

第 8 章 深度学习—RNN

教材： 王万良《人工智能导论》（第4版）

<https://www.icourse163.org/course/ZJUT-1002694018>

社区资源： <https://github.com/Microsoft/ai-edu>



导论

- 语言模型
- 什么是RNN?
- 双向RNN
- RNN的应用领域

循环神经网络(Recurrent Neural Network)

□ 全连接神经网络和卷积神经网络

- 只能单独的取处理一个个的输入，前一个输入和后一个输入是完全没有关系的。

- 但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的。

- 理解一句话意思时，孤立的理解这句话的每个词是不够的，需要处理这些词连接起来的整个序列；

- 当处理视频的时候，也不能只单独的去分析每一帧，而要分析这些帧连接起来的整个序列。

循环神经网络(Recurrent Neural Network)

□ 全连接神经网络和卷积神经网络

- 只能单独的取处理一个个的输入，前一个输入和后一个输入是完全没有关系的。

- 但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的。

- 理解一句话意思时，孤立的理解这句话的每个词是不够的，需要处理这些词连接起来的整个序列；

- 当处理视频的时候，也不能只单独的去分析每一帧，而要分析这些帧连接起来的整个序列。

语言模型

□ RNN最早应用于自然语言处理领域

- 和电脑玩一个游戏，我们写出一个句子前面的一些词，然后，让电脑帮我们写下接下来的一个词。比如下面这句：

□ 我昨天上学迟到了，老师批评了_____。

- 给电脑展示了这句话前面这些词，然后，让电脑写下接下来的一个词。在这个例子中，接下来的这个词最有可能是『我』，而不太可能是『小明』，甚至是『吃饭』

□ 语言模型

- 给定一句话前面的部分，预测接下来最有可能的一个词是什么。

语言模型

□ 语言模型

- 对一种语言的特征进行建模，它有很多很多用处。
 - 比如在语音转文本(STT)的应用中，声学模型输出的结果，往往是若干个可能的候选词，这时候就需要语言模型来从这些候选词中选择一个最可能的。
 - 同样也可以用在图像到文本的识别中(OCR)

语言模型

□ 语言模型——N-Gram

- 假设一个词出现的概率只与前面N个词相关。
- 以2-Gram为例。首先，对前面的一句话进行切词：

□ 我 昨天 上学 迟到了，老师 批评了_____。

- 如果用2-Gram进行建模，那么电脑在预测的时候，只会看到前面的『了』，然后，电脑会在语料库中，搜索『了』后面最可能的一个词。不管最后电脑选的是不是『我』，我们都知道这个模型是不靠谱的，因为『了』前面说了那么一大堆实际上是没有用到的。

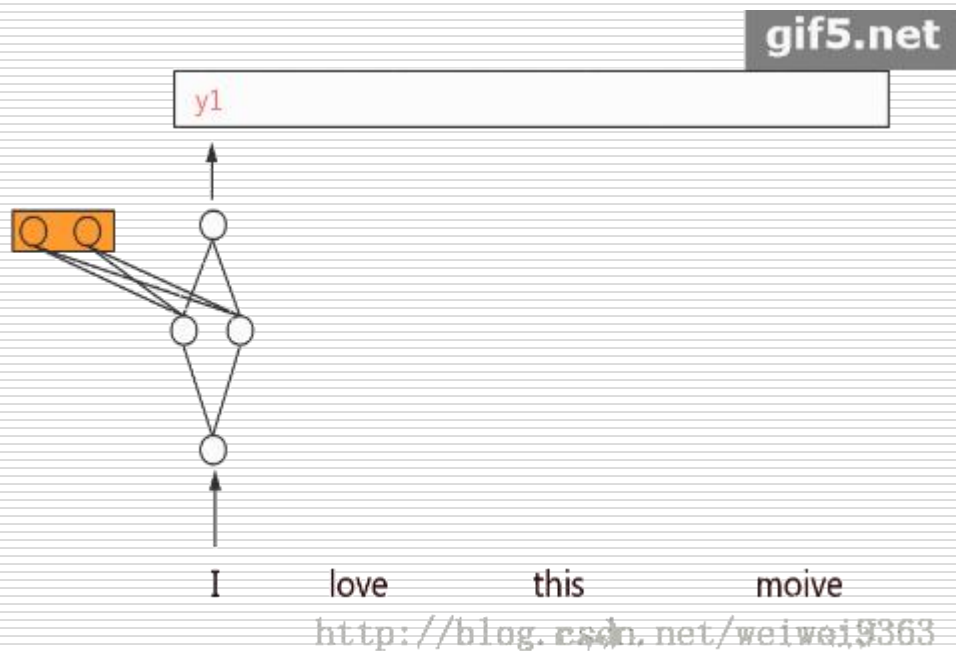
- 如果是3-Gram模型，会搜索『批评了』后面最可能的词，感觉上比2-Gram靠谱了不少，但还是远远不够的。因为这句话最关键的信息『我』，远在9个词之前！

□ 模型的大小和N的关系是指数级的，4-Gram模型就会占用海量的存储空间

语言模型

□ 语言模型

■ RNN理论上可以往前看(往后看)任意多个词



RNN

□ 为什么循环？

- 对一组序列输入重复进行同样的操作

□ 为什么需要 RNN？

■ 序列

- 相互依赖的（有限或无限）数据流，比如时间序列数据、信息性的字符串、对话等
- 当输入数据具有依赖性且是序列模式时，CNN 的结果一般都不太好
- RNN 对之前发生在数据序列中的事是有一定记忆
 - 有助于系统获取上下文。

何时使用 RNN?

□ 语言建模和文本生成

- 给出一个词语序列，预测下一个词语的可能性

□ 机器翻译

- 日常使用的实用系统都用了某种高级版本的 RNN

□ 语音识别

- 基于输入的声波预测语音片段，从而确定词语

□ 生成图像描述

- CNN 做图像分割，RNN 用分割后的数据重建描述

□ 视频标记

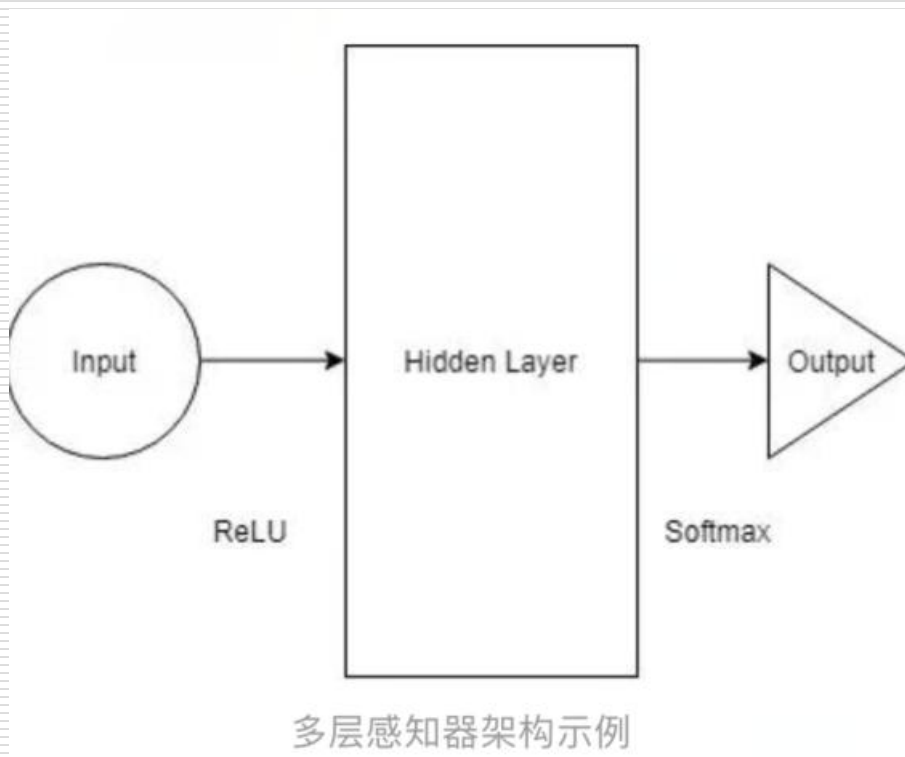
- 可以通过一帧一帧地标记视频进行视频搜索

循环网络

- 不仅将当前的输入样例作为网络输入，还将它们之前感知到的一并作为输入

多层感知器

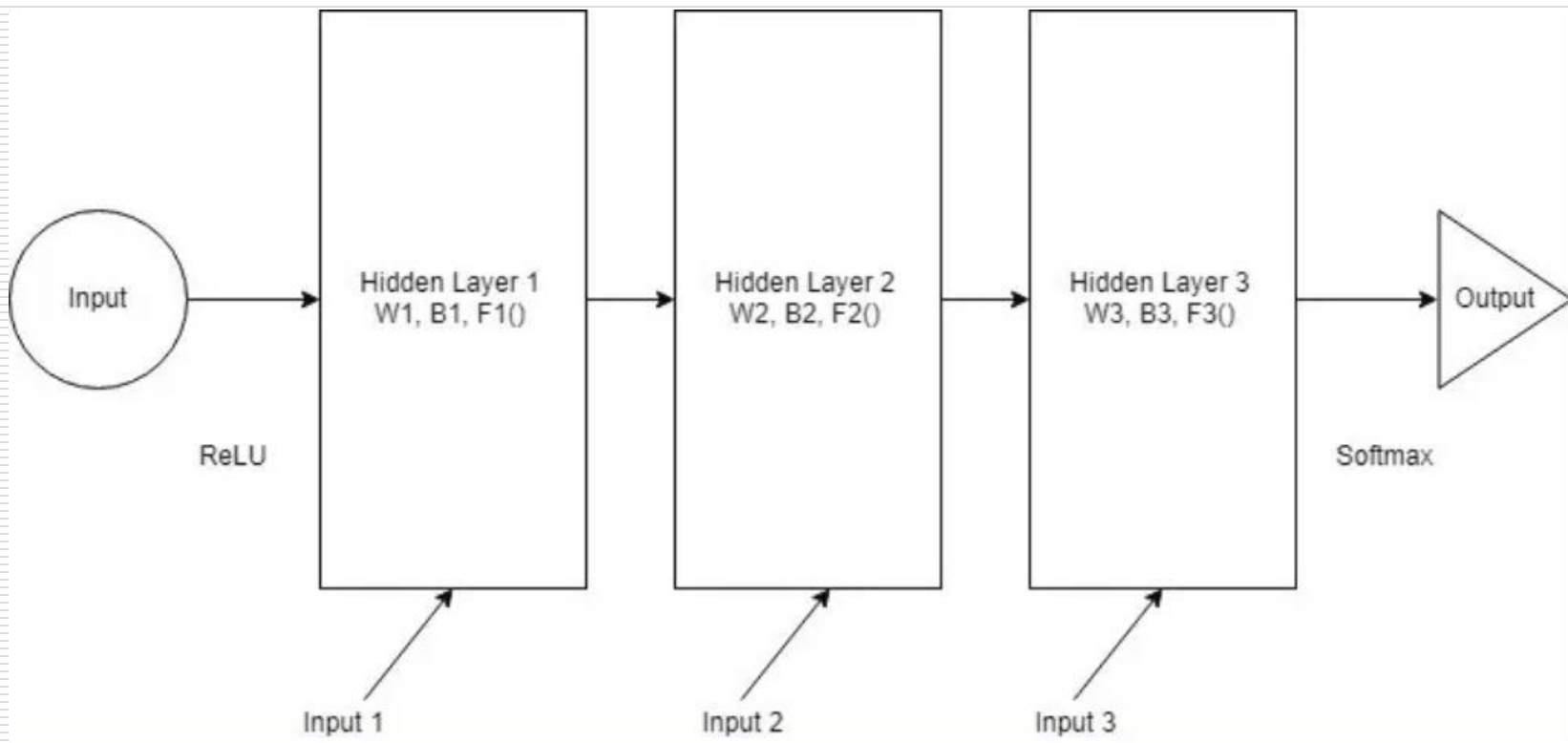
- 它有一个输入层、一个具备特定激活函数的隐藏层，最终可以得到输出



多层感知器

□ 扩展

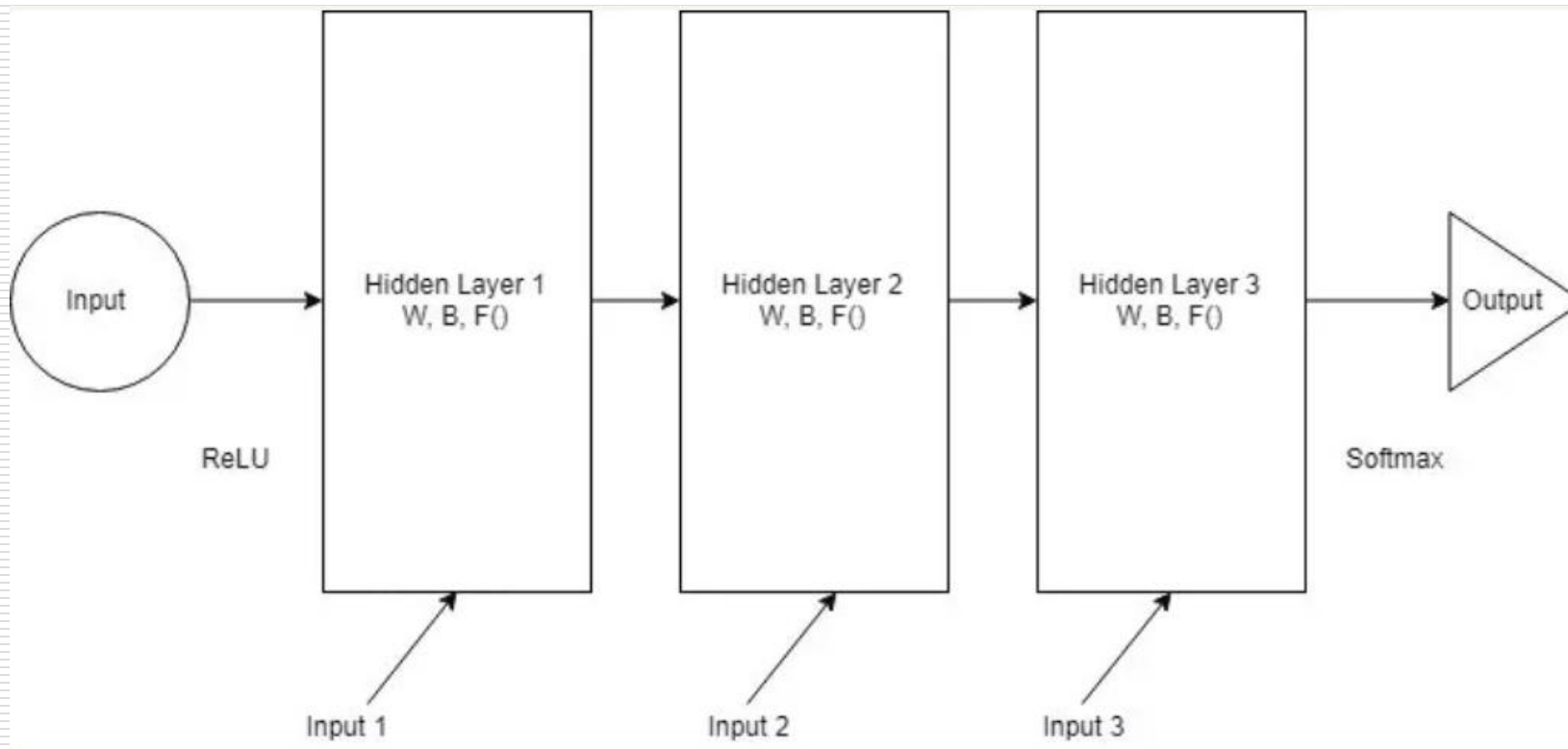
- 层数增加、每一个隐藏层都有自己的权重和偏置项



多层感知器

□ 扩展

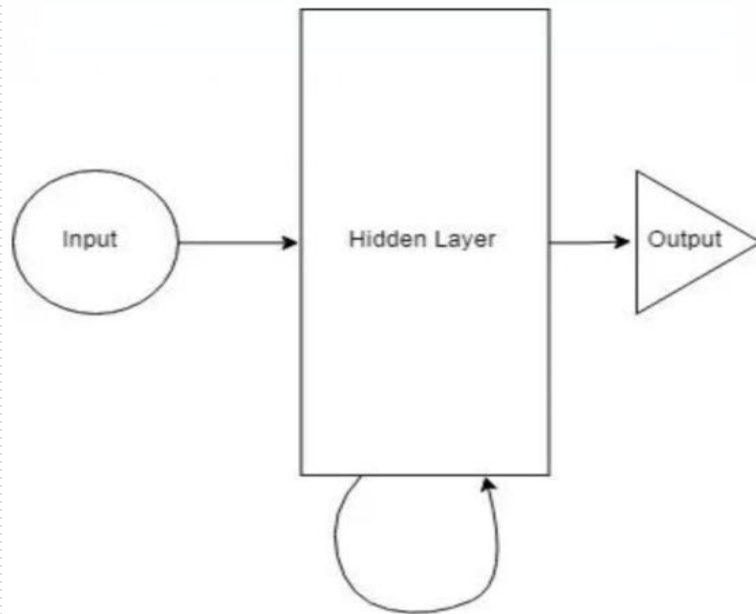
- 将所有层的权重和偏置项替换成相同的值



多层感知器

□ 扩展

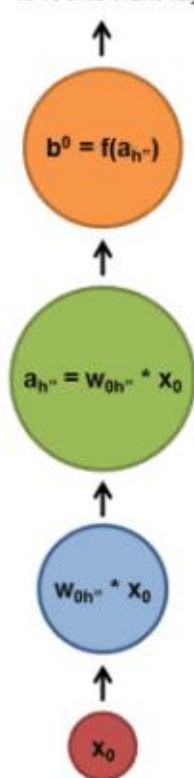
- 将所有层合并在一起了。所有的隐藏层都可以结合在一个循环层
- 每一步都会向隐藏层提供输入
- $t-1$ 步的决策影响到第 t 步做的决策



例子

- 一个有 8 个字母的单词 namaskar
- 向网络输入 7 个字母后试着找出第 8 个字母
- 隐藏层会经历 8 次迭代。如果展开网络的话就是一个 8 层的网络，每一层对应一个字母

b^0 is fed to next layer



什么是循环神经网络？

□ 假设用户输入：what time is it?

■ 首先，将句子分解为单个单词。RNN按顺序工作

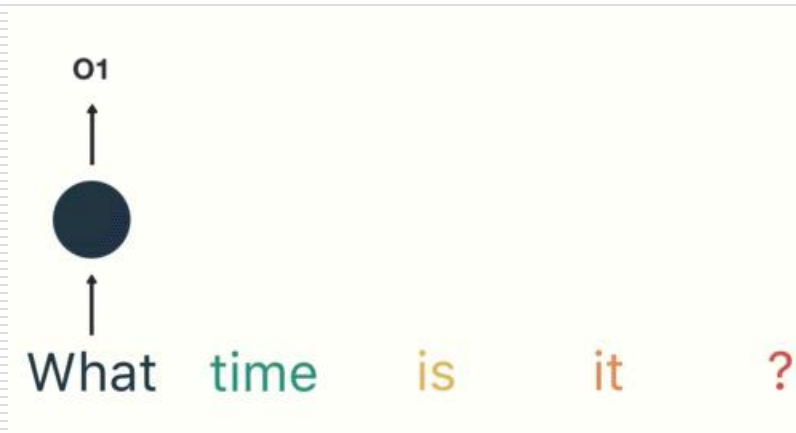
What time is it?

□ 第一步是将“**What**”输入RNN，RNN编码“**what**”并产生输出

What time is it ?

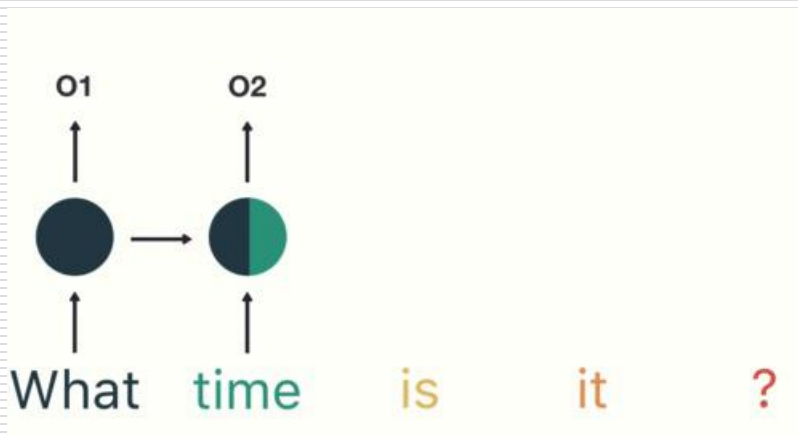
什么是循环神经网络？

- 假设用户输入：what time is it?
 - 提供单词“time”和上一步中的隐藏状态。RNN现在有关于“what”和“time”这两个词的信息



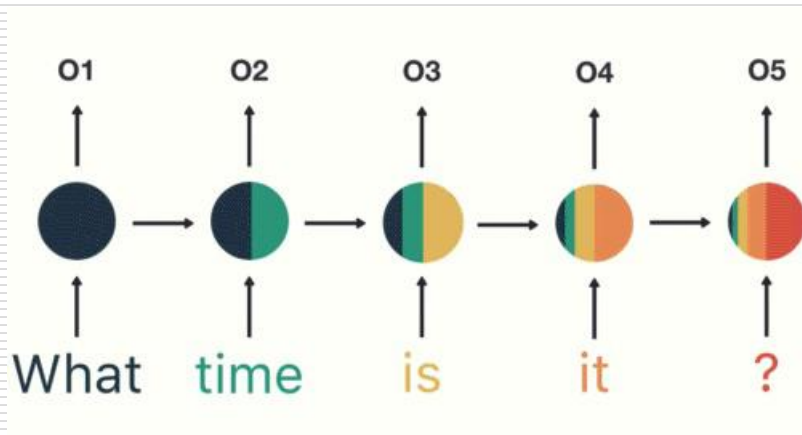
什么是循环神经网络？

- 假设用户输入：what time is it?
- 重复这个过程，直到最后一步。你可以通过最后一步看到RNN编码了前面步骤中所有单词的信息。




什么是循环神经网络？

- 假设用户输入：what time is it?
- 由于最终输出是从序列的部分创建的，因此能够获取最终输出并将其传递给前馈层以对意图进行分类



什么是循环神经网络？

□ 假设用户输入：what time is it?



```
rnn = RNN()
ff = FeedForwardNN()
hidden_state = [0.0, 0.0, 0.0, 0.0]

for word in input:
    output, hidden_state = rnn(word, hidden_state)

prediction = ff(output)
```

云栖社区 yq.aliyun.com

什么是循环神经网络？

□ 基本循环神经网络

■ 由输入层、一个隐藏层和一个输出层组成

□ x 是一个向量，它表示输入层的值

□ s 是一个向量，它表示隐藏层的值

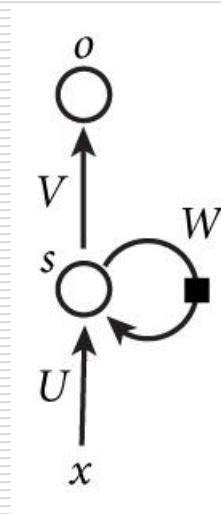
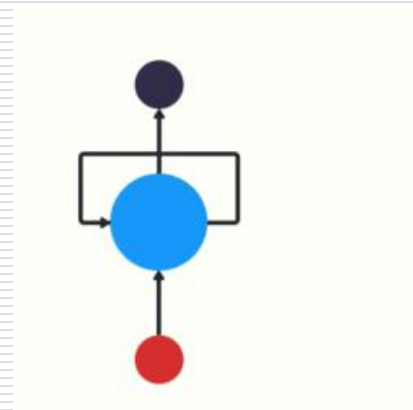
□ U 是输入层到隐藏层的权重矩阵

□ o 也是一个向量，它表示输出层的值

□ V 是隐藏层到输出层的权重矩阵

□ 权重矩阵 W 就是隐藏层上一次的值作为这一次的输入的权重

■ 循环神经网络的隐藏层的值 s 不仅仅取决于当前这次的输入 x ，还取决于上一次隐藏层的值 s 。

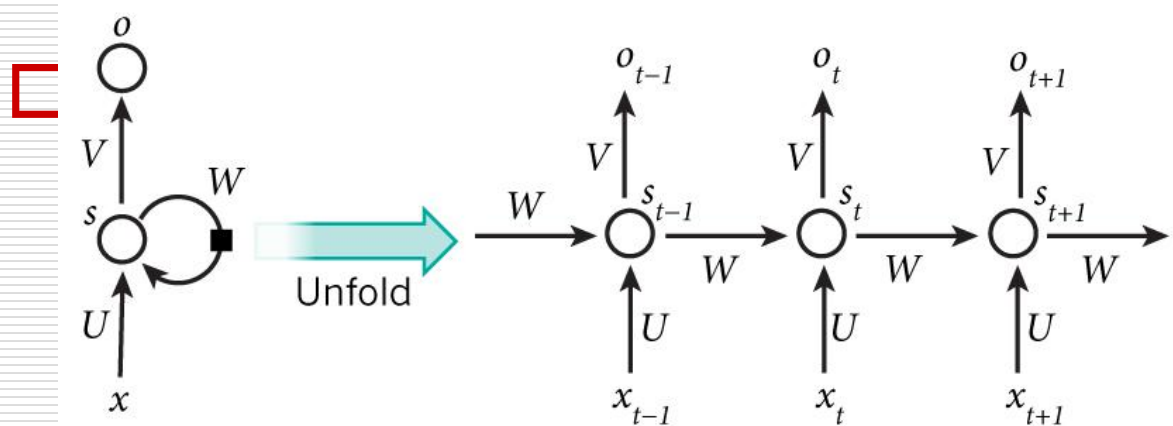


什么是循环神经网络？

□ 基本循环神经网络



什么是循环神经网络？



$$O_t = g(Vs_t)$$
$$s_t = f(Ux_t + Ws_{t-1})$$

- 输出层：全连接层
- 隐藏层：循环层

$$\begin{aligned} o_t &= g(Vs_t) \\ &= Vf(Ux_t + Ws_{t-1}) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2})) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3}))) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \dots)))) \end{aligned}$$

- 循环神经网络的输出值 O_t ，受前面历次输入值 x_t 、 x_{t-1} 、 x_{t-2} 、.....影响
- 循环神经网络可以往前看任意多个输入值的原因
- 注意：
 - 这里的 W, U, V 在每个时刻都是相等的(权重共享).
 - 隐藏状态可以理解为: $S=f(\text{现有的输入}+\text{过去记忆总结})$

RNN的反向传播

□ RNN的权重参数 W, U, V 都是怎么更新的呢？

■ 每一次的输出值 O_t 都会产生一个误差值 e_t , 则总的误差可以表示为: $E = \sum_t e_t$

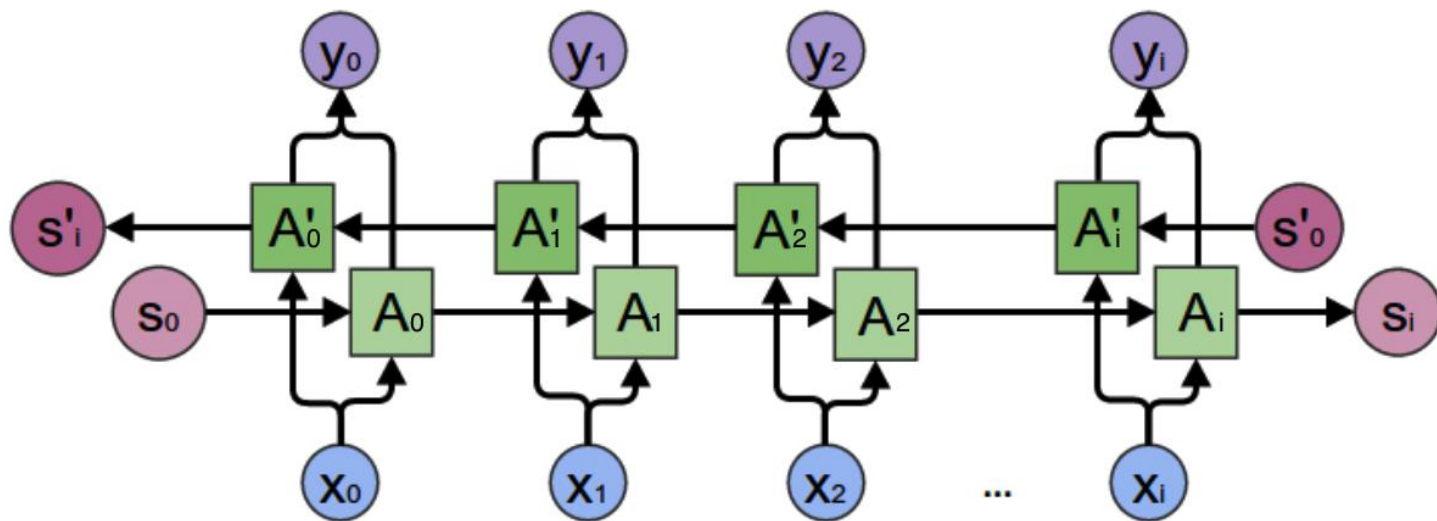
■ 则损失函数可以使用交叉熵损失函数也可以使用平方误差损失函数.

■ 由于每一步的输出不仅仅依赖当前步的网络, 并且还需要前若干步网络的状态, 那么这种BP改版的算法叫做Backpropagation Through Time(BPTT), 也就是将输出端的误差值反向传递, 运用梯度下降法进行更新

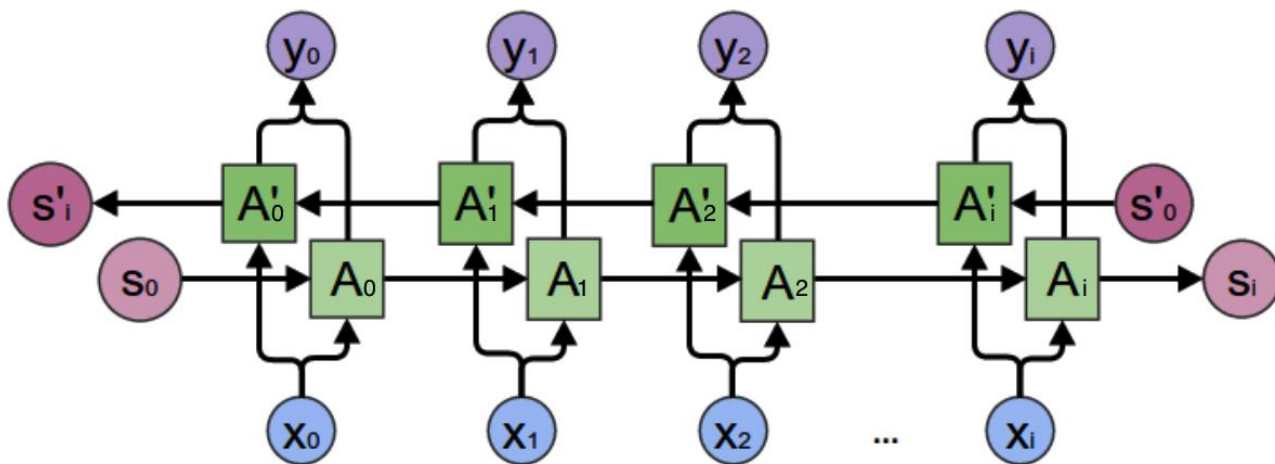
双向循环神经网络

□ 对于语言模型来说，很多时候光看前面的词是不够的，比如下面这句话：

■ 我的手机坏了，我打算_____一部新手机。



双向循环神经网络



- 隐藏层要保存两个值： A 和 A' ，一个 A 参与正向计算，另一个 A' 参与反向计算。最终的输出值 y_2 取决于 A_2 和 A'_2 ： $y_2 = g(VA_2 + V'A'_2)$

$$A_2 = f(WA_1 + Ux_2)$$

$$A'_2 = f(W'A'_3 + U'x_2)$$

$$o_t = g(Vs_t + V's'_t)$$

$$s_t = f(Ux_t + Ws_{t-1})$$

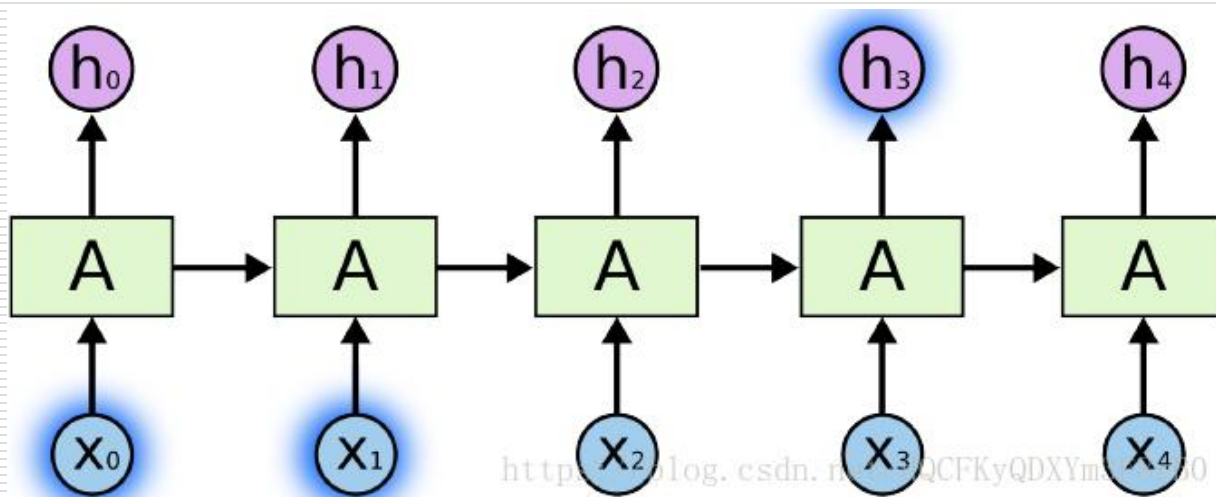
$$s'_t = f(U'x_t + W's'_{t+1})$$

双向循环神经网络

- 双向RNN需要的内存是单向RNN的两倍，因为在同一时间点，双向RNN需要保存两个方向上的权重参数，在分类的时候，需要同时输入两个隐藏层输出的信息。

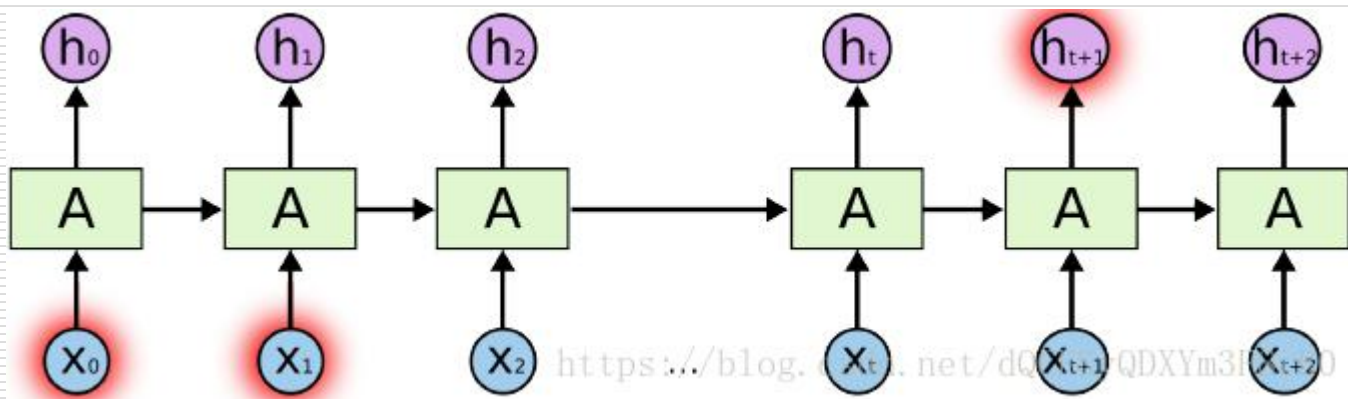
LSTM

- 有的时候，我们只需要最近的一些信息就可以很好的预测当前的任务。
 - 比如在语言模型里，我们需要预测”白云飘浮在（天空）”的下一个单词，我们很容易根据之前的这几个词就可以预测最可能的词是”天空”。我们要预测的 h_3 需要的信息 x_0, x_1 距离不是太远。



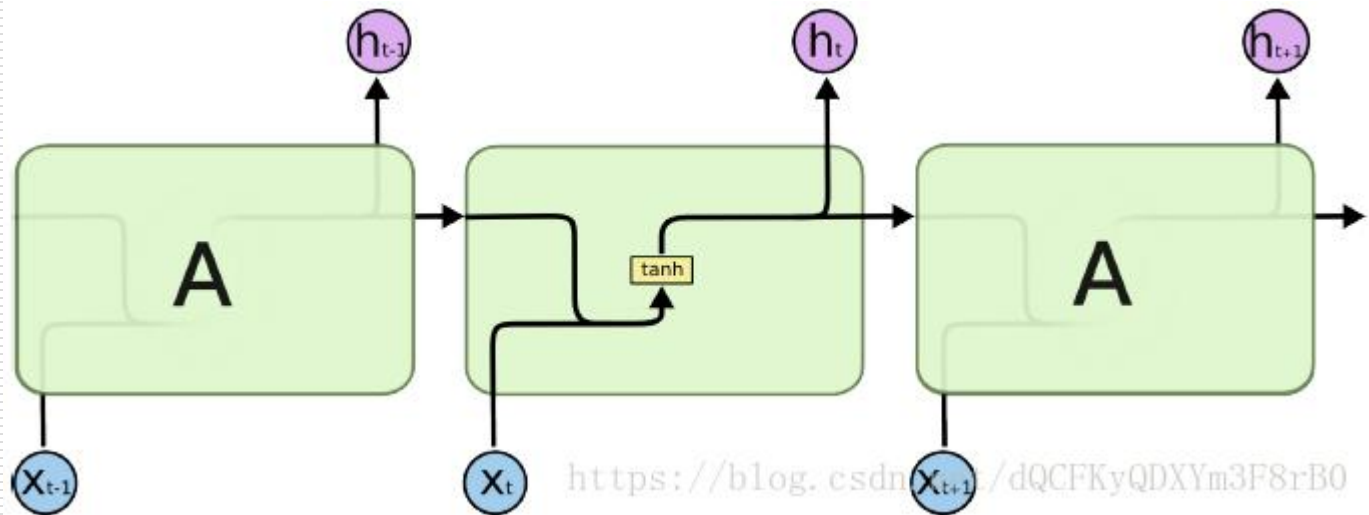
LSTM

- 但是有的时候我们需要更多的上下文信息来预测。
 - 比如“我从小生长在四川.....我会讲流利的 (四川话)”。最近的信息“我会讲流利的” 暗示后面很可能是一种语言，但是我们无法确定是哪种语言，除非我们有更久之前的上下文“我从小生长在四川”。
 - 因此为了准确的预测，我们可能需要依赖很长距离的上下文。如图所示，为了预测 x_{t+1} ，我们需要很远的 x_0, x_1 。理论上，如果我们的参数学得足够好，RNN 是可以学习到这种长距离依赖关系的。但是很不幸的是，在实际应用中RNN 很难学到。

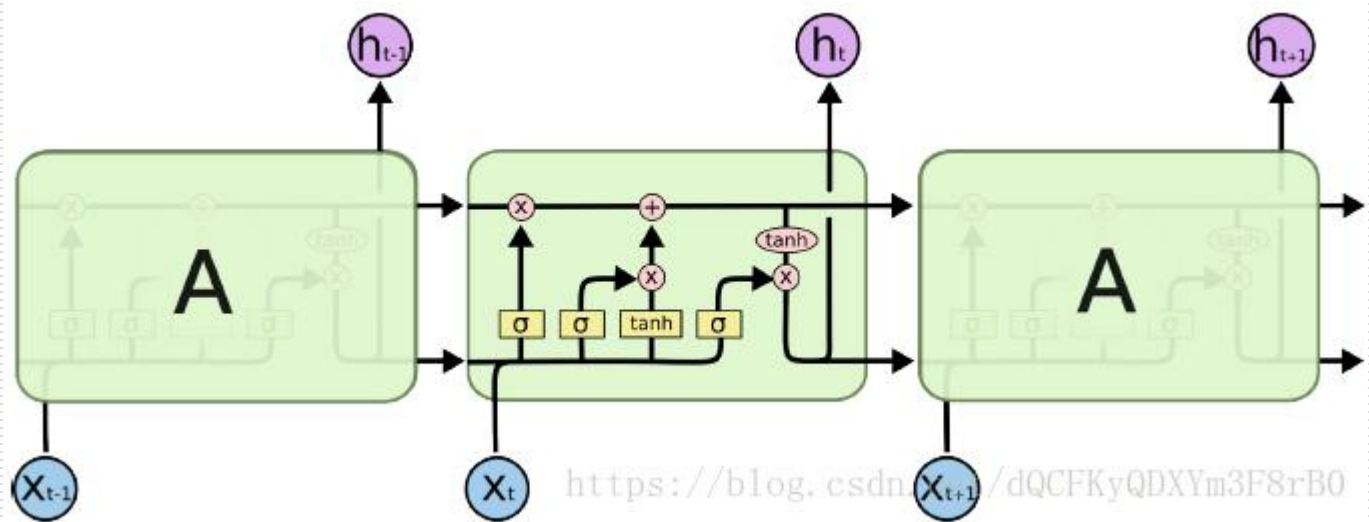


LSTM

□ RNN

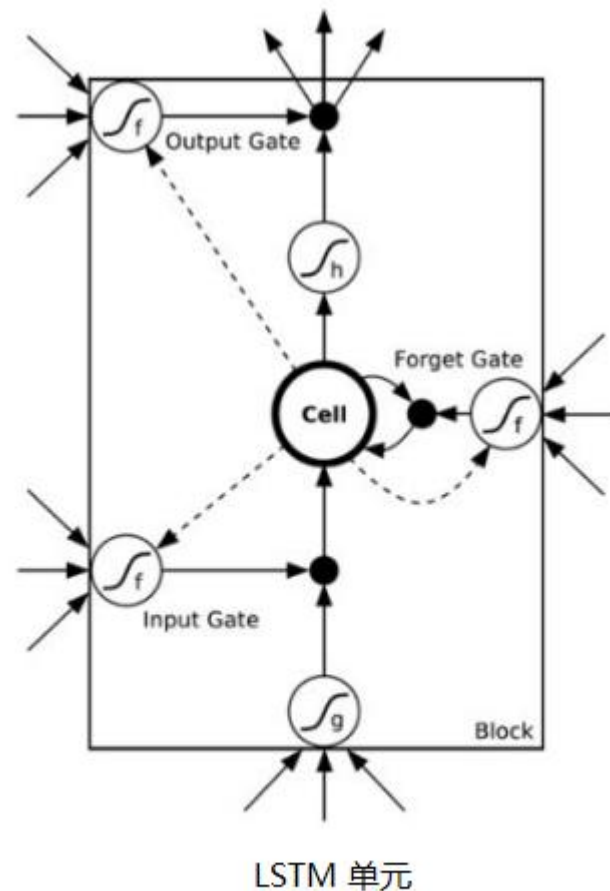


□ LSTM



LSTM

- 中间有一个cell（细胞）
 - LSTM用于判断信息是否有用的“处理器”。
- cell旁边被放置了三扇门
 - 分别是输入门（Input Gate）
 - 遗忘门（Forget Gate）
 - 输出门（Output Gate）
- 一个信息进入LSTM的网络当中，可以根据规则来判断是否有用，只有符合要求的信息才会被留下，不符合的信息则会通过遗忘门被遗忘

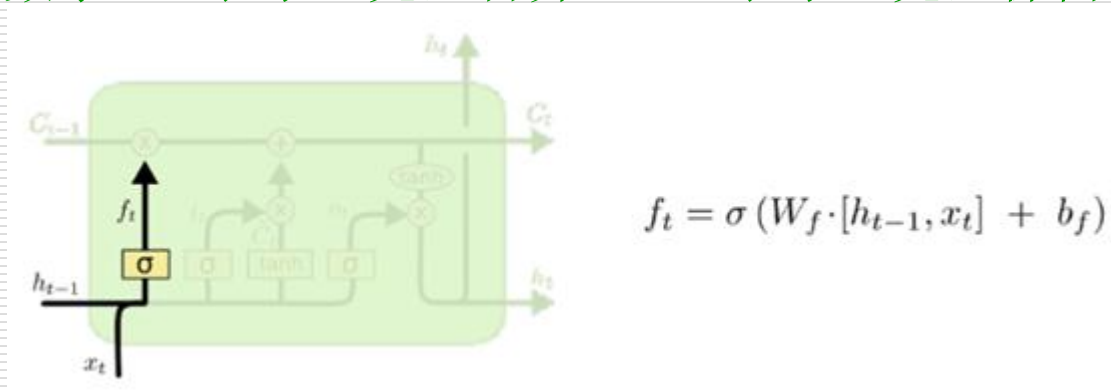


LSTM

□ 小明刚吃完米饭，现在准备要吃水果，然后拿起了一个（）

■ (1) 遗忘门 (Forget Gate)

□ 该门的示意图如下，该门会读取 h_{t-1} 和 x_t 的信息，通过sigmoid层输出一个介于0到1之间的数值，作为给每个在细胞状态 C_{t-1} 中的数字，0表示“完全舍弃”，1表示“完全保留”。



□ “小明刚吃完米饭”，这句话主语是“小明”，宾语是“米饭”，下一句话“现在准备要吃水果”，这时宾语已经变成了新的词“水果”，那第三句话要预测的词，就是跟“水果”有关了，跟“米饭”已经没有什么关系，因此，这时便可以利用“遗忘门”将“米饭”遗忘掉

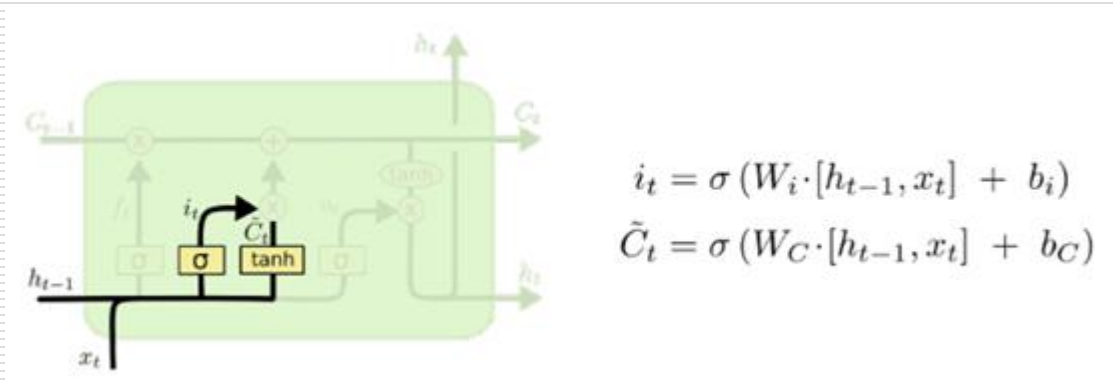
LSTM

□ 小明刚吃完米饭，现在准备要吃水果，然后拿起了一个（）

■ (2) 输入门 (Input Gate)

□ 下一步是确定什么样的新信息被存放在细胞状态中。这里包含两部分：

■ 首先是经过“输入门”，这一层是决定我们将要更新什么值；然后，一个 \tanh 层创建一个新的候选值向量，加入到状态中



■ 希望将新的代词“水果”增加到细胞状态中，来替代旧的需要忘记的代词“米饭”。

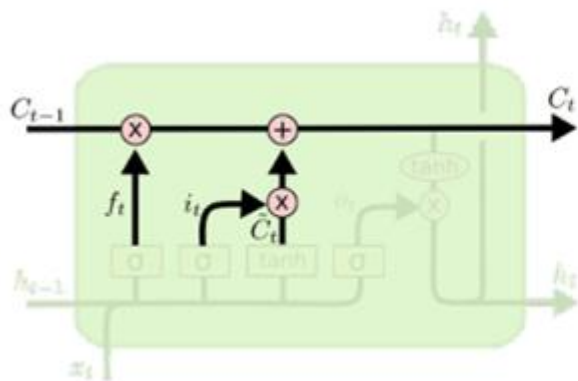
LSTM

□ 小明刚吃完米饭，现在准备要吃水果，然后拿起了一个（）

■ （2）输入门（Input Gate）

□ 下一步是确定什么样的新信息被存放在细胞状态中。这里包含两部分：

■ 更新旧细胞的状态，由 C_{t-1} 更新为 C_t ，更新方式为：（1）把旧状态 C_{t-1} 与 f_t 相乘（遗忘门，输出遗忘程度，即0到1之间的值），丢掉需要丢弃的信息（如遗忘门输出0，则相乘后变成0，该信息就被丢弃了）；（2）然后再加上 i_t 与候选值相乘。这两者合并后就变成一个新的候选值



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

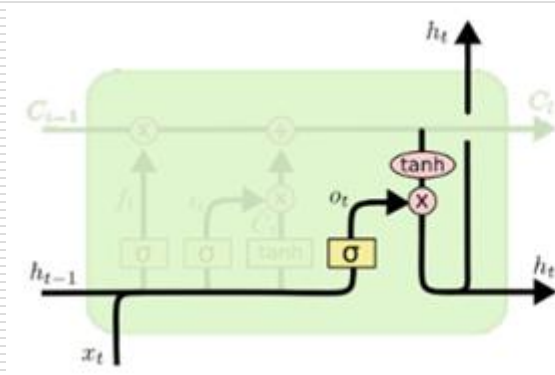
■ 根据前面确定的目标，丢弃旧的代词信息（米饭）并添加新的信息（水果）的地方

LSTM

□ 小明刚吃完米饭，现在准备要吃水果，然后拿起了一个（）

■ (3) 输出门 (Output Gate)

□ 最后我们要确定输出什么值，首先，通过一个sigmoid层来确定细胞状态的哪个部分将要输出出去，接着，把细胞状态通过 \tanh 进行处理（得到一个介于-1到1之间的值）并将它和 sigmoid 的输出结果相乘，最终将会仅仅输出我们需要的那部分信息。



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

□ 因为看到了一个新的代词（水果），可能需要输出与之相关的信息（苹果、梨、香蕉……）

RNN的应用领域

- ❑ 自然语言处理(NLP): 主要有视频处理, 文本生成, 语言模型, 图像处理
- ❑ 机器翻译, 机器写小说
- ❑ 语音识别
- ❑ 图像描述生成
- ❑ 文本相似度计算
- ❑ 音乐推荐、网易考拉商品推荐、Youtube视频推荐等新的应用领域.

RNN应用——写歌词、写诗

□ <https://github.com/hzy46/Char-RNN-TensorFlow>

使用LSTM模型预测股价——基于Keras

□ 加载数据集

- `dataset_train = pd.read_csv('NSE-TATAGLOBAL.csv')`
- `training_set = dataset_train.iloc[:,1:2].values`
- **Open**列是股票交易的开盘价，**Close**列是收盘价，**High**列是最高价，**Low**列是最低价

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

使用LSTM模型预测股价——基于Keras

□ 特征归一化

- `from sklearn.preprocessing import MinMaxScaler`
- `sc = MinMaxScaler(feature_range = (0,1))`
- `training_set_scaled = sc.fit_transform(training_set)`

使用LSTM模型预测股价——基于Keras

□ 按步长创建数据

- LSTM要求数据有特殊格式，通常是3D数组格式。初始按照60的步长创建数据，并通过Numpy转化到数组中。然后，把 X_train的数据转化到3D维度的数组中，时间步长设置为60，每一步表示一个特征

- `X_train = []`
- `y_train = []`
- `for i in range(60,2035):`
- `X_train.append(training_set_scaled[i-60:i,0])`
- `y_train.append(training_set_scaled[i,0])`
- `X_train, y_train = np.array(X_train), np.array(y_train)`
- `X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1],1))`

使用LSTM模型预测股价——基于Keras

□ 构建LSTM

- 顺序初始化神经网络
- 添加一个紧密连接的神经网络层
- 添加长短时记忆层(LSTM)
- 添加dropout层防止过拟合
 - `from keras.models import Sequential`
 - `from keras.layers import Dense`
 - `from keras.layers import LSTM`
 - `from keras.layers import Dropout`

使用LSTM模型预测股价——基于Keras

□ LSTM层

- 1、50 units 表示输出空间是50维度的单位
- 2、return_sequences=True 表示是返回输出序列中的最后一个输出，还是返回完整序列
- 3、input_shape 训练集的大小
 - regressor = Sequential()
 - regressor.add(LSTM(units =50,return_sequences =True,input_shape = (X_train.shape[1],1)))
 - regressor.add(Dropout(0.2))
 - regressor.add(LSTM(units =50,return_sequences =True))
 - regressor.add(Dropout(0.2))
 - regressor.add(LSTM(units =50,return_sequences =True))
 - regressor.add(Dropout(0.2))
 - regressor.add(LSTM(units =50))
 - regressor.add(Dropout(0.2))
 - regressor.add(Dense(units =1))
 - regressor.compile(optimizer ='adam',loss ='mean_squared_error')
 - regressor.fit(X_train, y_train, epochs =100,batch_size =32)

使用LSTM模型预测股价——基于Keras

□ 测试集上预测股价

■ 导入股价预测的测试集:

□ `dataset_test = pd.read_csv('tatatest.csv')`

□ `real_stock_price = dataset_test.iloc[:,1:2].values`

■ 为了预测未来的股票价格，测试集加载后

□ 在0轴上合并训练集和测试集

□ 将时间步长设置为60

□ 使用MinMaxScaler函数转换新数据集

□ 按照前面所做的那样重新规整数据集

■ 预测之后，我们用inverse_transform函数处理，以返回正常可读格式的股票价格

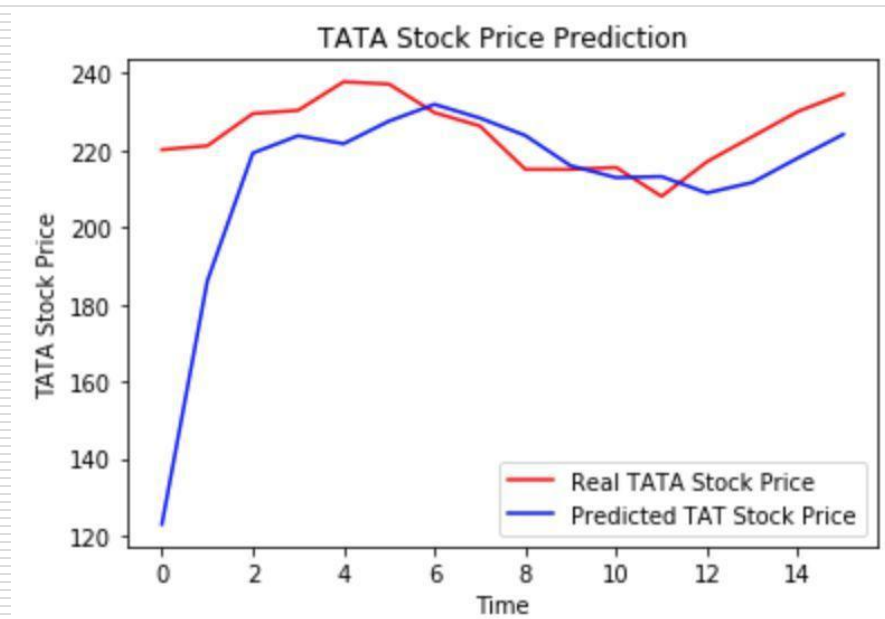
使用LSTM模型预测股价——基于Keras

- ❑ `dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis =0)`
- ❑ `inputs = dataset_total[len(dataset_total) - len(dataset_test) -60:].values`
- ❑ `inputs = inputs.reshape(-1,1)`
- ❑ `inputs = sc.transform(inputs)`
- ❑ `X_test = []`
- ❑ `for i in range(60,76):`
- ❑ `X_test.append(inputs[i-60:i,0])`
- ❑ `X_test = np.array(X_test)`
- ❑ `X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1))`
- ❑ `predicted_stock_price = regressor.predict(X_test)`
- ❑ `predicted_stock_price = sc.inverse_transform(predicted_stock_price)`

使用LSTM模型预测股价——基于Keras

□ 展示结果

- `plt.plot(real_stock_price, color='black',label='TATA Stock Price')`
- `plt.plot(predicted_stock_price, color='green',label='Predicted TATA Stock Price')`
- `plt.title('TATA Stock Price Prediction')`
- `plt.xlabel('Time')`
- `plt.ylabel('TATA Stock Price')`
- `plt.legend()`
- `plt.show()`



RNN与CNN的结合应用：看图说话

- 在图像处理中，目前做的最好的是CNN
- 自然语言处理中，表现比较好的是RNN
- 因此，我们能否把他们结合起来，一起用呢？
 - 看图说话
 - 可疑活动视频分析

可疑活动视频分析

□ 将视频分为三类：

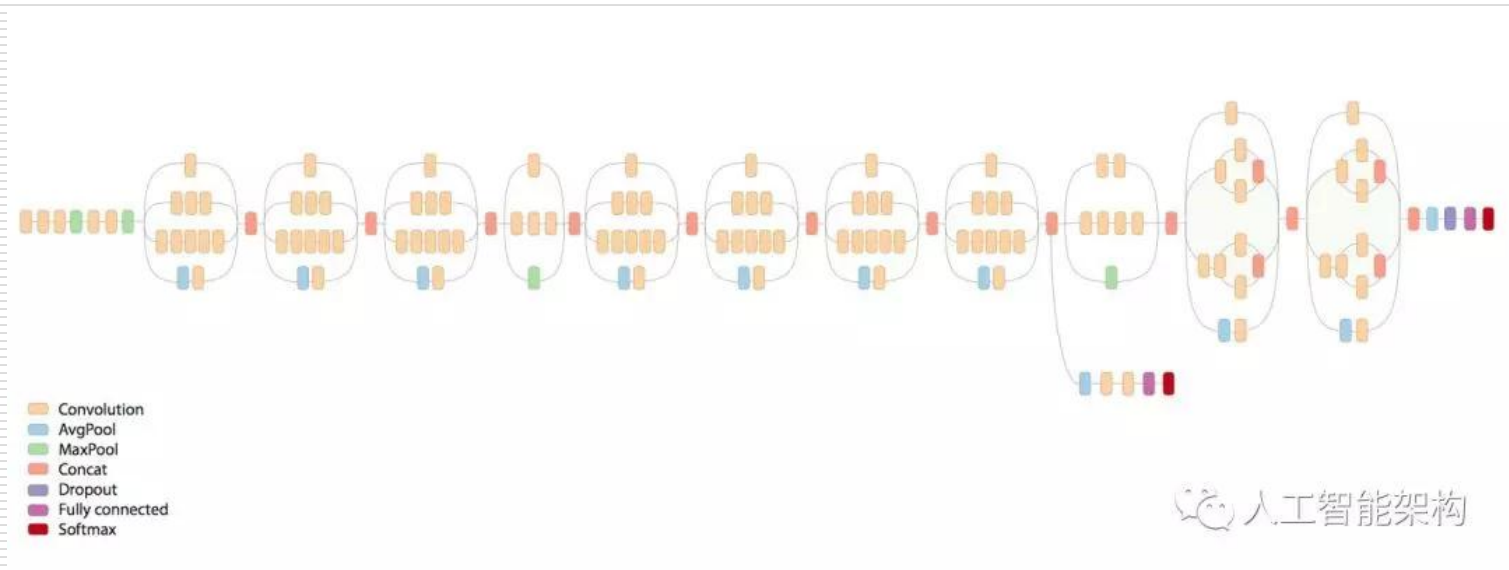
- 1、犯罪或暴力活动；
- 2、可能是可疑的；
- 3、安全；

□ 解决方案

- 基于卷积神经网络和递归神经网络架构

可疑活动视频分析——解决方案架构描述

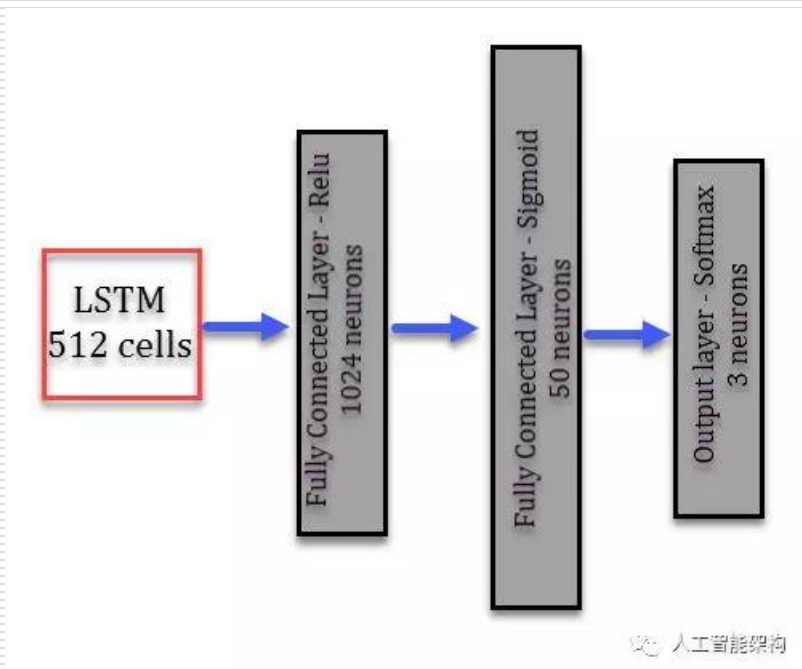
- 第一个神经网络——卷积神经网络，目的是提取图像的高级特征，并降低输入的复杂性。
- 由Google开发的Inception层预训练模型
 - Inception v3在ImageNet大型视觉识别挑战数据集上进行了训练，模型试图将整个图像分为1000个类
 - 将已经训练好的模型的一部分知识（网络结构）直接应用到另一个新的类似模型中



可疑活动视频分析——解决方案架构描述

□ 第二个神经网络——递归神经网络

- 目的是理解所描绘动作的顺序。
- 第一层有一个LSTM单元，随后是两个隐藏层（一个有11024个神经元和ReLU激活；另一个有50个神经元，一个sigmoid激活），输出层是一个带有softmax激活的三神经元层，它给出了最后的分类



可疑活动视频分析——解决方案架构描述

□ 提取视频的帧

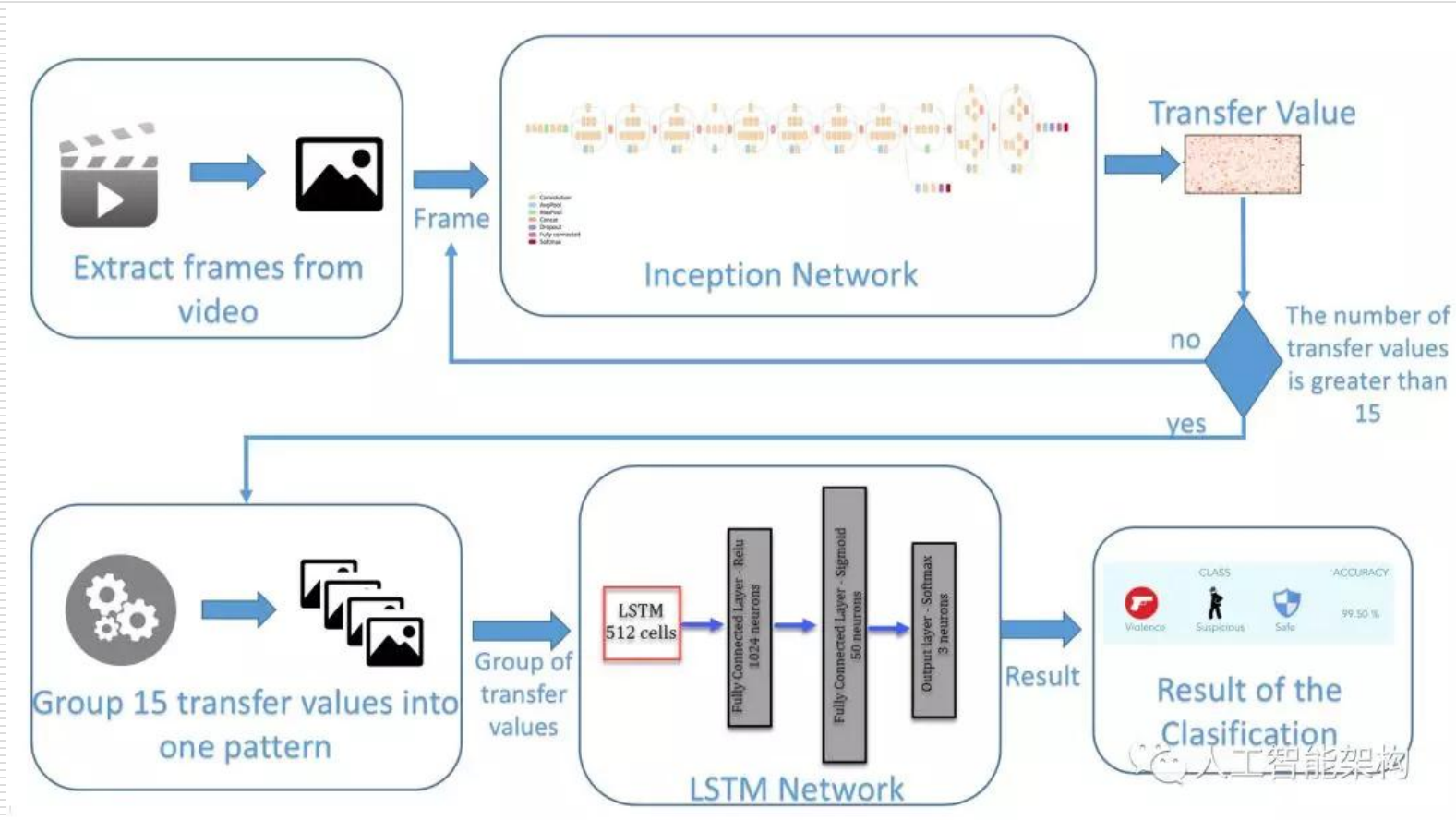
- 每隔**0.2**秒提取一帧，使用**Inception** 模型对提取的该帧进行预测
- 提取最后一个池化层的结果，该层是**2048**个值的向量（高级特征映射）

□ 存储了由**Inception** 模型预测生成的**15**个特征映射，相当于**3**秒的视频

□ 将这组特征映射连接成一个单独的模式，这将是第二个神经网络的输入，即递归，以便获得系统的最终分类

可疑活动视频分析——解决方案架构描述

- 在屏幕上看到的是视频的实时分类，每3秒钟会看到视频的分类：安全、可疑或犯罪活动



可疑活动视频分析——训练数据集

- 用于训练网络的数据集包括150分钟，分为38个视频
 - 采用每0.2秒长的帧，用于训练的数据集共有45000帧——相当于3000段视频，考虑到视频的一段代表3秒钟（或15帧）
 - 对整个数据集标记并分组：80%用于训练，20%用于测试



Criminal Activity



Suspicious Activity



Safe Activity

人工智能架构

可疑活动视频分析——执行

- 使用OpenCV for Python来分割帧中的视频，并将它们调整成为200*200像素
 - 每个预测结果表示从该特定帧提取的高级特征映射的“传输值”。
 - 将变量保存在transfer_value变量中，将其各自的标签保存在label_train变量中。
 - 将它们分成15帧的组，并将结果保存在joint_transfer变量中。

```
frames_num=15
count = 0
joint_transfer=[]
for i in range(int(len(transfer_values)/frames_num)):
    inc = count+frames_num
    joint_transfer.append([transfer_values[count:inc],labels_train[count]])
    count =inc
```


可疑活动视频分析——执行

□ 基于keras创建模型

```
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.layers import LSTM
chunk_size = 2048
n_chunks = 15
rnn_size = 512
model = Sequential()
model.add(LSTM(rnn_size, input_shape=(n_chunks, chunk_size)))
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dense(50))
model.add(Activation(sigmoid))
model.add(Dense(3))
model.add(Activation('softmax'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

可疑活动视频分析——执行

□ 训练模型

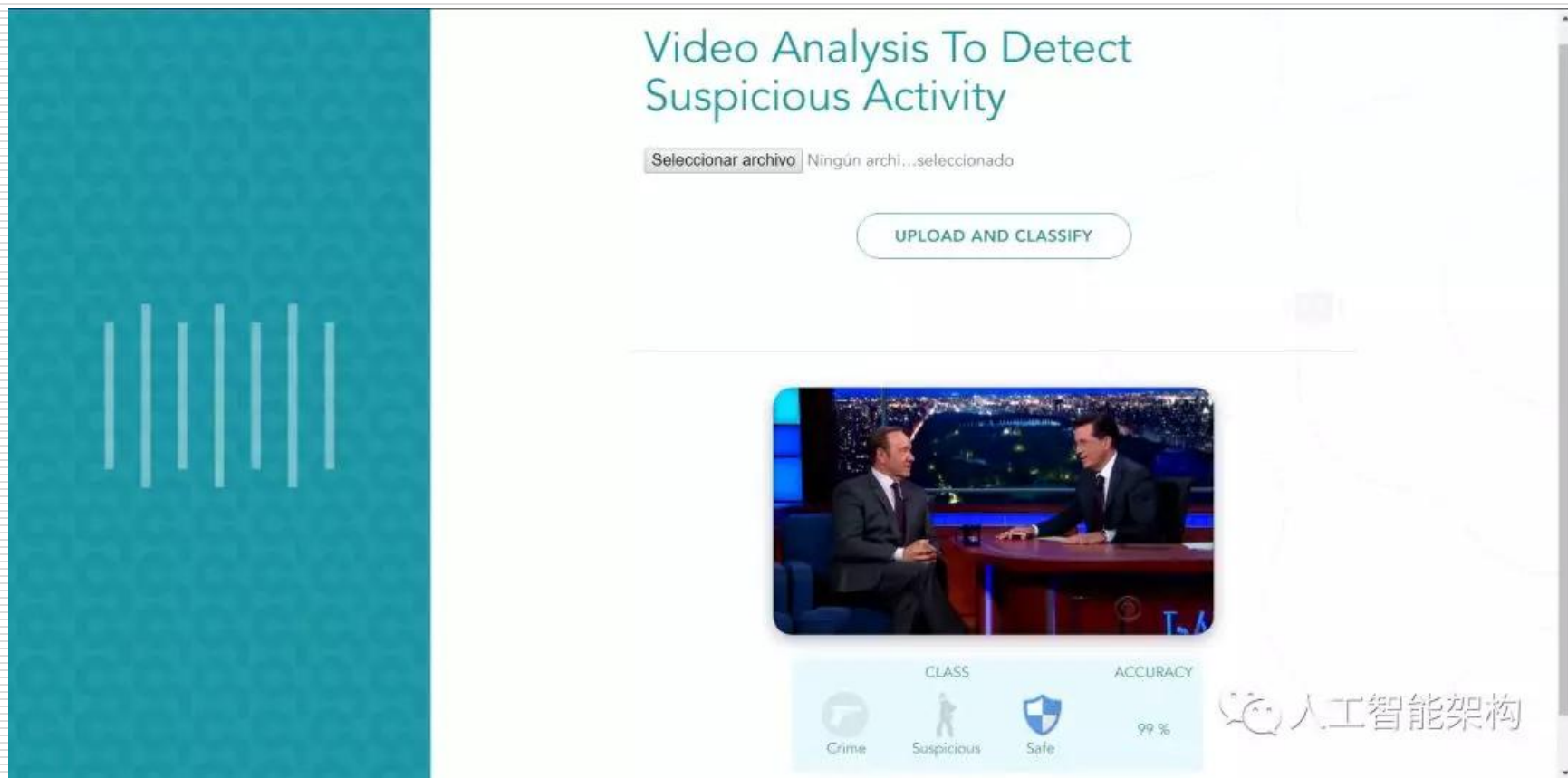
```
data = []  
target = []  
epoch = 1500  
batchS = 100  
for i in joint_transfer:  
    data.append(i[0])  
    target.append(np.array(i[1]))  
model.fit(data, target, epochs=epoch, batch_size=batchS, verbose=1)
```

□ 保存模型

```
model.save("rnn.h5", overwrite=True)
```


可疑活动视频分析——结果和其他应用

- 前端页面，可以在其中上传视频并开始实时分类。可以看到类是如何不断变化的，以及该类对应的准确性。这些值每3秒更新，直到视频结束

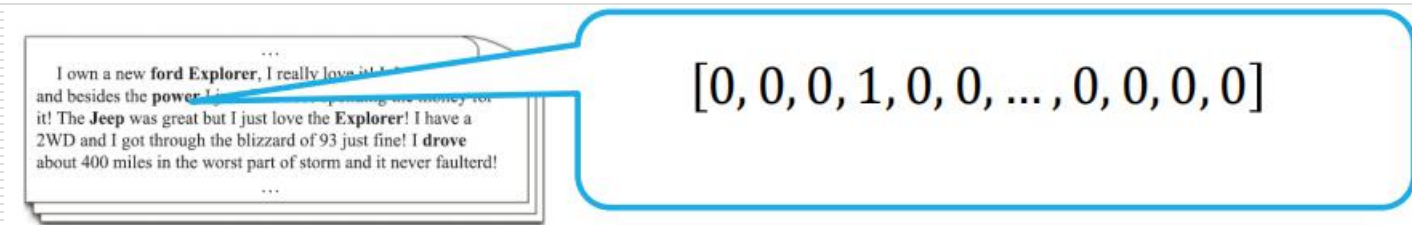


可疑活动视频分析——结果和其他应用

- 将它连接到安全摄像机，对视频进行实时分析，当系统检测到犯罪或可疑活动时，可以激活报警或是提醒警察。
- 可以使用经过适当数据训练的类似系统来检测不同类型的活动，例如：使用位于学校的摄像头，检测目标（可能是学生等，待检测对象）是否受到欺凌

深度学习的应用—学习单词的表达：词向量(Word2Vec)

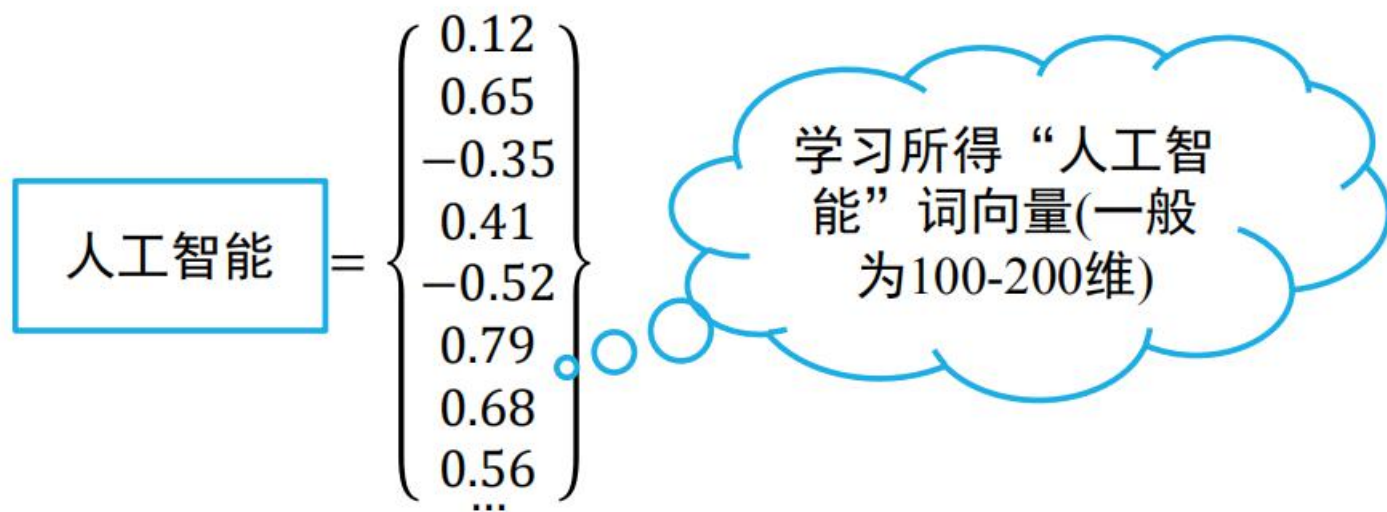
- ❑ 在基于规则和统计的自然语言传统方法中，将单词视为独立符号
- ❑ 在向量空间中，一个单词按照其在文档中出现的有无，被表示为如下向量（按照字典序）：



- 上述表示方法称为One-hot向量。
- ❑ 缺点：
 - 维数灾难的困扰
 - 无法刻画词与词之间的相似性：任意两个词之间都是孤立的

深度学习的应用—学习单词的表达：词向量(Word2Vec)

- ❑ One-hot 表达与单词的分布无关
- ❑ 通过深度学习方法，将单词表征为K维实数值向量(distribution representation)。这样，把对文本内容分析简化为 K 维向量空间中的向量运算，而向量空间上的相似度可以用来表示文本语义上的相似。用深度学习算法生成每个单词的向量表达所有单词的向量表达组成了一个“词向量空间”
- ❑ 单词表达为词向量后，很多 NLP 相关工作（如聚类、同义词计算、主题挖掘等)可以顺利开展



深度学习的应用—学习单词的表达：词向量(Word2Vec)

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

深度学习	[0, 0, 0, 0 ... 0, 1, 0, ... 0, 0, 0]
机器学习	[0, 1, 0, 0 ... 0, 0, 0, ... 0, 0, 0]
一种	[0, 0, 0, 0 ... 0, 0, 0, ... 1, 0, 0]
基于	[0, 0, 1, 0 ... 0, 0, 0, ... 0, 0, 0]
数据	[0, 0, 0, 0 ... 0, 0, 0, ... 0, 1, 0]
表征学习	[1, 0, 0, 0 ... 0, 0, 0, ... 0, 0, 0]
方法	[0, 0, 0, 0 ... 1, 0, 0, ... 0, 0, 0]
⋮	
手工	[0, 0, 0, 1 ... 0, 0, 0, ... 0, 0, 0]
获取	[0, 0, 0, 0 ... 0, 0, 0, ... 0, 0, 1]
特征	[0, 0, 0, 0 ... 0, 0, 1, ... 0, 0, 0]

One-Hot
单词之间的关联丢失

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

深度学习	[0.23, -0.18, 0.45... .. 0.68, -0.33]
机器学习	[0.11, -0.25, 0.78... ..0.22, 0.61]
一种	[0.08, 0.09, 0.52... ..-0.37, -0.42]
基于	[-0.16, 0.29, 0.80... ..-0.21, 0.20]
数据	[-0.67, -0.03, 0.29... ..0.46, 0.87]
表征学习	[0.26, 0.55, 0.70... ..0.62, -0.40]
方法	[0.59, 0.92, 0.33... ..0.53, 0.90]
⋮	
手工	[0.15, -0.66, -0.72... ..0.48, 0.59]
获取	[0.19, 0.23, 0.81,0.43, -0.81]
特征	[0.76, 0.49, -0.38... ..0.65, 0.07]

词向量
可基于单词形成的向量进行后续操作

深度学习的应用—学习单词的表达：词向量(Word2Vec)

- 通过某个单词 w_t 上下文单词的词向量来预测单词 w_t 的词向量

$$f(w_t, w_{t-1}, \dots, w_{t-n+2}, w_{t-n+1}) = p(w_t | context)$$

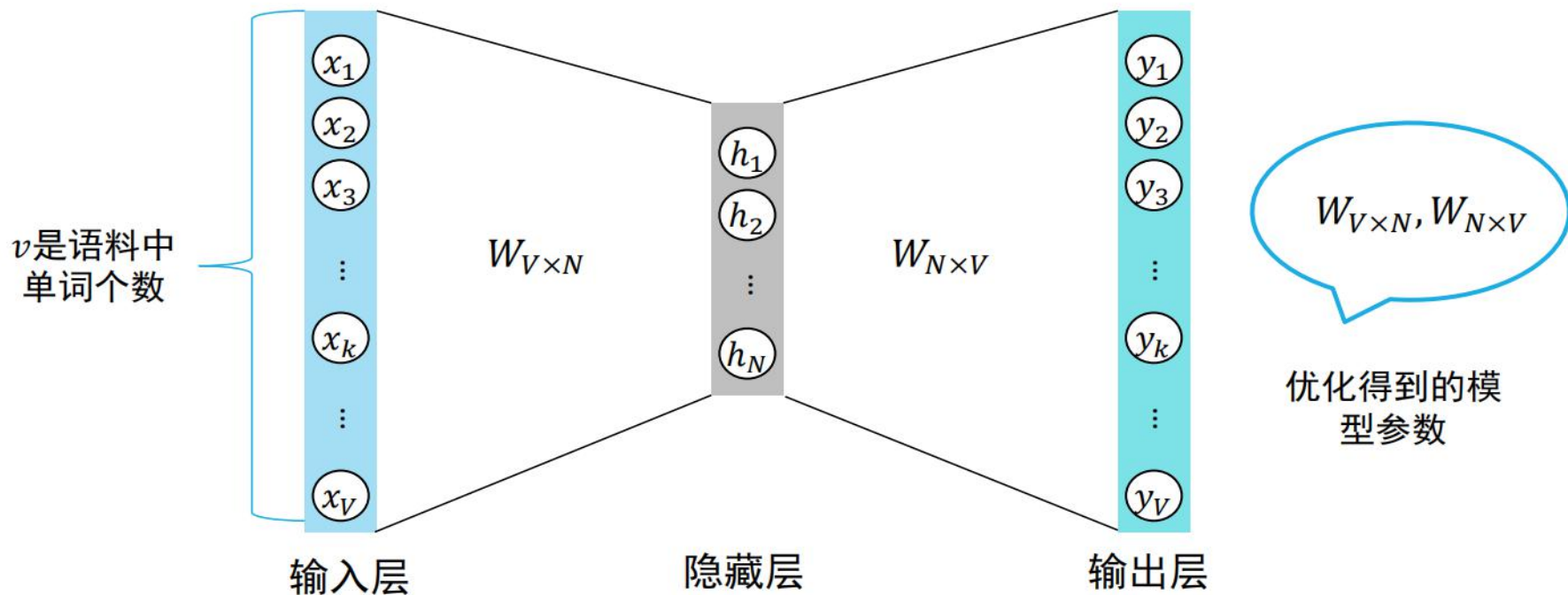
- 如下优化模型参数 Θ ，以最大化训练数据的对数似然函数

$$J = \max_{\theta} (\log f(w_t, w_{t-1}, \dots, w_{t-n+2}, w_{t-n+1}; \theta) + R(\theta))$$

- Yoshua Bengio, A Neural Probabilistic Language Model, Journal of Machine Learning Research 3 (2003) 1137–1155

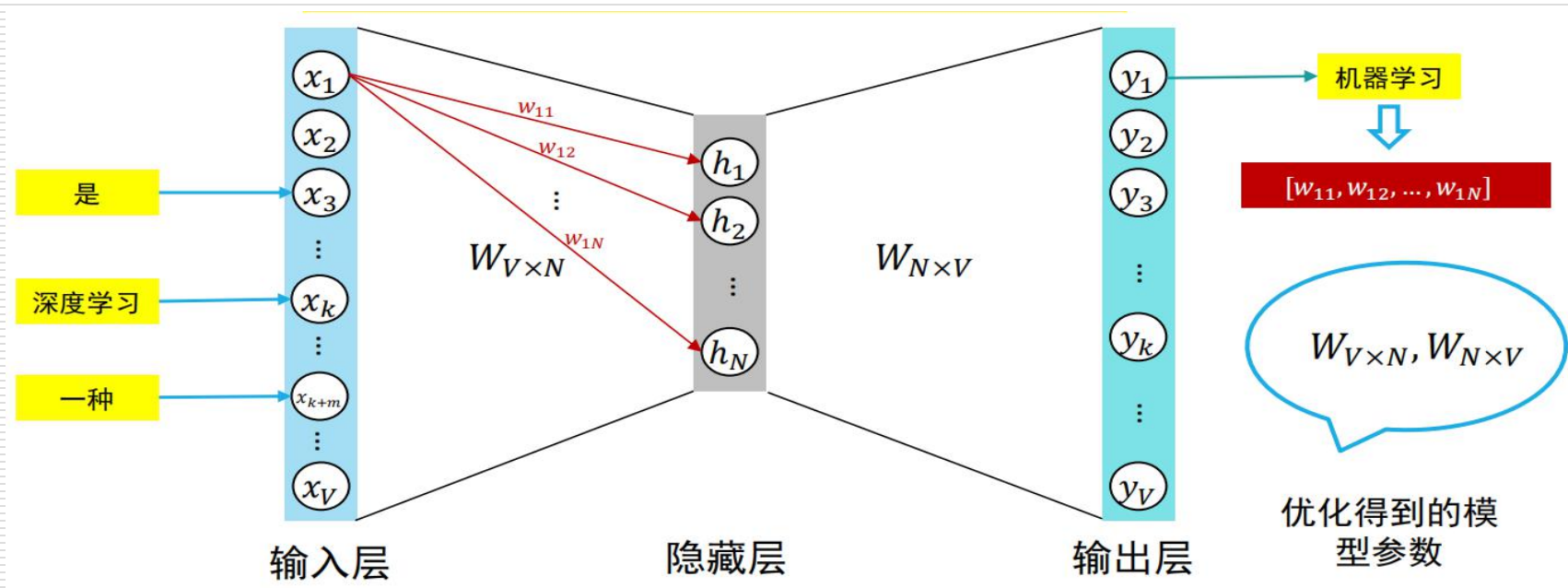
词向量模型的基本思想

- 词向量模型由一层输入层，一层隐藏层，一层输出层构成：实现了每个单词N维向量的表达



词向量模型的基本思想

- 词向量模型由一层输入层，一层隐藏层，一层输出层构成：实现了每个单词N维向量的表达
- 深度学习 是一种 机器学习 的方法



词向量模型：两种训练模式

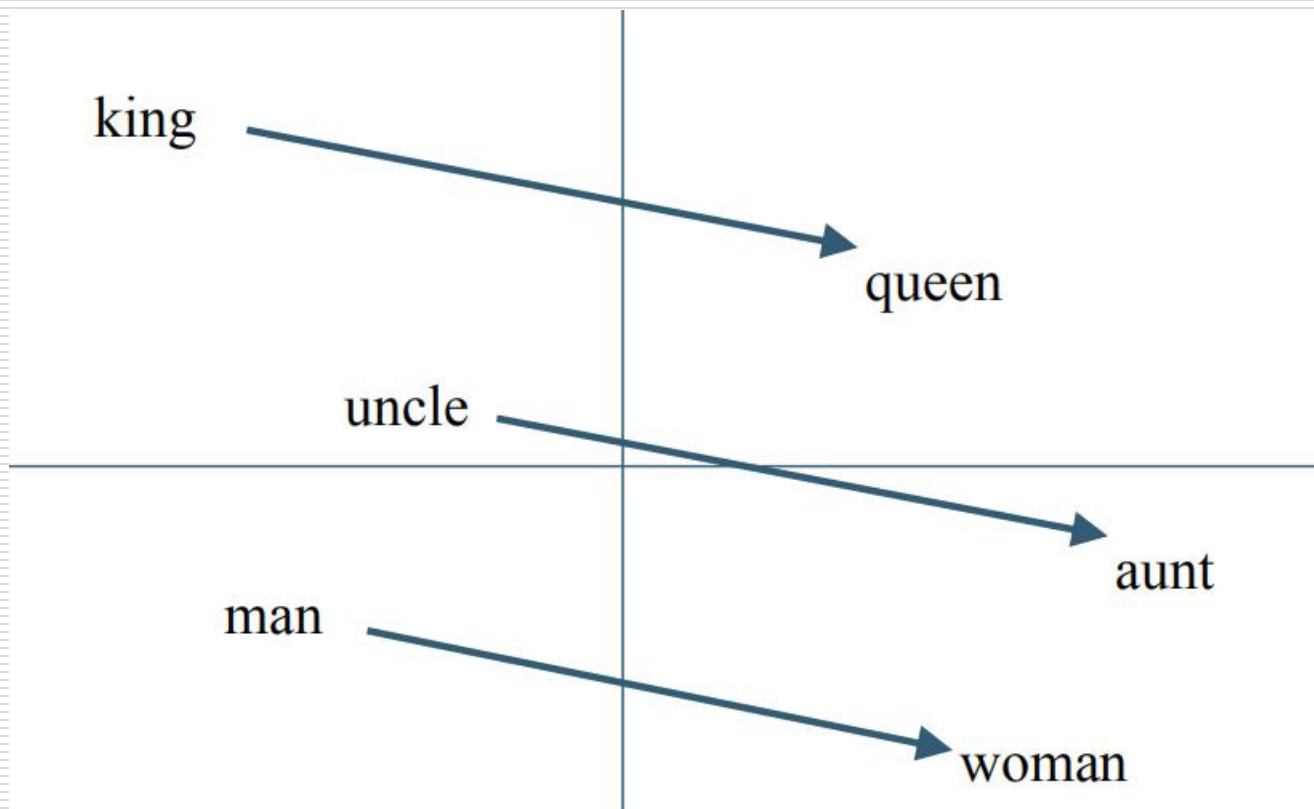
- **Continue Bag-of-Words (CBoW):** 根据某个单词的上下文单词来预测该单词
- **Skip-gram:** 利用某个单词来分别预测该单词的上下文单词

Word2Vec的改进算法

- 对一个包含10000个单词的语料库，每个单词的词向量设为200维，则需要 $200 \times 10000 (2000000)$ 和 $10000 \times 200 (2000000)$ 异常庞大的权重矩阵
- 在如此大神经网络上进行梯度下降耗时
- 为了解决这个不足，后续出现了如下改进手段：
 - Hierarchical Softmax (引入霍夫曼树)
 - Negative Sampling

基于词向量的操作：单词类比

□ king - man + woman = queen



对联项目

□ AI机智对对联: <https://ai.binwang.me/couplet/>

欢迎使用自动对对联系统

对联小贴士：本系统暂时不支持繁体字和特殊符号，断句请用全角逗号分隔。

一枝独秀 对下联

上联：一 枝 独 秀

下联：万 木 争 春

https://blog.csdn.net/m0_38106923

seq2seq模型

- 目前自然语言处理技术中非常重要而且非常流行的一个模型
- 该技术突破了传统的固定大小输入问题框架，开通了将经典深度神经网络模型运用于翻译与智能问答这一类序列型任务的先河，并且被证实各主流语言之间的相互翻译以及语音助手人机短问快答的应用中有着非常好的表现

seq2seq模型思想

- 2014年，是由Google Brain团队和Yoshua Bengio 两个团队各自独立的提出来
- 一个翻译模型，把一个语言序列翻译成另一种语言序列，整个处理过程是通过使用深度神经网络(LSTM或者RNN) 将一个序列作为输入影射为另外一个输出序列

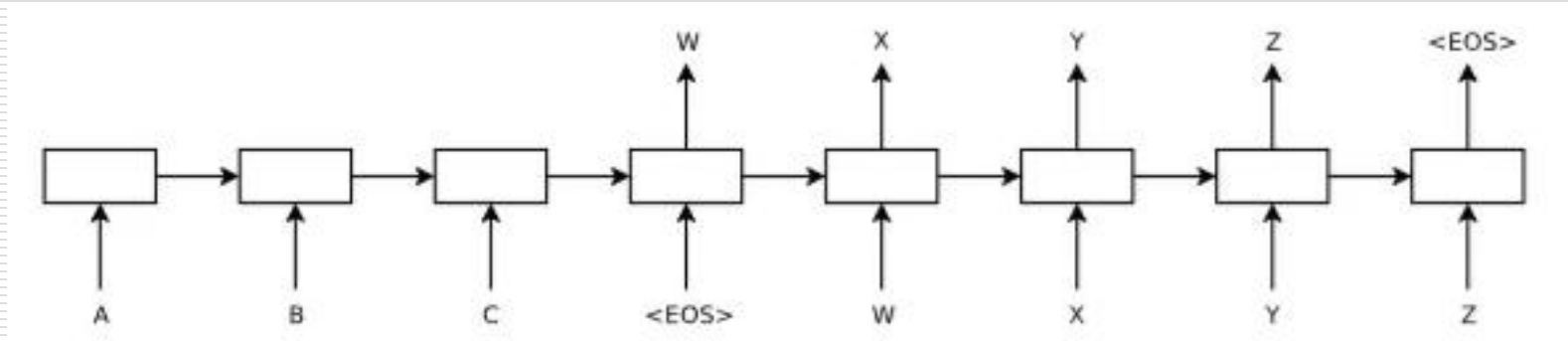
seq2seq模型思想

□ Encoder

- 左边使用一个神经网络，接收输入序列"A B C EOS (EOS=End of Sentence, 句末标记)", 在这个过程中每一个时间点接收一个词或者字，并在读取的EOS时终止接受输入，最后输出一个向量作为输入序列的语义表示向量

□ Decoder

- 第二个神经网络接收到第一个神经网络产生的输出向量后输出相应的输出语义向量，并且在这个时候每一个时刻输出词的概率都与前一个时刻的输出有关系，模型会将这些序列一次映射为"W X Y Z EOS"



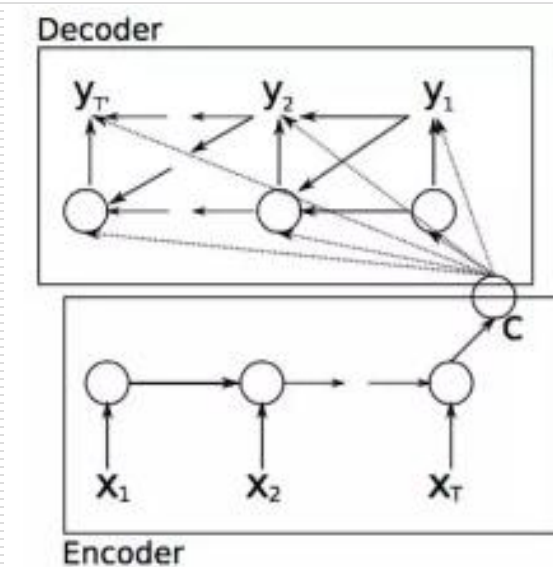
seq2seq模型思想

□ 整个过程的结构



seq2seq模型

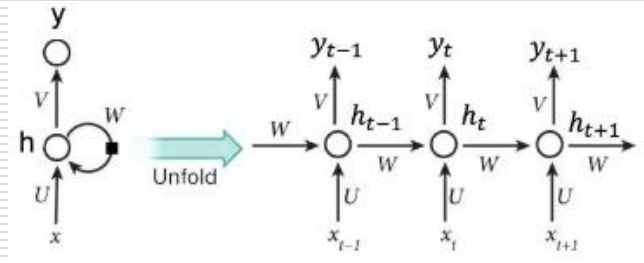
- 使用的是RNN网络作为基本的神经网络对输入序列和输出序列进行学习
- 整个模型分为解码和编码的过程，编码的过程结束后输出一个语义向量 \mathbf{c} ，之后整个解码过程根据 \mathbf{c} 进行相应的学习输出



seq2seq模型

□ 编码过程

- RNN网络学习的的过程，最后输出一个向量c



- 其中h是隐藏层，y是输出层，输入是一个时间序列 $x = (x_1, x_2, \dots, x_T)$ ，对于每一个时间t，RNN中隐藏层的h的更新由下面的表达式决定：

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

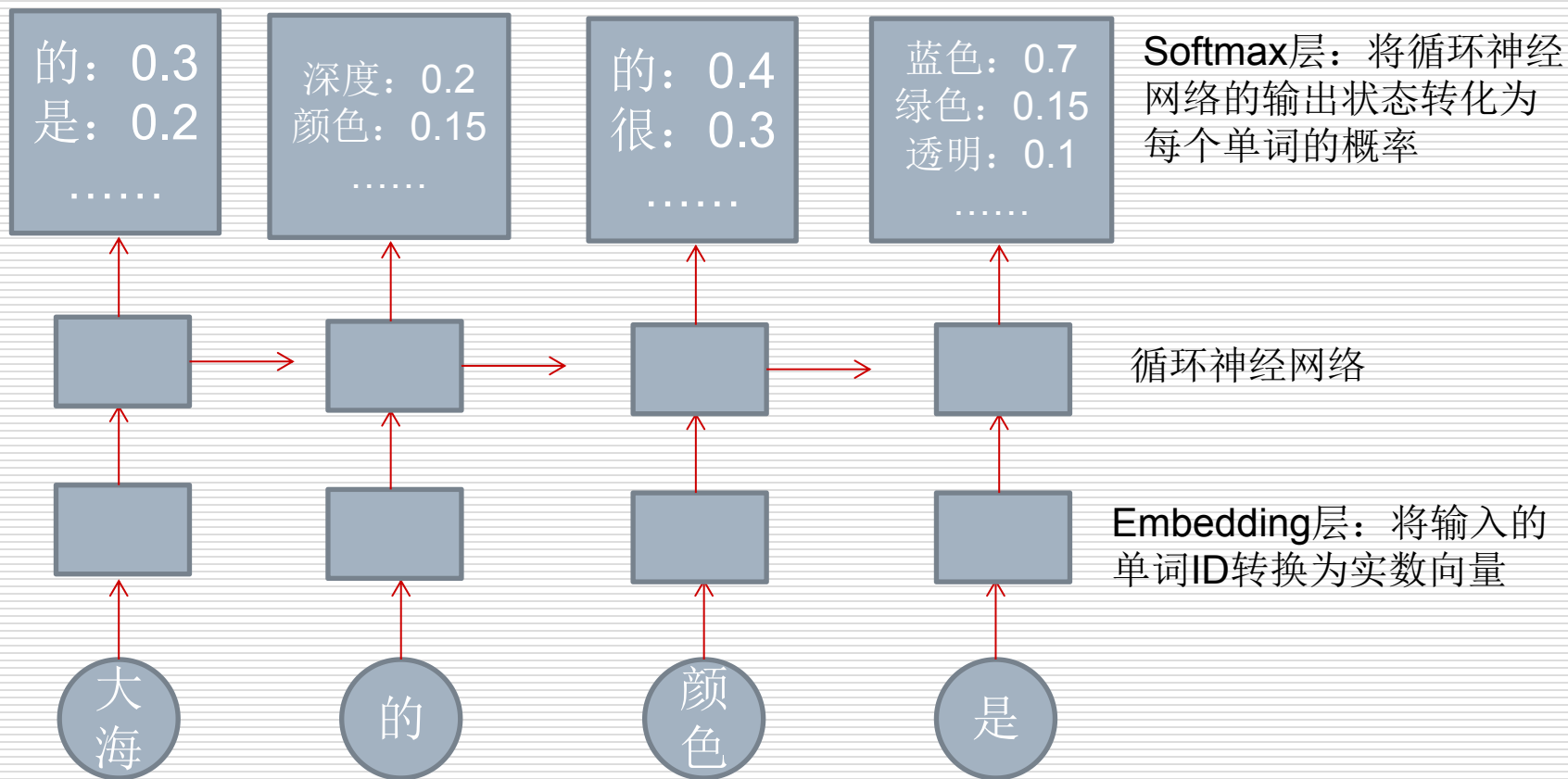
- RNN网络可以通过学习整个输入序列的概率分布来对下一个字或者词进行预测。对于时间t时，其概率分布为 $P(x_t | x_{t-1}, \dots, x_1)$

$$p(x_{t,j} = 1 | x_{t-1}, \dots, x_1) = \frac{\exp(w_j h_{(t)})}{\sum_{j'=1}^K \exp(w_{j'} h_{(t)})}$$

- 对j遍历词袋中可能的值，就可以得到每个字或者词在下一个时间出现的是概率值

seq2seq模型

□ 编码过程



seq2seq模型

□ 解码过程

- 对应的是另外一个RNN网络，其隐藏层状态在t时刻的更新根据如下方程进行更新：

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{t-1}, \mathbf{c})$$

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, \mathbf{c}) = g(\mathbf{h}_{(t)}, y_{t-1}, \mathbf{c})$$

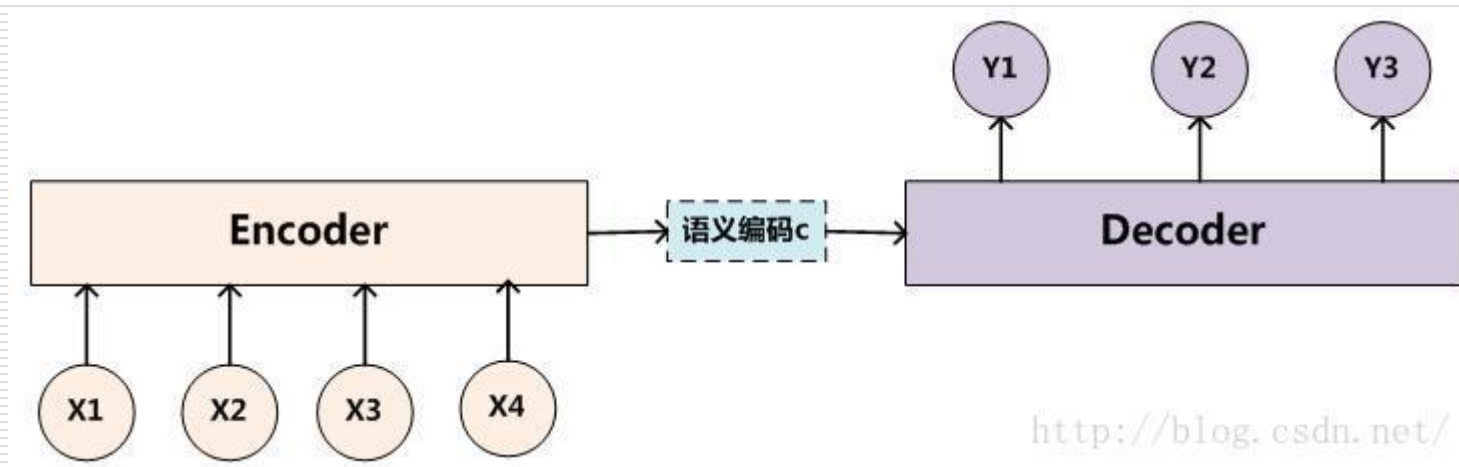
- 对于整个输入编码和解码的过程中，使用梯度优化算法以及最大似然条件概率为损失函数去进行模型的训练和优化：

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log p_{\theta}(y_n | \mathbf{x}_n),$$

- 其中 θ 为相应模型中的参数， (\mathbf{x}_n, y_n) 是相应的输入和输出的序列

使用Encoder-Decoder模型自动生成对联

- ❑ Encoder-Decoder框架可以看作是一种文本处理领域的研究模式，应用场景异常广泛



使用Encoder-Decoder模型自动生成对联

□ Encoder-Decoder模型

- 适合处理由一个句子（或篇章）生成另外一个句子（或篇章）的通用处理模型。
- 对于句子对 $\langle X, Y \rangle$ ，我们的目标是给定输入句子 X ，期待通过Encoder-Decoder框架来生成目标句子 Y 。 X 和 Y 可以是同一种语言，也可以是两种不同的语言。而 X 和 Y 分别由各自的单词序列构成：

$$X = \langle x_1, x_2 \dots x_m \rangle$$

$$Y = \langle y_1, y_2 \dots y_n \rangle$$

使用Encoder-Decoder模型自动生成对联

□ Encoder-Decoder模型

■ Encoder

- 对输入句子X进行编码，将输入句子通过非线性变换转化为中间语义表示C:

$$C = \mathcal{F}(x_1, x_2 \dots x_m)$$

■ Decoder

- 根据句子X的中间语义表示C和之前已经生成的历史信息 y_1, y_2, \dots, y_{i-1} 来生成i时刻要生成的单词 y_i

$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

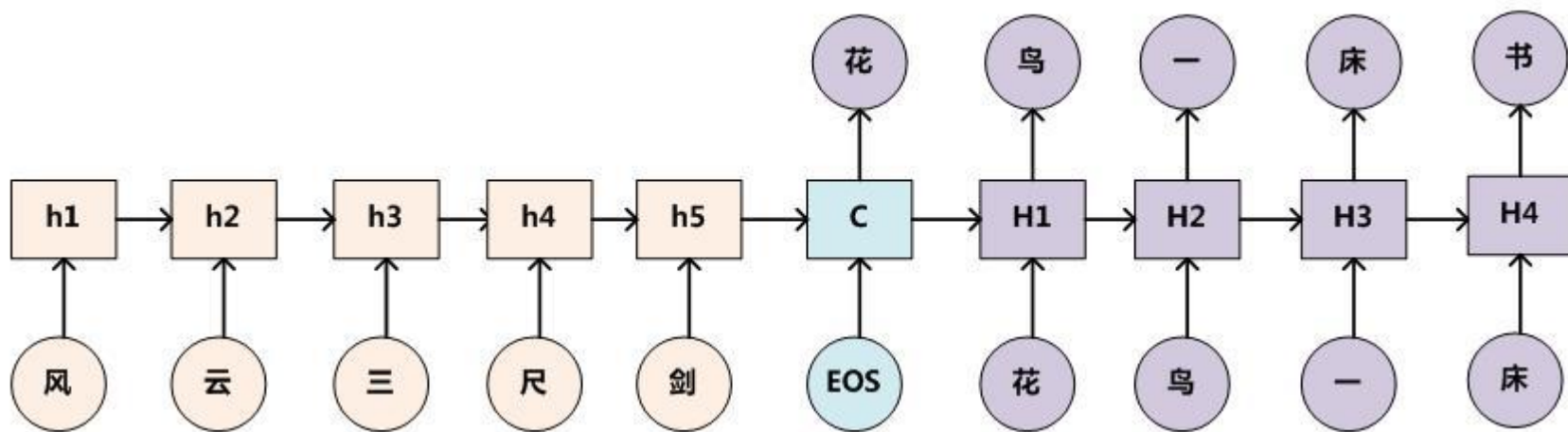
- 每个 y_i 都依次这么产生，那么看起来就是整个系统根据输入句子X生成了目标句子Y

使用Encoder-Decoder模型自动生成对联

- 一种情形是：假设对联的上联是已经知道的，比如人自己想的，任务是由机器来自动产生下联；
- 第二种情况是：假设要求上下联全部都由机器自动生成。

使用Encoder-Decoder模型自动生成对联

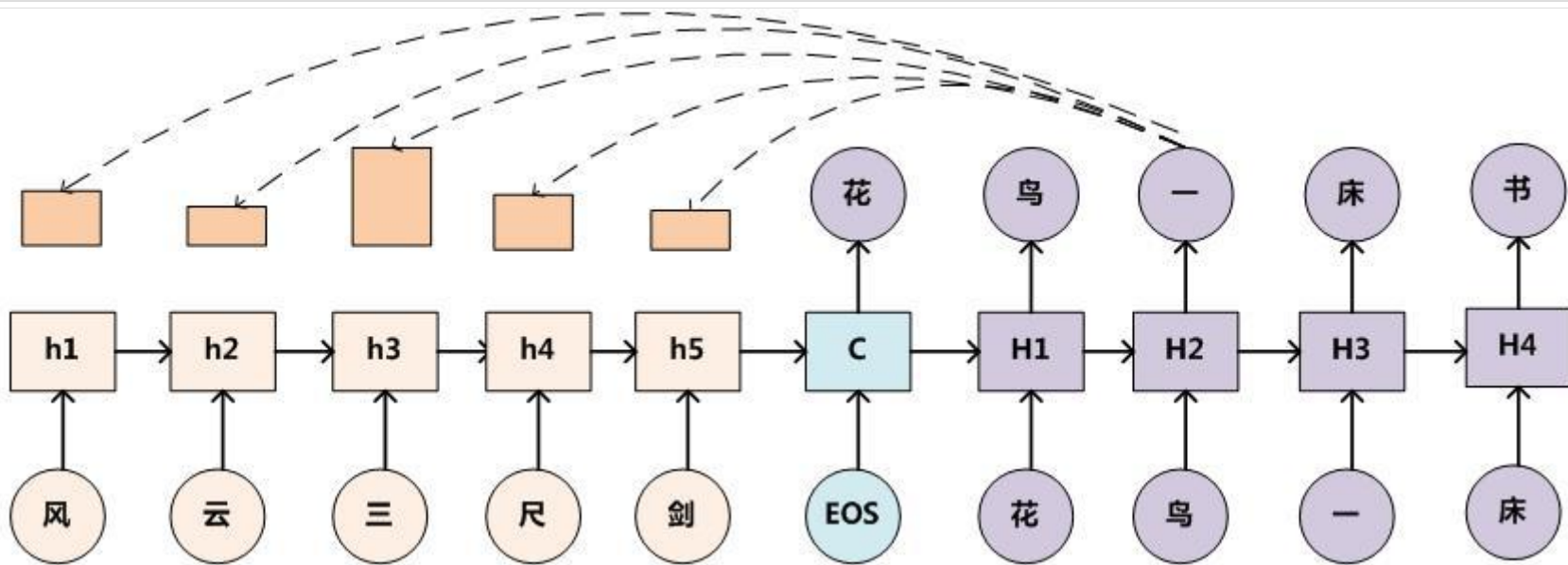
- 情形一：已知上联，机器自动生成下联
 - 例如：“风云三尺剑”，如何让机器自动生成下联？
 - 配置好Encoder-Decoder框架的具体模型，比如Encoder和Decoder都采用RNN模型



使用Encoder-Decoder模型自动生成对联

□ Encoder-Decoder模型+Attention

- 提升产生下联的质量，原因还是因为它是要求严格对仗



使用Encoder-Decoder模型自动生成对联

□ Encoder-Decoder模型问题

■ 语义和语境

□ “雨风万丈刀”，单看每个字对仗的都很工整，但是作为一个整体，语义看上去不那么协调

■ 上下联对应汉字的平仄问题

使用Encoder-Decoder模型自动生成对联

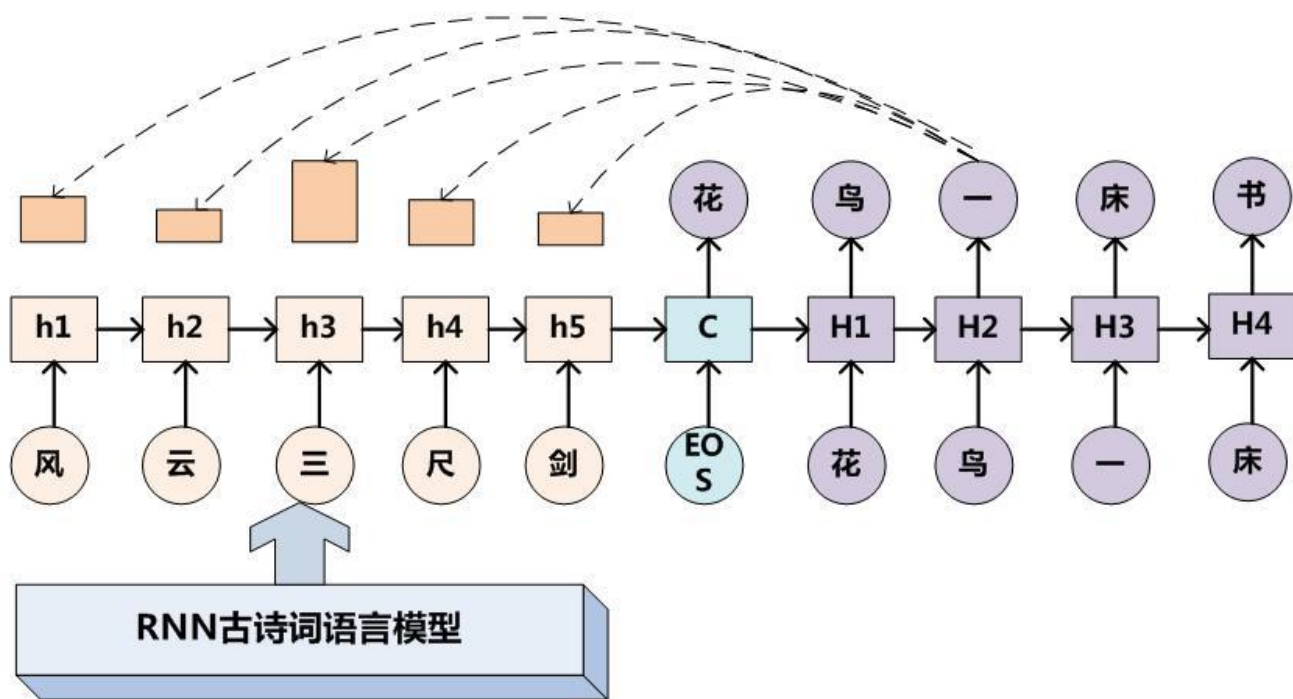
□ 情形二：对联由机器完全自动生成

- 第一步是不论用什么方法，先生成一句上联。第二步根据上联自动生成下联。
- 第二步可以使用情形一训练出的模型来做。
- 情形二的关键问题转换为：如何在一无所知情况下生成一句上联？

使用Encoder-Decoder模型自动生成对联

□ 情形二：对联由机器完全自动生成

■ 使用RNN构建一个古诗词的语言模型，然后上联通过这个RNN语言模型自动生成



模型扩展

- ❑ 在Encoder-Decoder模型中，Encoder只将最后一个输出递给Decoder，Decoder只知道梗概意思，而无法得到更多输入的细节。
- ❑ 这个模型的局限性在于编解码之间唯一的联系是固定长度的语义向量 \mathbf{C} 。
- ❑ 而这个向量无法完全表示整个序列的信息。此外，先输入的内容携带的信息会被后输入的信息覆盖，输入序列越长，问题越严重。这就使得在解码时没有办法获得足够的信息，解码的准确性就不会太高

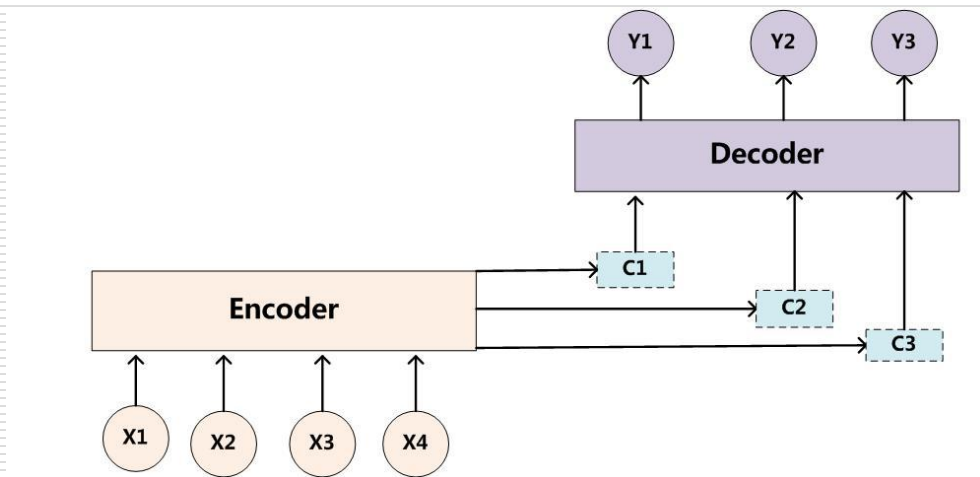
模型扩展

❑ 存在问题

- 起始的时间序列被编码转化成语义向量 \mathbf{c} ，之后再被解码，那么一开始的信息经过长时间的从左往右传播已经丢失了很多，而最后编码的信息也是在最后解码
- ❑ 在对输入的序列编码时，使用倒序输入，也就是原始的输入顺序为" $\mathbf{A} \ \mathbf{B} \ \mathbf{C}$ "的，那么新的方式编码的输入方式为" $\mathbf{C} \ \mathbf{B} \ \mathbf{A}$ "，这样 \mathbf{A} 编码成 \mathbf{c} 之后，就会马上进行解码，这样丢失的信息就没有之前那么多，经过这样的处理之后，取得了很大的提升，而且这样处理使得模型能够很好的处理长句子
- ❑ 但是对于输入词 \mathbf{C} 来讲，信息丢失的依旧很厉害
- ❑ 注意力机制。
 - 除了对最后的语义向量进行提取信息，还会对每一时刻的 \mathbf{h}_t 输出的结果进行信息的提取，这样Encoder过程中的隐藏状态都被利用上了

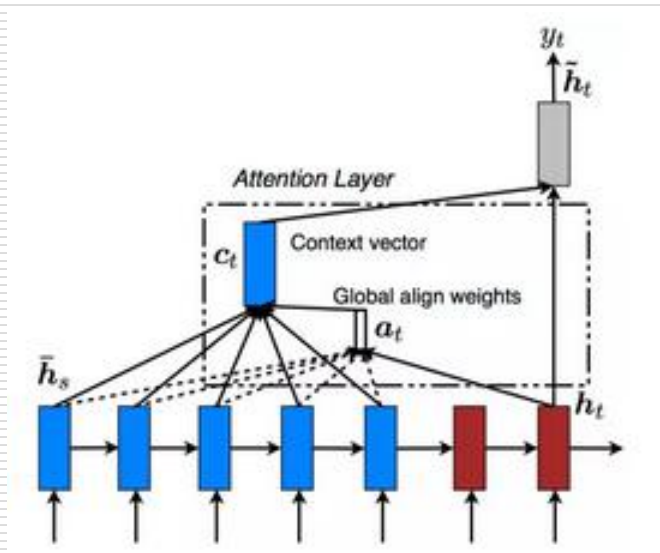
注意力模型

- 注意力模型来源于认知心理学，是指人们会因为关注整体的某一个局部而忽略其余部分，对于整体而言，关注度是有权重区分的，这是核心思想。注意力模型示意图如下图所示。



模型扩展

□ 注意力机制

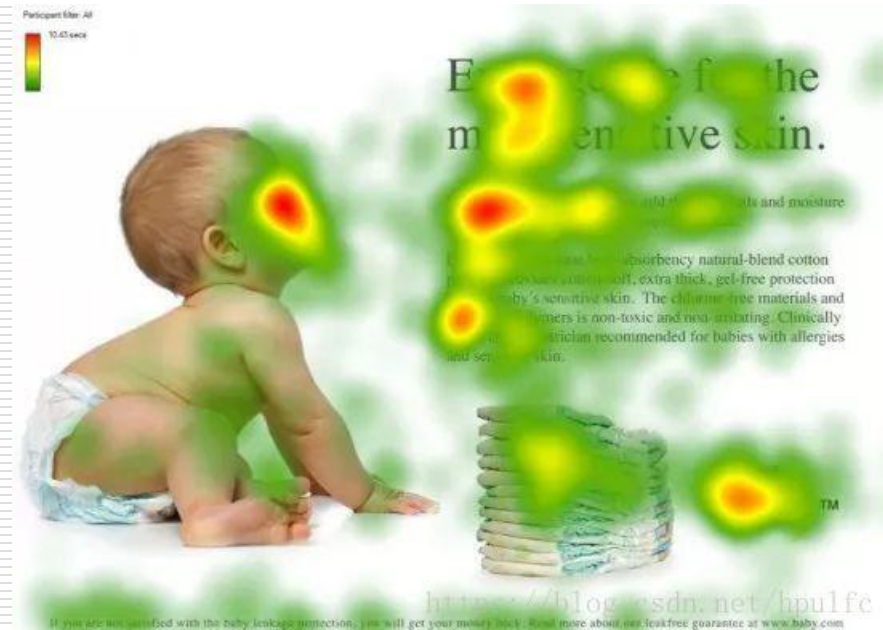


- 其中 c_t 和 a_t 为注意力层，对每一个隐藏层的状态都提取相应的信息，之后再整体的信息给编码层

Attention

□ 人类视觉的选择性注意力机制

- 人类视觉所特有的大脑信号处理机制
- 人类视觉通过快速扫描全局图像，获得需要重点关注的目标区域，也就是一般所说的注意力焦点，而后对这一区域投入更多注意力资源，以获取更多所需要关注目标的细节信息，而抑制其他无用信息
- 红色区域表明视觉系统更关注的目标
 - 人的脸部，文本的标题以及文章首句等位置



Attention

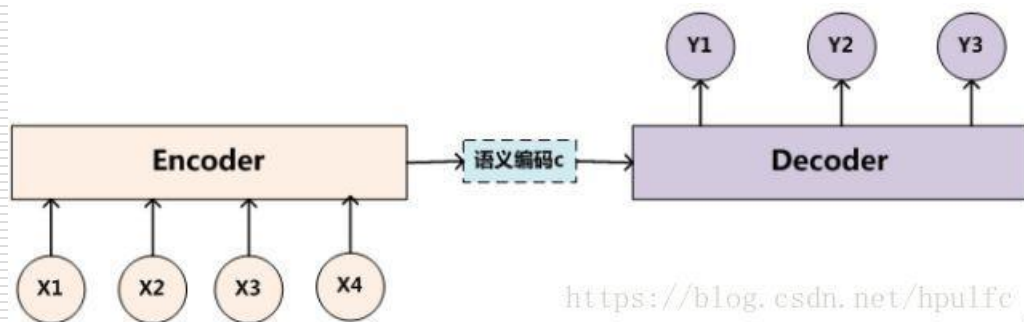
□ 深度学习中的注意力机制

- 核心目标也是从众多信息中选择出对当前任务目标更关键的信息

$$y_1 = f(C)$$

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

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



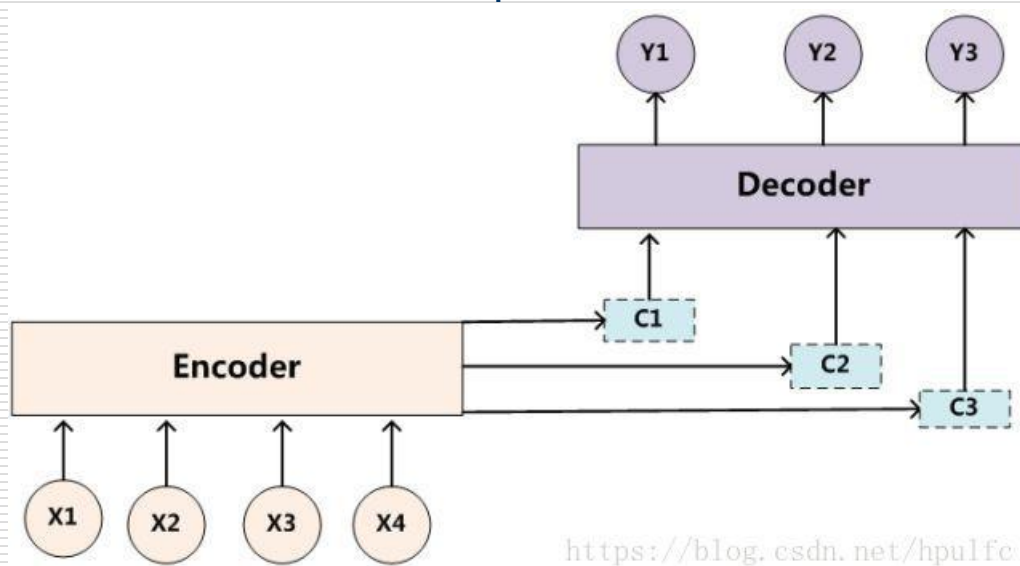
- 在生成目标句子的单词时，不论生成哪个单词，它们使用的输入句子Source的语义编码C都是一样的，没有任何区别
- 比如输入的是英文句子：Tom chase Jerry，Encoder-Decoder框架逐步生成中文单词：“汤姆”，“追逐”，“杰瑞”
 - 显然“Jerry”对于翻译成“杰瑞”更重要，但是模型无法体现这一点
 - 如果引入Attention模型的话，应该在翻译“杰瑞”的时候，体现出英文单词对于翻译当前中文单词不同的影响程度，比如给出类似下面一个概率分布值：（Tom,0.3）(Chase,0.2) (Jerry,0.5)

Attention

□ Attention模型的关键

- 由固定的中间语义表示C换成了根据当前输出单词来调整成加入注意力模型的变化了的 C_i

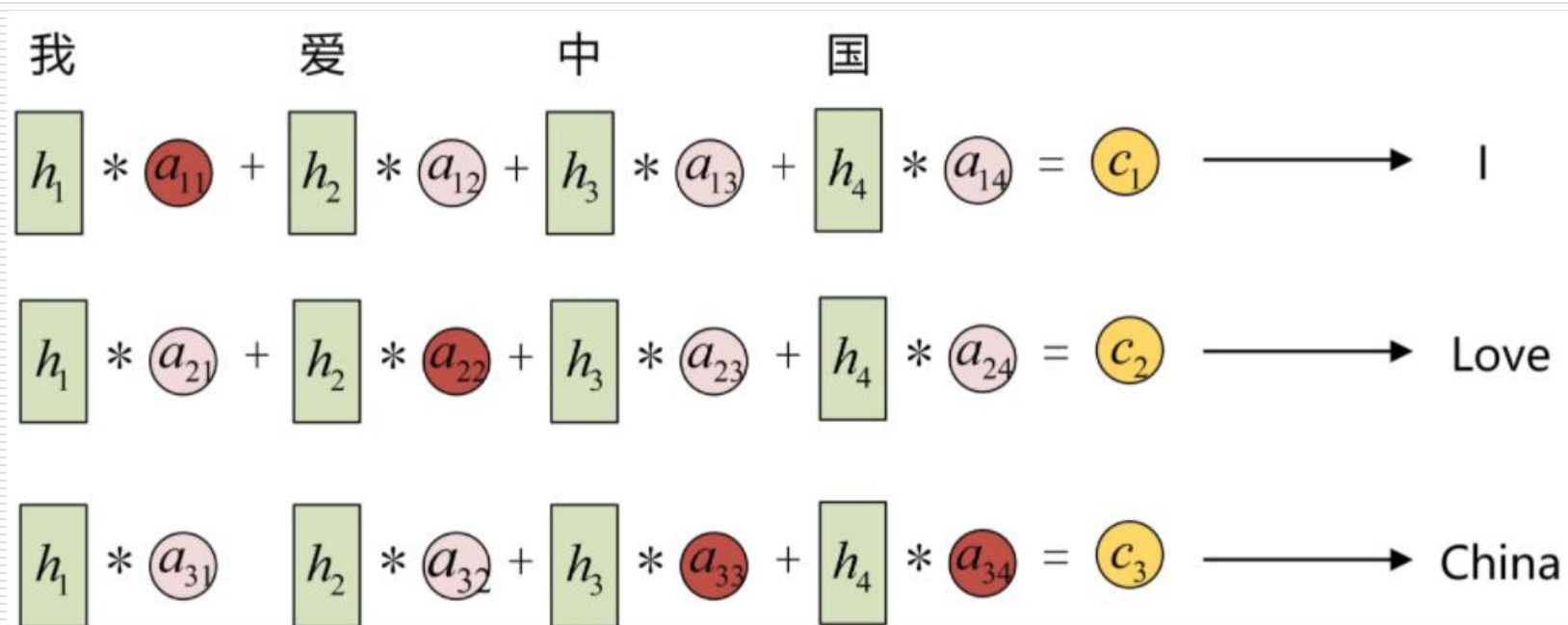
$$\begin{aligned}y_1 &= f1(C_1) \\ y_2 &= f1(C_2, y_1) \\ y_3 &= f1(C_3, y_1, y_2)\end{aligned}$$



- 每个 C_i 可能对应着不同的源语句单词的注意力分配概率分布

Attention

□ 以机器翻译为例



Attention

□ Tom chase Jerry

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

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

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

- f2函数代表Encoder对输入英文单词的某种变换函数
 - 如果Encoder采RNN模型，f2函数的结果往往是某个时刻输入 x_i 后隐层节点的状态值；
- g代表Encoder根据单词的中间表示合成整个句子中间语义表示的变换函数，一般g函数就是对构成元素加权求和

$$C_i = \sum_{j=1}^{L_x} a_{ij} h_j$$

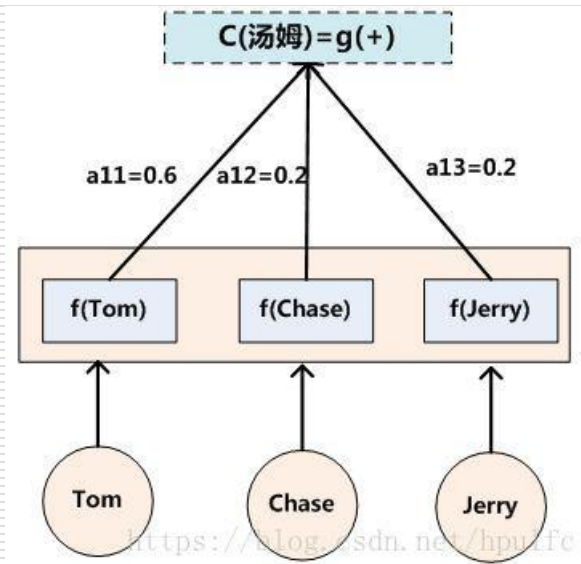
- L_x 代表输入句子Source的长度， a_{ij} 代表在Target输出第i个单词时Source输入句子中第j个单词的注意力分配系数，而 h_j 则是Source输入句子中第j个单词的语义编码

Attention

$$C_{\text{汤姆}} = g(0.6 * f_2(\text{"Tom"}), 0.2 * f_2(\text{Chase}), 0.2 * f_2(\text{"Jerry"}))$$

$$C_i = \sum_{j=1}^{L_x} a_{ij} h_j$$

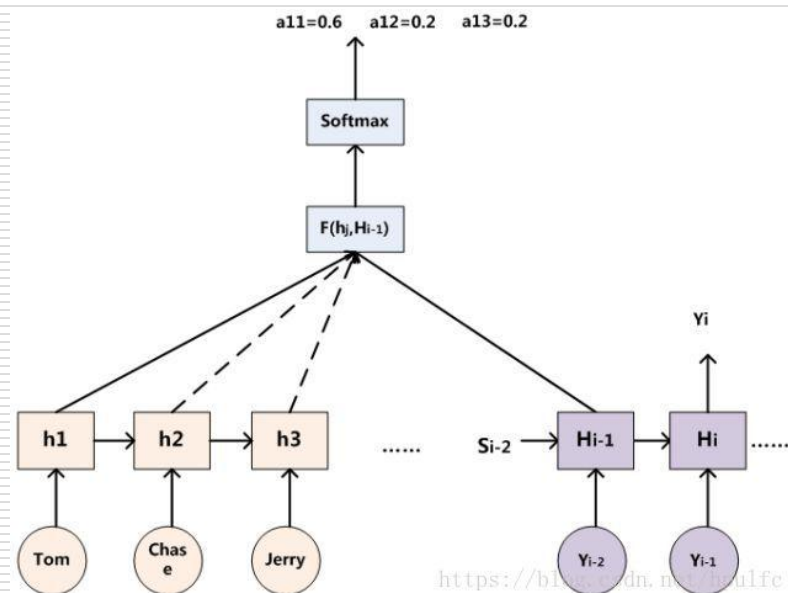
- $L_x=3$, $h_1=f(\text{"Tom"})$, $h_2=f(\text{"Chase"})$, $h_3=f(\text{"Jerry"})$ 分别是输入句子每个单词的语义编码，对应的注意力模型权值则分别是0.6,0.2,0.2
- Attention的形成过程



Attention

□ 注意力分配概率计算

- 对 Y_i 来说的注意力分配概率分布，那么可以用Target输出句子 $i-1$ 时刻的隐层节点状态 H_{i-1} 去一一和输入句子Source中每个单词对应的RNN隐层节点状态 h_j 进行对比，即通过函数 $F(h_j, H_{i-1})$ 来获得目标单词 y_i 和每个输入单词对应的对齐可能性，这个 F 函数在不同论文里可能会采取不同的方法，然后函数 F 的输出经过Softmax进行归一化就得到了符合概率分布取值区间的注意力分配概率分布数值

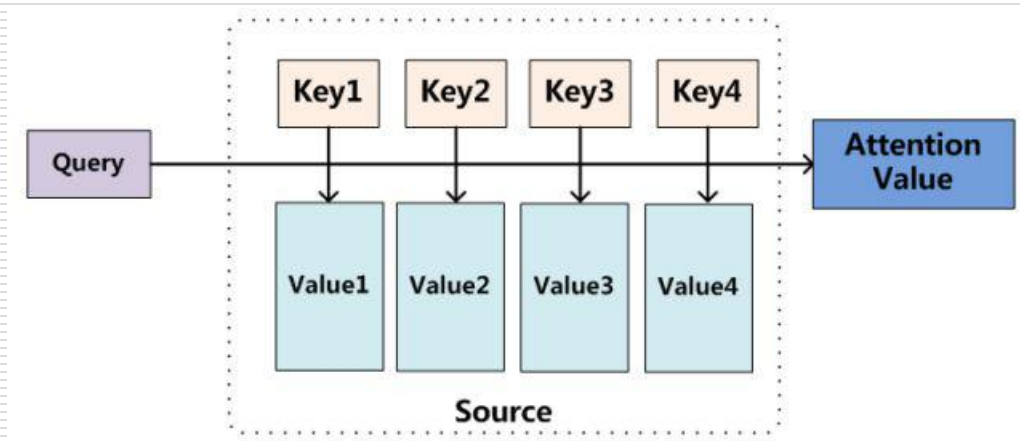


Attention机制的本质思想

- 对Source中元素的Value值进行加权求和，而Query和Key用来计算对应Value的权重系数

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

- $L_x = ||\text{Source}||$ 代表Source的长度



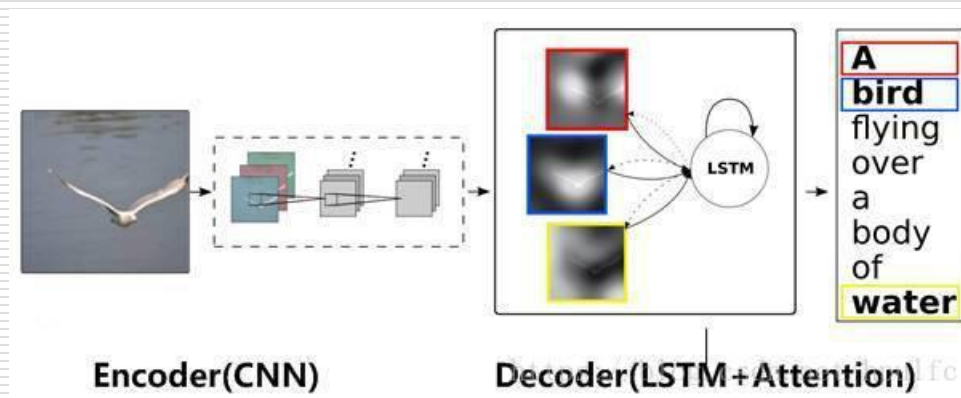
- 聚焦的过程体现在权重系数的计算上，权重越大越聚焦于其对应的Value值上，即权重代表了信息的重要性，而Value是其对应的信息

Attention机制的应用

□ 图像处理领域

■ 图片描述（Image-Caption）

- 一种典型的图文结合的深度学习应用，输入一张图片，人工智能系统输出一句描述句子，语义等价地描述图片所示内容
- **Encoder**输入部分是一张图片，一般会用**CNN**来对图片进行特征抽取
- **Decoder**部分使用**RNN**或者**LSTM**来输出自然语言句子

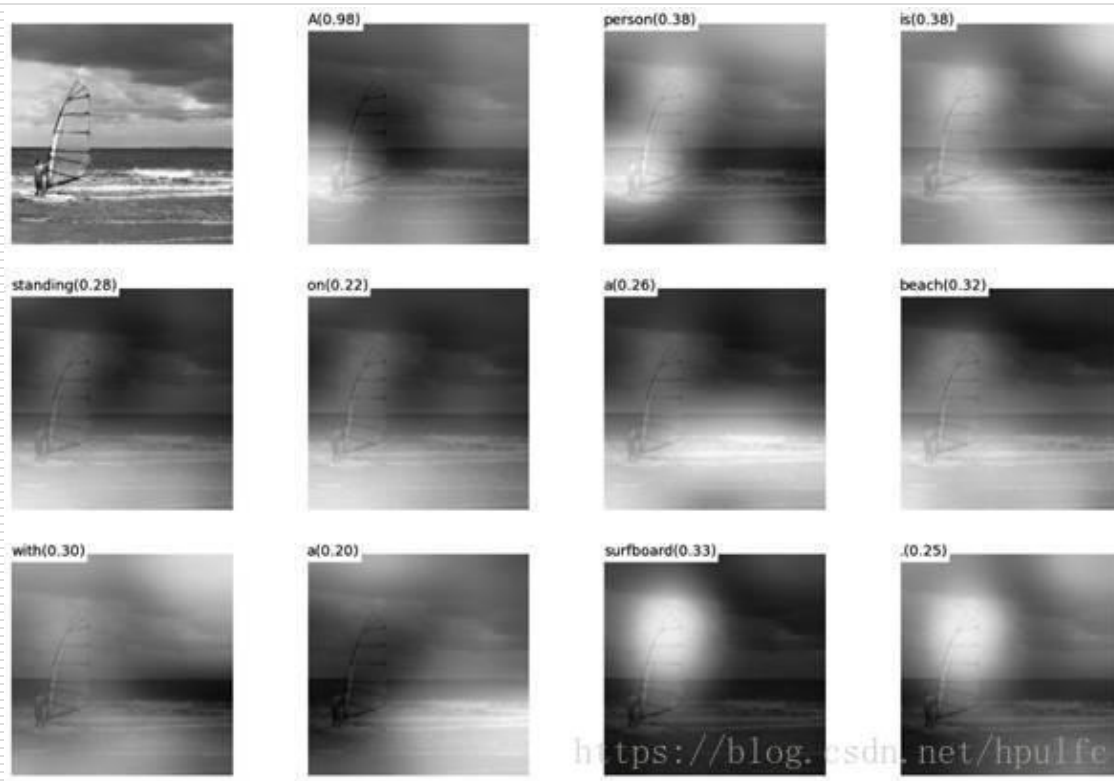


Attention机制的应用

□ 加入Attention机制

- 在输出某个实体单词的时候会将注意力焦点聚焦在图片中相应的区域上

□ “A person is standing on a beach with a surfboard.”



□ seq2seq对联生成解读（1）

- 了解每个文件是干什么的

- https://blog.csdn.net/weixin_41303016/article/details/88561102

□ seq2seq对联生成解读（2）

- seq2seq模型的构造(encoder)

- https://blog.csdn.net/weixin_41303016/article/details/88636766

□ seq2seq对联生成解读（3）

- seq2seq模型的构造(decoder)

- https://blog.csdn.net/weixin_41303016/article/details/88637632



THE END

视觉问答技术

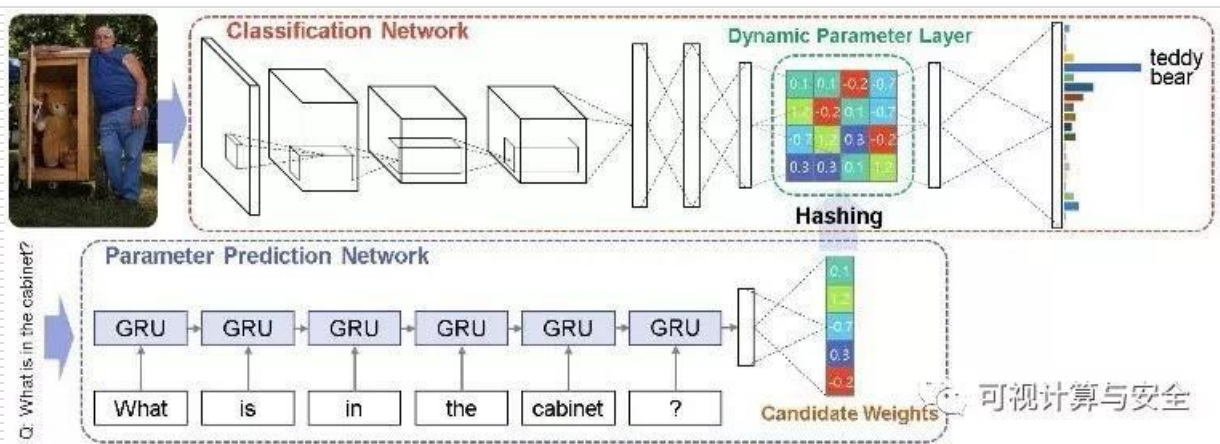
■ 基线模型（Baseline Models）

- 论文：H. Noh, P. H. Seo, and B. Han, “Image question answering using convolutional neural network with dynamic parameter prediction,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- 工作：通过学习一个动态参数层的卷积神经网络(CNN)，进行网络权值的自适应预测。
- 在自适应参数预测中，我们采用了一个单独的参数预测网络，该网络由以问题为输入的门控递归单元和生成一组候选权重作为输出的全连通层组成。通过使用哈希技术来减少这个问题的复杂性，使用预定义的哈希函数选择参数预测网络给出的候选权重，以确定动态参数层中的各个权重，进而提高基线模型的准确率。

视觉问答技术

■ 基线模型（Baseline Models）

□ 代码: <https://github.com/MarvinTeichmann/tensorflow-fcn>



视觉问答技术

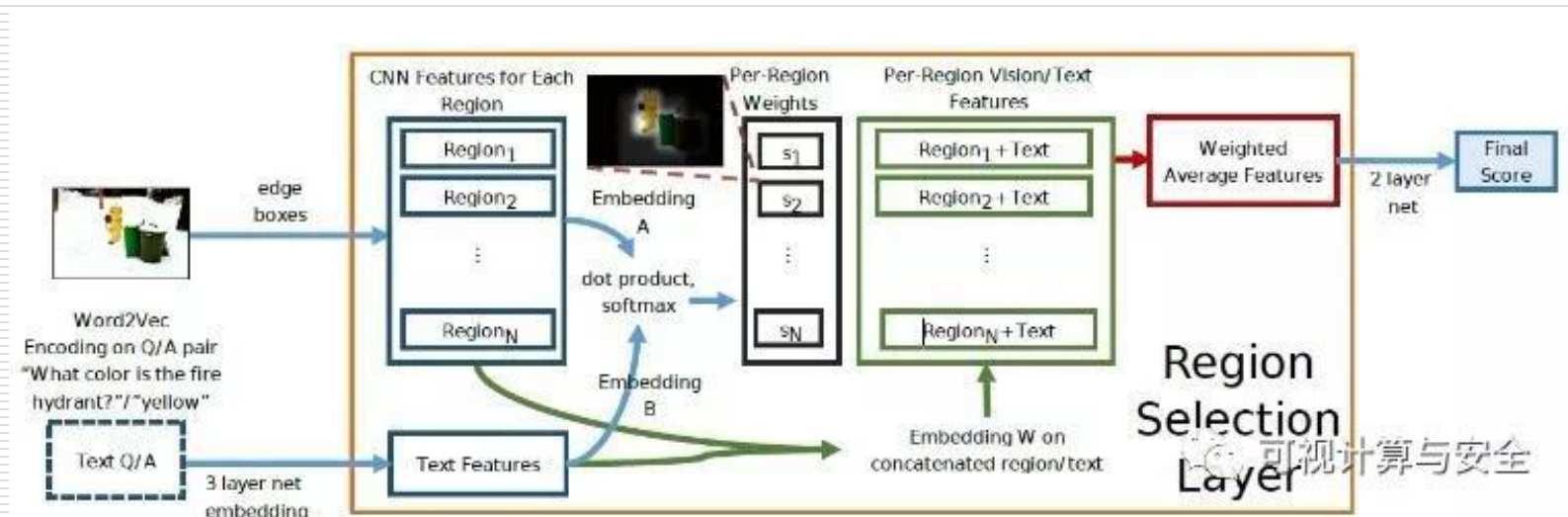
■ 注意力模型（Attention Based Models）

- 论文：K. J. Shih, S. Singh, and D. Hoiem, “Where to look: Focus regions for visual question answering,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- 工作：美国伊利诺伊大学的这项工作使用CNN来从这些盒子中提取特征。VQA系统的输入包括这些CNN特征，问题特征和一个选择性的答案。他们的系统被训练为每一个选择题的答案都能得到一个分数，并选出得分最高的答案。通过传递CNN区域特征点积以及问题，最后合并到一个全连接层中，可以简单地计算出权重的每个区域的加权平均得分。

视觉问答技术

■ 注意力模型（Attention Based Models）

□ 代码: https://github.com/kevjshih/wtl_vqa



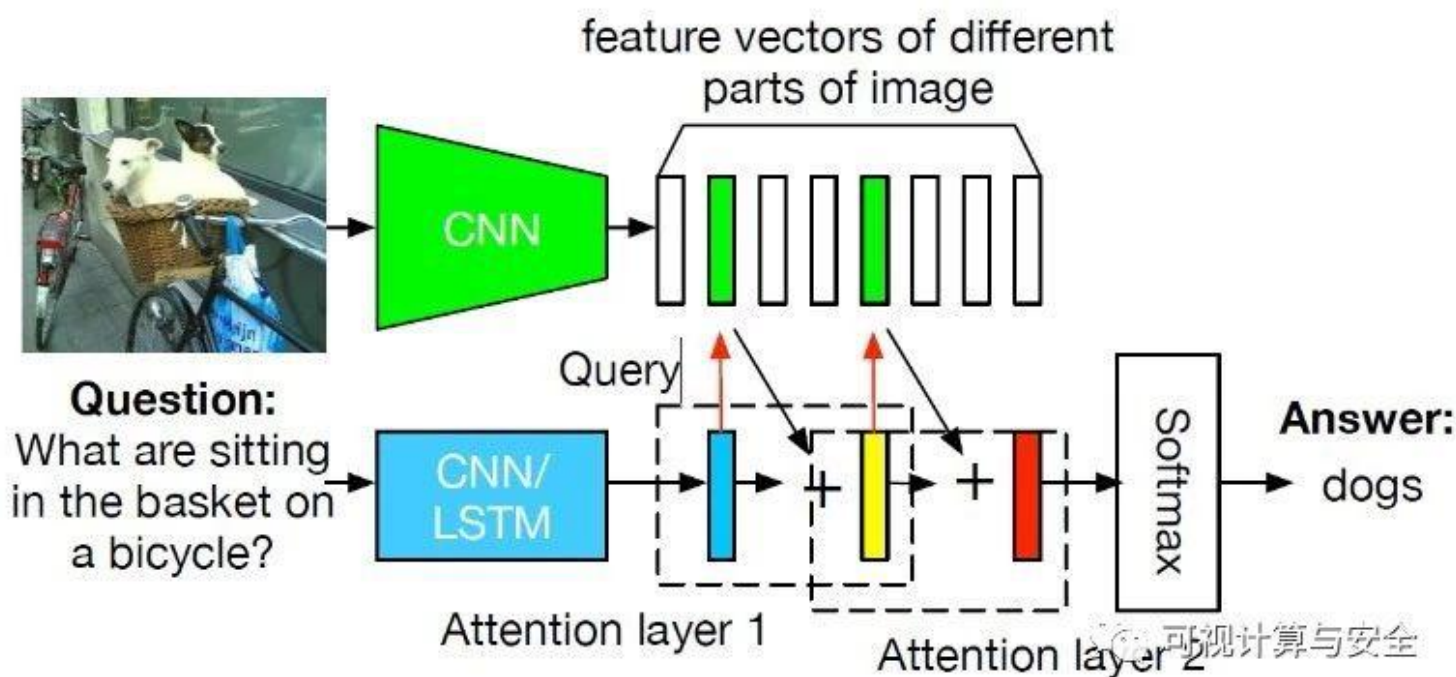
视觉问答技术

- 堆积注意力网络 Stacked Attention Network
 - 论文：Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, “Stacked attention networks for image question answering,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
 - 工作：美国卡耐基梅隆大学和微软提出的动态注意力网络。其注意层由一个单独的权重层确定，该权重层用问题和带激活函数的CNN的特征图计算图像位置上的注意力分布。然后将该分布应用到CNN的特征层中，使用加权和在空间特征位置之间进行聚合，从而生成一个完整的图像，它比其他区域更强调某些空间的区域。将这个特征向量与问题特征向量结合，可得到预测答案。

视觉问答技术

■ 堆积注意力网络 Stacked Attention Network

□ 代码: <https://github.com/zcyang/imageqa-san>



视觉问答技术

■ 空间记忆网络Spatial Memory Network

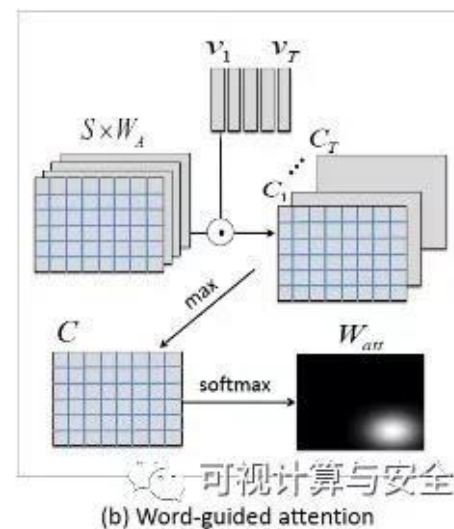
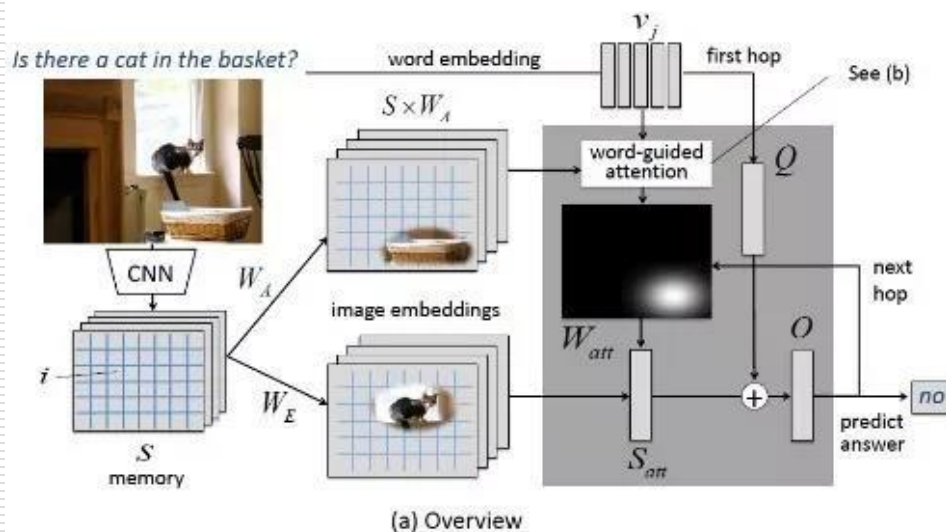
- 论文：H. Xu and K.Saenko, “Ask, attend and answer: Exploring question-guided spatial attention for visual question answering,” in European Conference on Computer Vision(ECCV), 2016.
- 工作：美国麻省理工学院提出的空间记忆网络，该模型通过估计图像斑块与问题中单个单词的相关性来产生空间关联性。这种文字引导的注意力被用来预测注意力分布，然后用来计算图像视觉特征的加权和。然后探索了两种不同的模型。在单跳模型中，将编码整个问题的特征与加权视觉特征相结合并预测答案。在两跳模型中，将视觉特征和问题特征的组合循环回到关注机制中，用于细化注意力分配。

视觉问答技术

■ 空间记忆网络 Spatial Memory Network

□ 代码:

https://github.com/VisionLearningGroup/Ask_Attend_and_Answer

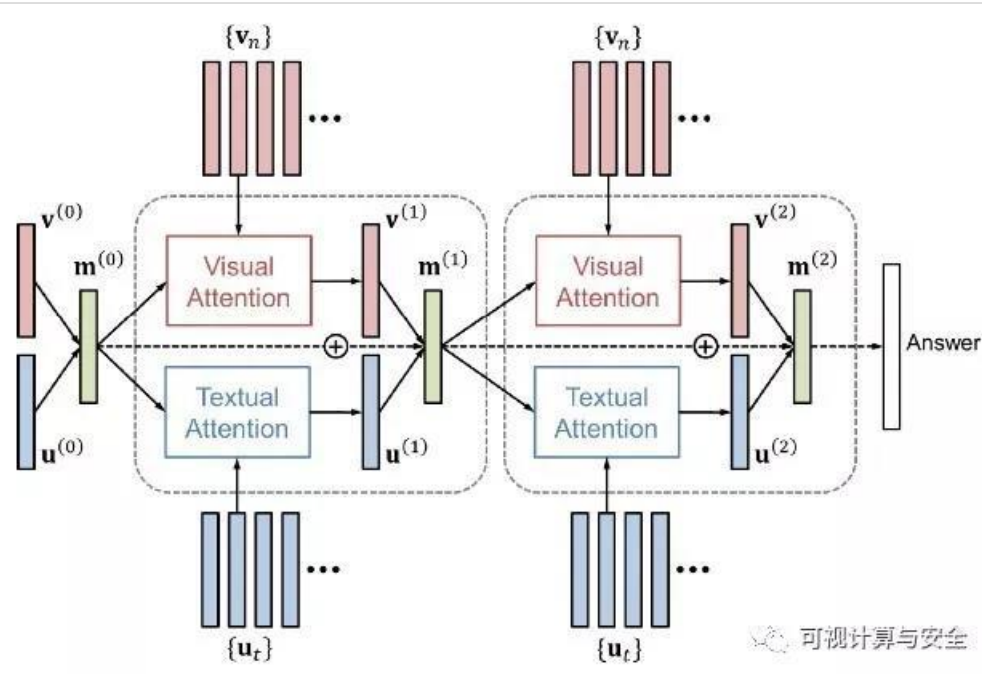


可视计算与安全

视觉问答技术

■ DAN

- 论文: H. Nam, J. Ha, and J. Kim. Dual attention networks for multimodal reasoning and matching. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- 代码: <https://github.com/iammrhelo/pytorch-vqa-dan>



视觉问答技术

■ 双线性融合（Bilinear Pooling Methods）

□ MLB

- 论文：J.-H.Kim, K.-W. On, J. Kim, J.-W. Ha, and B.-T. Zhang, “Hadamard product for low-rank bilinear pooling,” arXiv preprint arXiv:1610.04325, 2016.
- 工作：首尔国立大学的工作者们使用多模态低秩双线性池(MLB)方案，使用Hadamard乘积和线性映射来实现近似双线性池。当与空间视觉注意机制一起使用时，MLB可以与VQA中的MCB相媲美，但计算复杂度较低，并且使用的神经网络参数较少。
- 代码：<https://github.com/jnhwkim/MulLowBiVQA>

视觉问答技术

■ 双线性融合（Bilinear Pooling Methods）

□ MLB

- 论文：J.-H.Kim, K.-W. On, J. Kim, J.-W. Ha, and B.-T. Zhang, “Hadamard product for low-rank bilinear pooling,” arXiv preprint arXiv:1610.04325, 2016.
- 工作：首尔国立大学的工作者们使用多模态低秩双线性池(MLB)方案，使用Hadamard乘积和线性映射来实现近似双线性池。当与空间视觉注意机制一起使用时，MLB可以与VQA中的MCB相媲美，但计算复杂度较低，并且使用的神经网络参数较少。
- 代码：<https://github.com/jnhwkim/MulLowBiVQA>

视觉问答技术

■ 双线性融合（Bilinear Pooling Methods）

□ MCB

- 论文：A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, “Multimodal compact bilinear pooling for visual question answering and visual grounding,” in Conference on Empirical Methods on Natural Language Processing (EMNLP), 2016.
- 工作：美国加州大学和日本索尼公司在众多的基线模型中发现，由于在双线性融合的过程中，计算高维双线性外积通常是不可行的，而使用多模态双线性池(MCB)来高效地表达多模态特征。在视觉问答方面对MCB进行了范围较广的评价，取得了目前所有基线模型中效果最好的结果。
- 代码：<https://github.com/akirafukui/vqa-mcb>

视觉问答技术

■ 双线性融合（Bilinear Pooling Methods）

□ MFB

- 论文：Z. Yu, J. Yu, J. Fan, and D. Tao. Multi-modal Factorized Bilinear Pooling with Co-Attention Learning for Visual Question Answering. ArXiv e-prints, Aug. 2017.
- 工作：本文提出了一种多模态分解双线性池化方法，有效地结合了多模态特征，使VQA性能优于其他双线性融合方法。对于细粒度的图像和问题表示，我们开发了一种“共享注意力”机制，使用端到端深度网络架构来共同学习图像和问题注意力。
- 代码：<https://github.com/yuzccccc/vqa-mfb>

视觉问答技术

■ 双线性融合（Bilinear Pooling Methods）

□ MFH

- 论文：Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. TNNLS, 2018.
- 工作：针对多模态特征融合问题，作者提出了一种广义多模态因子分解的高阶池化方法(MFH)，通过充分利用多模态特征之间的相关性，实现多模态特征的更有效融合，进一步提高VQA性能。在答案预测中，利用KL (Kullback-Leibler)散度作为损失函数，可以更准确地表征具有相同或相似含义的多个不同答案之间的复杂关联，从而使我们能够获得更快的收敛速度，并在答案预测中获得略好的准确性。
- 代码：<https://github.com/yuzccccc/vqa-mfb>