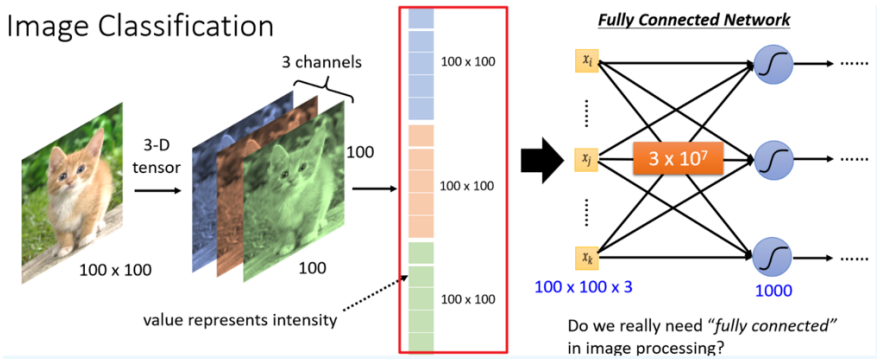
Seq2Seq 模型

1. 引言

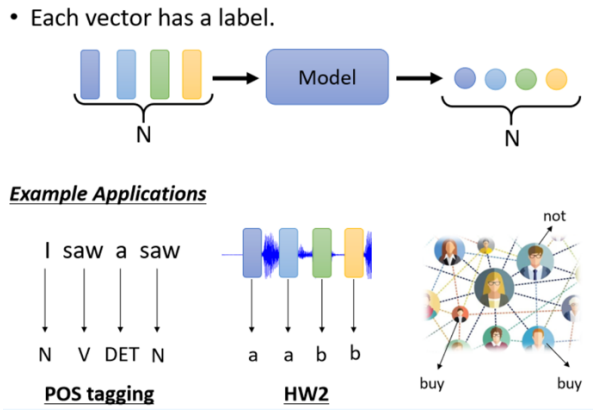
1.1 Seq2Seq 的由来

- 在 Seq2Seq 框架提出之前，深度神经网络在图像分类等问题上取得了非常好的效果。在其擅长解决的问题中，**输入和输出通常都可以表示为固定长度的向量**，如果长度稍有变化，会使用**补零**等操作。(如最原始 CNN 的输入是图像经过 flatten 后的向量，下图红框部分，往往将数据处理成固定大小的向量作为输入)

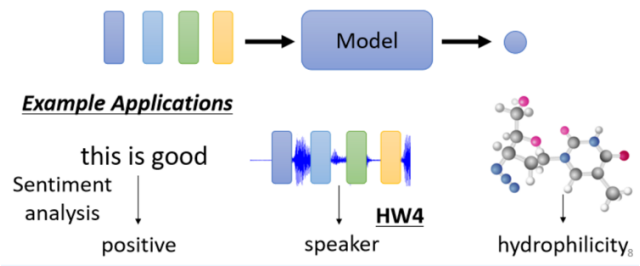


- 然而许多重要的问题，例如机器翻译、语音识别、自动对话等，表示成序列后，**其长度事先并不知道**。因此如何突破先前深度神经网络的局限，使其可以适应这些场景，成为了 13 年以来的研究热点，Seq2Seq 框架应运而生。
- 当输入是**多个向量**，而且这个**输入向量的数目是会改变的场景**时，Decoder 的输出可以有以下三种形式

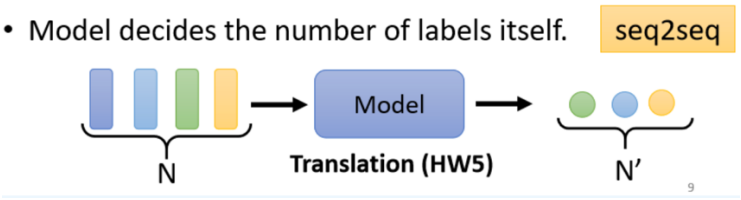
a) 输出个数与输入向量个数相同，即每一个向量都有对应的一个 label 或 value (如命名实体识别 NER、词性标注 POS tagging 等任务)，也叫 Sequence Labeling。



b) 只需要输出一个 Label 或 value (比如文本分类、情感分析)。



c) 输出个数与输入向量个数不一定相同，机器要自己决定应该要输出多少个 Label 或 value（比如文本翻译、语音识别），也叫做 **Sequence to Sequence(Seq2Seq)**的任务。

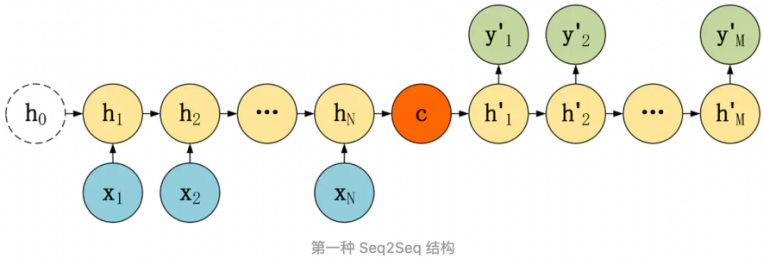


1.2 Seq2Seq 和 Encoder-Decoder 的关系

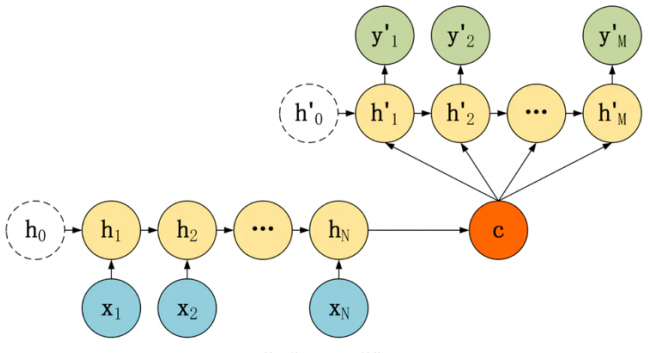
- Seq2Seq 使用的具体方法基本都属于 Encoder-Decoder 模型（强调方法）的范畴，Seq2Seq（强调目的）不特指具体方法，满足"输入序列、输出序列"的目的，都可以统称为 Seq2Seq 模型。
- Seq2Seq 模型是基于 Encoder-Decoder 框架设计的，用于解决序列到序列问题的模型。

2. Seq2Seq 模型

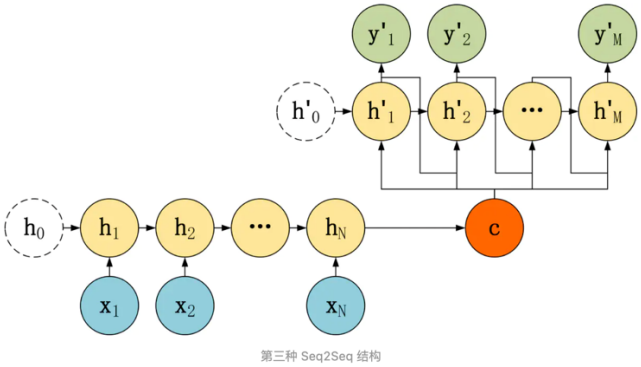
- Sequence-to-Sequence (Seq2Seq) 模型，其输入是一个序列，输出也是一个序列。**其最重要的地方在于输入序列和输出序列的长度是可变的。** 最基础的 Seq2Seq 模型包含了三个部分，即 Encoder、Decoder 以及连接两者的中间语义向量C，Encoder 通过学习输入，将其编码成一个固定大小的向量C，继而将C传给 Decoder，Decoder 再通过对状态向量C的学习来进行输出。
- 常见结构:



第一种 Seq2Seq 结构



第二种 Seq2Seq 结构



第三种 Seq2Seq 结构

- Seq2Seq 模型缺点

  Seq2Seq 模型缺点包括了 RNN 模块存在的缺点，和基础 Encoder-Decoder 框架存在的问题

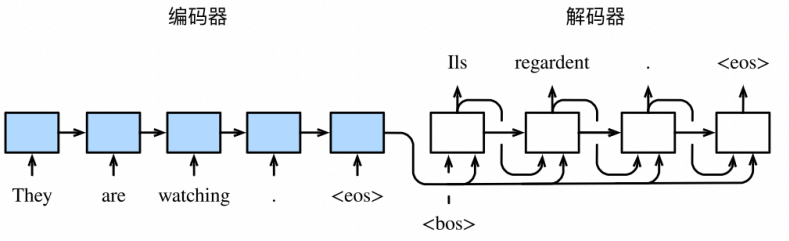
  a) 中间语义向量C无法完全表达整个输入序列的信息。

  b) 中间语义向量C对输出$y_1, y_2, \ldots, y_m$所产生的贡献都是一样的，即分配到的权重是相同的。

  c) 随着输入信息长度的增加，先前编码好的信息会被后来的信息覆盖，丢失很多信息。

3. 详解 Seq2Seq

- 下图描述了使用编码器—解码器将上述英语句子翻译成法语句子的一种方法。在训练数据集中，我们可以在每个句子后附上特殊符号“**<eos>**”（end of sequence）以表示序列的终止。编码器每个时间步的输入依次为英语句子中的单词、标点和特殊符号“<eos>”。图中使用了**编码器在最终时间步的隐藏状态作为输入句子的表征或编码信息。解码器在各个时间步中使用输入句子的编码信息和上个时间步的输出以及隐藏状态作为输入**。我们希望解码器在各个时间步能正确依次输出翻译后的法语单词，标点和特殊符号"<eos>"。需要注意的是，解码器在最初时间步的输入用到了一个表示序列开始的特殊符号"**<bos>**"（beginning of sequence）。



3.1 编码器(以 RNN 为例)

- 作用是把一个不定长的输入序列变换成一个定长的语义向量C，并在该语义向量中编码输入序列信息。
- 具体过程

  a) 假设输入序列是$x_1, x_2, \ldots, x_T$，x_i表示输入句子的第i个词。在时间步t，RNN将输入x_t的特征向量$\boldsymbol{x_t}$和上个时间步的隐藏状态$\boldsymbol{h_{t-1}}$更换到当前时间步的隐藏状态$\boldsymbol{h_t}$，用函数f表示为:

$$\boldsymbol{h_t} = f(\boldsymbol{x_t}, \boldsymbol{h_{t-1}})$$

b) 接下来，编码器通过自定义函数q将各个时间步的隐藏状态变换为语义向量C

$$\boldsymbol{C} = q(\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots \boldsymbol{h_T})$$

- 以上描述的编码器是一个单向的循环神经网络，每个时间步的隐藏状态只取决于该时间步及之前的输入子序列。我们也可以使用双向循环神经网络构造编码器。在这种情况下，编码器每个时间步的隐藏状态同时取决于该时间步之前和之后的子序列（包括当前时间步的输入），并编码了整个序列的信息。

3.2 解码器(以 RNN 为例)

- 给定训练样本中的输出序列$y_1, y_2, \ldots, y_{T'}$，对每个时间步t'（符号与编码器的时间步t有区别），解码器输出$y_{t'}$的条件概率将基于之前对输出序列$y_1, y_2, \ldots, y_{t'-1}$和语义向量$C$，即:

$$P(y_{t'} | y_1, y_2, \ldots, y_{t'-1}, \boldsymbol{C})$$

- 为此可以使用另一个循环神经网络作为解码器。在输出序列的时间步t'，解码器将上一时间步的输出以及语义向量C作为输入，并将它们与上一时间步的隐藏状态$s_{t'-1}$变换为当前时间步的隐藏状态$s_{t'}$，用函数g表示为:

$$\boldsymbol{s_{t'}} = g(y_{t'-1}, \boldsymbol{C}, \boldsymbol{s_{t'-1}})$$

- 有了解码器的隐藏状态后，我们可以使用自定义的输出层和 softmax 运算来计算$P(y_{t'}|y_1, y_2, \ldots, y_{t'-1}, C)$，例如，基于当前时间步的解码器隐藏状态$s_{t'}$，上一时间步的输出$y_{t'-1}$以及语义向量$C$来计算当前时间步输出$y_{t'}$的概率分布。

3.3 训练模型

- 根据最大似然估计，我们可以最大化输出序列基于输入序列的条件概率

$$\begin{aligned} P(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T) &= \prod_{t'=1}^{T'} P(y_{t'} \mid y_1, \ldots, y_{t'-1}, x_1, \ldots, x_T) \\ &= \prod_{t'=1}^{T'} P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c}), \end{aligned}$$

- 并得到该输出序列的损失

$$-\log P(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T) = -\sum_{t'=1}^{T'} \log P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c}),$$

- 在模型训练中，所有输出序列损失的均值通常作为需要最小化的损失函数。在上图所描述的模型预测中，需要将解码器在上一个时间步的输出作为当前时间步的输入。与此不同，在训练中也可以将标签序列（**训练集的真实输出序列**）在上一个时间步的标签作为解码器在当前时间步的输入。这叫作**强制教学**（teacher forcing）。