着源文本长度的增加其效果会变弱。

局部注意力(Local Attention)：

（1）具体实现：先基于查询q_t预测一个源文本中的对齐位置p_t，并以p_t为中心确定窗口$[p_t - D, p_t + D]$，后续步骤和全局注意力相似，不仅要计算窗口内部元素的权重分布，最后用高斯分布加强p_t附近的权重。

$$p_t = L \cdot sigmoid\left(\boldsymbol{v}^{\mathrm{T}} \tanh\left(\boldsymbol{W}\boldsymbol{q}_t\right)\right) \tag{4}$$

$$\alpha_t'(l) = \alpha_t(l) \cdot \exp\left(-\frac{(l - p_t)^2}{2\sigma^2}\right) \tag{5}$$

其中，L代表源文本序列的长度，l代表窗口内元素的位置序号，标准差$\sigma = \frac{D}{2}$。

（2） 有实验证明，局部注意力可以在更小的计算开销下获得**更准确**的译文，同时说明全局注意力可能会带来**噪声**干扰；局部注意力依赖于对齐位置，对齐位置的预测并不可靠，会给模型带来更多的不确定性。

（3） 硬注意力(Hard Attention)：局部注意力中 D=0 时，仅关注文本中一个元素的意义十分有限。

（4） 注意力计算时通常采用 Softmax 函数计算出元素的权重$a_{ti} \neq 0$，即使和查询无关的元素也会被分配一个很小的权重，致使所有元素对预测都有或多或少的影响，在实际应用过程中会导致**注意力分散**等弊端。因此提出用 Sparsemax 函数代替 Softmax，

$$Sparsemax(e) = \underset{\alpha \in \Delta^{K-1}}{\operatorname{argmin}} \|\alpha - e\|^2 \tag{6}$$

Sparsemax 函数返回的是注意力得分 e 在 $K-1$ 维概率单纯形上的欧几里得投影，这些投影倾向于触及单纯形的边界，从而导致部分概率值会降为 0，进而得到稀疏的概率分布。Sparsemax 函数在保留 Softmax 大部分重要性质的同时具有产生稀疏分布的能力，可以使注意力机制只关注源文本序列中的部分元素，理论上使模型的注意力更加集中，但绝大多数研究依然采用 Softmax 函数，Sparsemax 函数仅在多标签分类任务中体现出优势[12]。

（5）无论是 Softamax 还是 Sparsemax 都没有考虑输出层的结构信息，即关注的元素只有权

重大小之分，没有上下文的关系，如子序列关系或句法依赖关系，故提出**结构注意力**（Structured Attention），用条件随机场（CRF）重建注意力机制。CRF 不同于 Softmax，其计算出的不是各个元素的概率，而是一条路径的概率，使得模型可以选择出一系列具有关联的元素。

组合方式：层级注意力、双向注意力、多头注意力

层级注意力：

（1） 不仅要考虑词项对分类的影响，还考虑到**词项所处的上下文**结构信息。

（2） 即将源文本分割为片段，分别计算每个片段的注意力，再基于片段的注意力计算全文的注意力，并以此作为源文本的最终表示。

（3） 层级注意力可以更好地捕捉源文本的结构特征。层级注意力机制常被用于长文本任务中。

（4） 在实际应用中，以将源文本划分为不同粒度的区块（chunk），常见的区块包括句子、段落、语篇等。

（5） 层级注意力在具体实现上有两种方式：一种是分别计算出元素和区块的权重分布，之后用区块权重对元素权重进行缩放（scale），再基于缩放后的元素权重计算源文本注意力（公式 7）。另一种是先基于元素计算出每个区块的注意力（公式 8），并将其作为区块的键和值，再基于区块计算源文本注意力（公式 9）。

$$\alpha_{mi}^{scale} = \frac{\alpha_{mi}\alpha_m}{\sum_{s,w}\alpha_{sw}\alpha_s} \tag{7}$$

$$\boldsymbol{c}_m^{chunk} = Attention(\boldsymbol{q}, \boldsymbol{K}_m, \boldsymbol{V}_m) \tag{8}$$

$$\boldsymbol{c}^{doc} = Attention(\boldsymbol{q}, \boldsymbol{C}^{chunk}, \boldsymbol{C}^{chunk}) \tag{9}$$

其中，α_m代表第m个区块的权重，α_{mi}代表第m个区块中的第i个元素的权重，s代表区块数，w代表每个区块内的元素数，$\boldsymbol{K}_m$和$\boldsymbol{V}_m$分别代表第m个区块中的键、值矩阵，$\boldsymbol{C}^{chunk}$代表所有区块的注意力组成的矩阵。

双向注意力：

（1）通常注意力只基于 Q 计算 K 的注意力(K2Q)，但当 Q 是文本序列时，也可以基于 K 计算 Q 的注意力(Q2K)。将 K2Q 与 Q2K 加以组合，但为其赋予不同的名称，如双向注意力(Bi-Directional Attention)、协同注意力(Co-Attention)、双路注意力(Two-Way Attention)、交互注意力(Interactive Attention)、互注意力(Inter-Attention)等。

（2）为更好地理解两个序列间的关系，通常方法是将 K2Q 与 Q2K 的注意力输出进行**拼接或聚合**得到最终的输出。

（3）按照计算粒度分：

粗粒度是将其中一个序列压缩成向量，并作为查询，与另一个序列中的每个元素进行匹配，其计算过程和全局注意力一致，两个方向均是如此。

细粒度则是将两个序列中的所有元素相互匹配，得到注意力得分矩阵，之后分别按行和列对矩阵进行池化（pooling），得到两个序列的得分向量，再按公式（2）和公式（3）分别计算两个方向的注意力。

（4）按照计算顺序分：

并行（parallel）：对两个序列的注意力计算完全对称且同时进行

交替（alternating）：交替方式则是先基于序列 Q 计算出序列 K 的注意力，再基于已生成的序列 K 的注意力计算序列 Q 的注意力

多头注意力：

文本序列中每个元素的键向量和值向量都是高维向量，向量中的每个分量代表着元素不同方面的特征。当向量维度过高时，一次注意力计算很难充分捕捉元素的全部特征。

多头注意力：对文本序列并行做多次注意力计算，每次作为一个“头”。假设有 h 个头，每个头在计算时通过线性变换将 Q、K 以及 V 的维度降为原先的 1/h，即转换到一个子空间中。如公式（10）所示，线性变换的参数是可学习的，且每个头的参数不同，从而确保模型从不同的表示子空间学习到相关的特征。。最后将 h 个头的注意力结果进行拼接，再通过一次线性变换得到最终的注意力输出，如公式（11）所示。

$$head_i = Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V) \quad (10)$$

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, \cdots, head_h)\boldsymbol{W}^O \quad (11)$$

其中，参数矩阵 $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$，$\boldsymbol{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$，$\boldsymbol{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$，$\boldsymbol{W}_i^O \in \mathbb{R}^{hd_v \times d_{model}}$，$d_{model}$是元素向量的维度，$d_k = d_v = d_{model}/h$。多头注意力类似于集成学习，每个头只负责输出 1/h的文本特征，且相互独立。将

多头注意力机制的不足：

一是缺乏一种机制保证不同的头能切实捕捉到不同的特征。

二是用简单的拼接加线性变换对多个头的输出进行聚合，可能无法充分发挥多头注意力的表示能力。

针对第一个问题，Li 等[24]引入不一致正则化（disagreement regularization）项，作为模型训练的辅助目标，以鼓励多头之间的多样性。

针对第二个问题，Li 等[25]借鉴胶囊网络(CapsNet)中的协议路由(Routing-by-Agreement)算法[26]以改善多头注意力的信息聚合。协议路由算法可以个输入胶囊中的信息进行分配，并整合 N 个输出胶囊中。该算法根据输入胶囊和输出胶囊的一致性来迭代更新每个输入胶囊应该分配给每个输出胶囊的比例，最终将所有输出胶囊连在一起形成最终表示。

自注意力：

（1）注意力机制一般基于外部查询对源文本进行解读，试图**找到和查询最匹配的元素**，然而这种做法忽略了源文本自身的特征，即源文本内部元素间的关系。

（2）自注意力(Self-Attention)[27]是注意力机制在 **Q=K** 时的一种改进，即查询来自源文本序列自身，用于建模源文本序列内部元素间的依赖关系，以加强对源文本语义的理解。

（3）Cheng 等[22]提出的内部注意力(Intra-Attention)是自注意力的思想启蒙，因此自注意力有时也被称为内部注意力。

（4）**为了加强句子的语义表示**，Lin 等[27]提出利用自注意力进行句子嵌入(sentence embedding)。首先通过双向 LSTM 获得句子的隐层序列$H = (h_1, ..., h_N)$，之后采用自注意力计算句子所有元素间的权重矩阵 A，进而得到句子的矩阵表示 M=AH。

$$\boldsymbol{A} = softmax(\boldsymbol{W}_1 \tanh(\boldsymbol{W}_2\boldsymbol{H}^T)) \quad (12)$$

不同于将句子嵌入为固定向量的传统做法，该工作利用自注意力将句子嵌入为矩阵，矩阵的每一行体现了句子针对相应元素的语义特征。

（5）Shen 等[28]基于查询的粒度将自注意力分为 Token2Token 和 Source2Token。

（6）Token2Token(细粒度):解析了文本序列中任意两个元素间的依赖关系，得到文本的矩阵表示。

（7）Source2Token(粗粒度): 将整个文本序列压缩为查询向量，然后和每个元素进行匹配，探索每个元素和整个文本间的全局依赖关系，得到文本的向量表示。

自注意力网络：

（1）自注意力具有直接建模文本元素间关系的能力，无需循环或卷积神经网络，依靠自注意力完成文本的编码，并据此构建自注意力网。该网络由 6 个相同的层堆叠在一起，每个层由多头自注意力和全连接前馈网络(FNN)构成。

（2）自注意力网络直接基于词向量计算自注意力，在捕捉词项间依赖关系的同时完成编码。

$$SelfAttention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}})\boldsymbol{V} \quad (13)$$

（3）考虑到**语境信息**对建模的重要性，在预处理阶段采用位置嵌入(position embedding)为每个元素添加绝对位置信息以提升建模效果。

（4）相对位置：Shaw 等[29]认为自注意力网络中嵌入的绝对位置不如相对位置高效，提出在计算注意力时考虑元素的相对位置，即元素间的距离。

$$\boldsymbol{z}_i = \sum_{j=1}^{n} softmax(\frac{(\boldsymbol{x}_i\boldsymbol{W}^Q)(\boldsymbol{x}_j\boldsymbol{W}^K + \boldsymbol{a}_{ij}^K)^{\mathrm{T}}}{\sqrt{d_z}})(\boldsymbol{x}_j\boldsymbol{W}^V + \boldsymbol{a}_{ij}^V) \quad (14)$$

其中，$\boldsymbol{a}_{ij}^K$和$\boldsymbol{a}_{ij}^V$分别代表元素$\boldsymbol{x}_j$的键和值相对于元素$\boldsymbol{x}_i$的位置，其值是元素间距离的向量表示。该

该工作假设在一定距离之外精确的相对位置信息没有意义，因此将元素间的最大距离限制为 b，即元素间的距离超过 b 时将被裁(clip)为 b。最大距离限制还可以使模型更好地适应训练时未见过的序列长度。

（5）局部建模：相对位置让模型在计算注意力时考虑到元素间的距离，但没有针对性地加强邻近元素间的依赖关系。Yang 等[10]认为提高邻近元素间的依赖关系有利于捕捉到有用的短语结构，因此提出利用可学习的高斯偏差(Gaussian Bias)来强化局部权重分布。

$$\alpha_{ij} = \frac{\exp(e_{ij} + \boldsymbol{G}_{ij})}{\sum_{k=1}^{n}\exp(e_{ik} + \boldsymbol{G}_{ik})} \quad (16)$$

权重分布，如公式(16)所示。$\boldsymbol{G}$是一个掩码(mask)矩阵，其元素$\boldsymbol{G}_{ij} \in [0, -\infty)$，代表元素$\boldsymbol{x}_j$和需要得到更多关注的局部区域的中心位置的紧密程度，由预测得到的中心位置和窗口尺寸计算得到。实验证实相对位置和局部建模相结合可以进一步提升模型的表现。

（6）定向：自注意力网络允许一个元素和其前后的任何元素进行匹配，但在一些自然语言处理任务中需要考虑序列的时间顺序。对此，Shen 等[28]提出定向自注意力网络(Directional Self-Attention Network，DiSAN)以限定注意力的计算方向（前向或后向），使自注意力能像循环神经网络一样按时间顺序处理文本序列。将位置掩码(positional mask)矩阵和自注意力得分矩阵相加来限制计算方向。

向。位置掩码分为前向掩码和后向掩码。前向掩码矩阵中$i < j$的元素为0，其他为$-\infty$；和得分矩阵相加后，得分矩阵中$i \geqslant j$的元素会变成$-\infty$，其他元素不变；经Softmax函数归一化后$i \geqslant j$的元素权重为0，意味着不关注；此时元素$\boldsymbol{x}_i$只和比自己序号大的元素$\boldsymbol{x}_j$进行匹配，实现了注意力的前向计算；反之亦然。