Connectionist Temporal Classification(CTC)算法
问题阐述
RNN序列预测
CTC序列预测
CTC算法原理
优化目标
映射规则
训练和推理
求解CTC loss
确定合法路径
前后向计算路径和
CTC模型推理
贪心搜索（Greedy Search）
集束搜索（Beam Search）
前缀束搜索（Prefix Beam Search）

# Connectionist Temporal Classification(CTC)算法

参考资料1 ，参考资料2 参考资料3

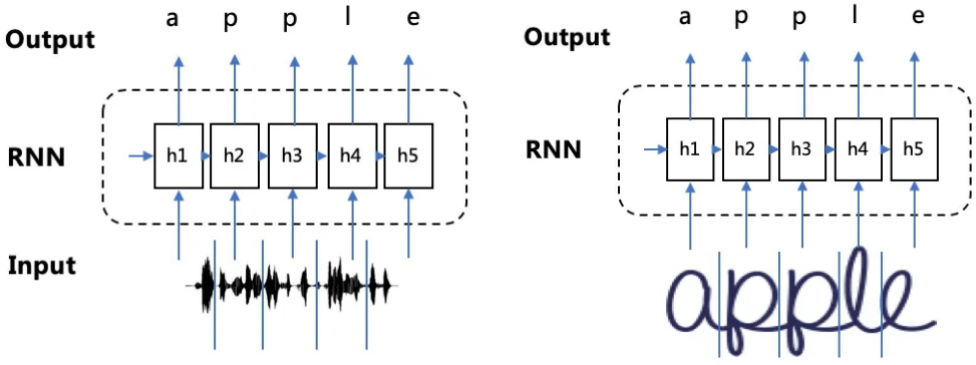
## 问题阐述

CTC算法是语音识别，OCR等领域常用的一种算法，其主要解决**序列到序列的预测问题**。

### RNN序列预测

RNN是解决序列预测问题的常用模型，但其一般要满足如下依赖条件：

- 输入序列和输出序列之间的映射关系需要事先标注好
- **输入序列和输出序列是一一对应的**

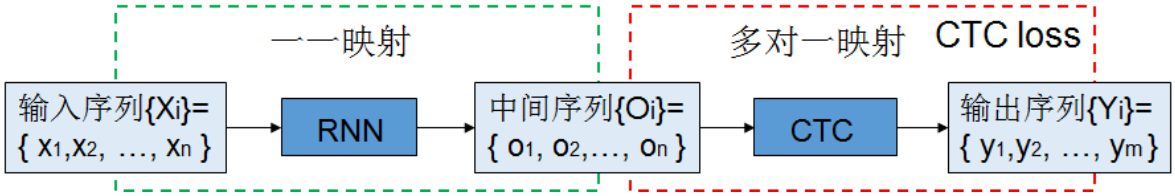



正因为输入输出一一对应，RNN模型能根据输出序列和标注样本间的差异直接定义Loss损失，完成**端到端的**训练。但在语音和文本识别领域，连续的音频信号和图像信号往往很难分割，输入序列和输出序列的映射关系没有提前标注好，传统的RNN训练方法不再适用。

### CTC序列预测

CTC算法主要在输入输出存在多对一映射的情况下，解决模型端到端训练的问题。

- 让RNN直接对序列数据进行学习，无需事先标注好训练数据中输入序列和输入序列的映射关系。
- 扩展了**RNN的输出层**，在输出序列和最终标签之间增加了多对一的空间映射。并定义CTC Loss函数。




- CTC Loss函数的求解，借鉴了HMM前后向算法的思路，利用动态规划解决了端到端训练的问题 。

# CTC算法原理

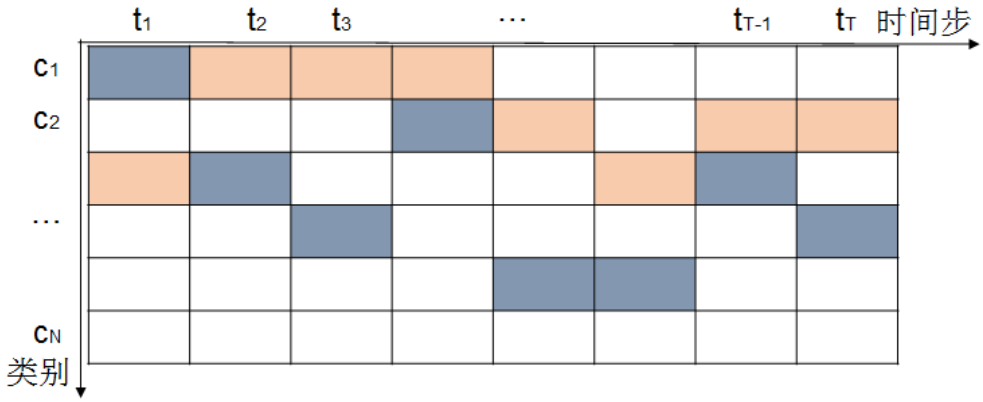
## 优化目标

CTC解决多对一映射问题。给定输入X，整体的优化目标是使输出Y的条件概率最大：

$$P(Y|X) = P(y_1, y_2, \ldots, y_m|x_1, x_2, \ldots, x_n)$$
$$= \sum_i P(O_i|(X)) = \sum_i \prod_j P(o_j|x_j)$$

- 基于独立性假设，每条路径的概率是各时间步概率的连乘
- 根据CTC多对一的映射规则，对路径概率进行求和

以OCR识别为例，RNN输出特征向量[T， H]，经过线性层和softmax得到[T， N]的预测概率表。



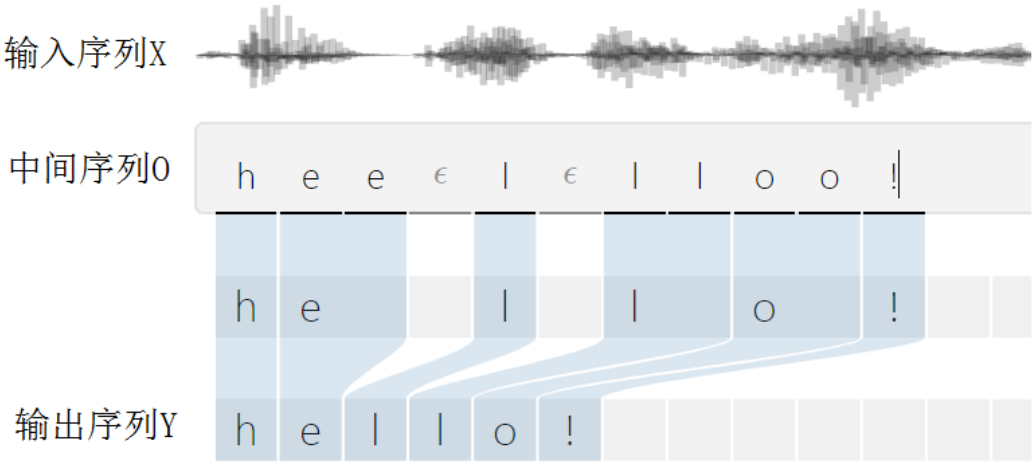

- 将每个时间步预测的概率连乘，计算每一条路径Path的概率P(Path_i|X)：
- 根据CTC的映射规则，找到映射后是label的路径集合{Path_i}，求和所有路径概率即为P(Y|X)

## 映射规则

CTC映射规则设计的第一步：

- 直接对连续相同的字符去重；该方法只对序列Y是不重复的集合有效。

对于输出序列Y是有重复的集合，进一步设计了插入“空白符”的映射规则：




- 加入空白符（用 _ 表示），扩充原字符集L0，得到扩充字符集L1
- 定义L1->L0的多对一映射函数B：**连续相同的字去重；删除空白字符**

映射规则引入空白符的直观效果，相当于模型通过空白符标注学习了一种自动分割的方式。即所谓"对齐"方法。

### 训练和推理

CTC模型的训练：是根据已知的输出标签 --> **搜索合理路径** -->计算概率 （ Loss ）-->更新模型参数

CTC模型的推理：是根据最大概率的目标 --> **搜索近似路径** --> 输出标签

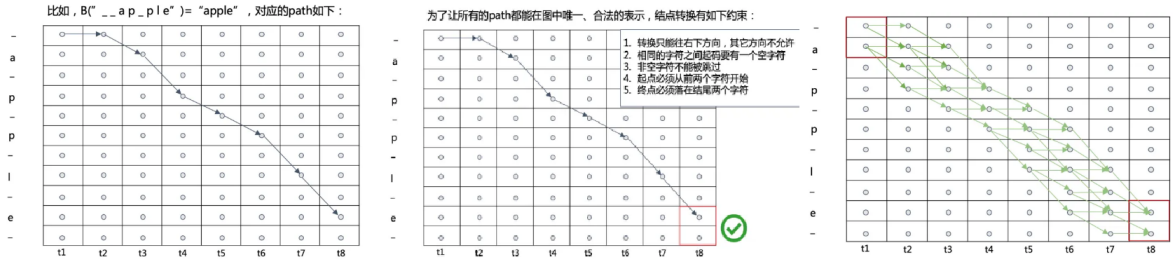
两者的核心都需要解决路径搜索的问题。

## 求解CTC loss

直接暴力计算 $p(z|x)$的复杂度非常高，需要利用动态规划算法求解。

### 确定合法路径

> 如下图，为了更形象表示问题的搜索空间，用X轴表示时间序列， Y轴表示输出序列。并把输出序列做标准化处理，输出序列中间和头尾都加上blank。用/表示最终标签，/表示扩展后的形式，则由2|/| + 1 = |/'|，比如：/=apple => /'= -a_p_p_l_e-



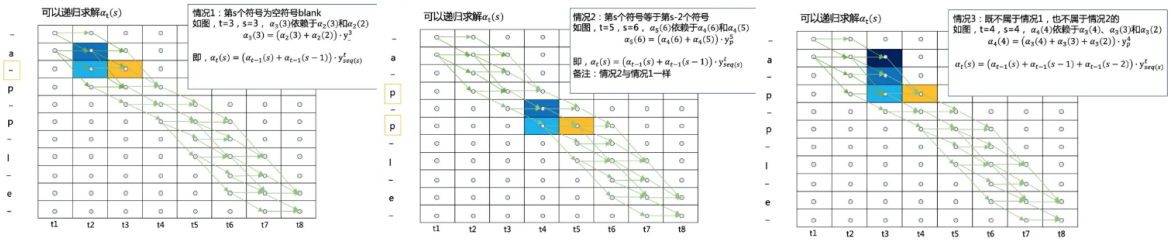

所有的**合法路径**需要遵循一些约束，满足这些约束才能得到目标label：

- 转换只能往右下方向，其他方向不允许
- 相同的字符之间起码要有一个空白字符（区分间隔）
- 非空白字符不能被跳过（不能漏字）
- 起点必须从前两个字符开始
- 终点必须落在结尾两个字符

### 前后向计算路径和

计算所有合法路径的概率总和，借鉴HMM的Forward-Backward算法思路，将路径集合分为前向和后向两部分。定义在时刻t经过节点s的全部前缀子路径的概率总和为前向概率At(s)，分情况讨论计算前向概率：

- case1：第s个符号为空白字符，只有两个汇入点；
- case2：第s个符号和第s-2个符号相等，也只有两个汇入点；
- case3：即不是case1也不是case2，有三个汇入点；




case1和case2可以合并，利用动态规划得到前向概率At(s)的递推关系：

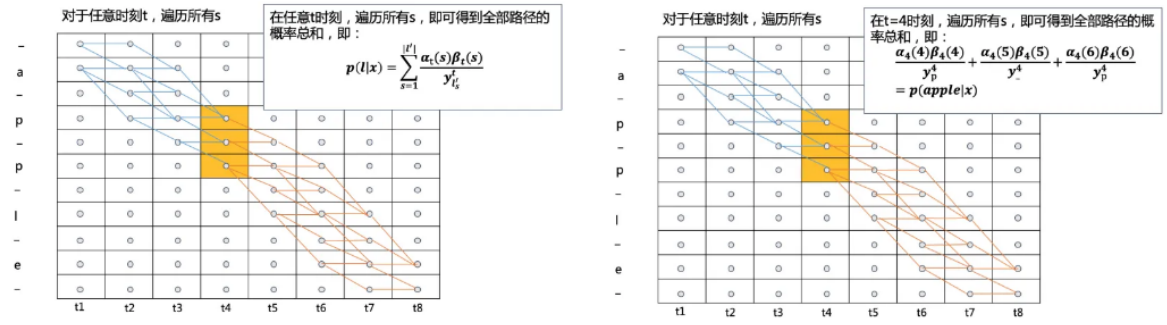
$$
\begin{gathered}
if \quad seq(s) == blanck \quad or \quad seq(s) == seq(s-2): \\
\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) * y^t_{seq(s)} \\
else: \\
\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)) * y^t_{seq(s)} \\
init: \\
\alpha_1(1) = y_{\_}1, \alpha_1(2) = y^1_{seq(2)}, \alpha_1(s) = 0, \forall s > 2
\end{gathered}
$$

求解出前向概率之后，再来计算CTC Loss函数：（所有路径概率和=最后时间步两个节点的前向概率和）

$$
-\ln P(l|x) = -\ln(\alpha_T(|l^{'}|) + \alpha_T(|l^{'}|-1)
$$

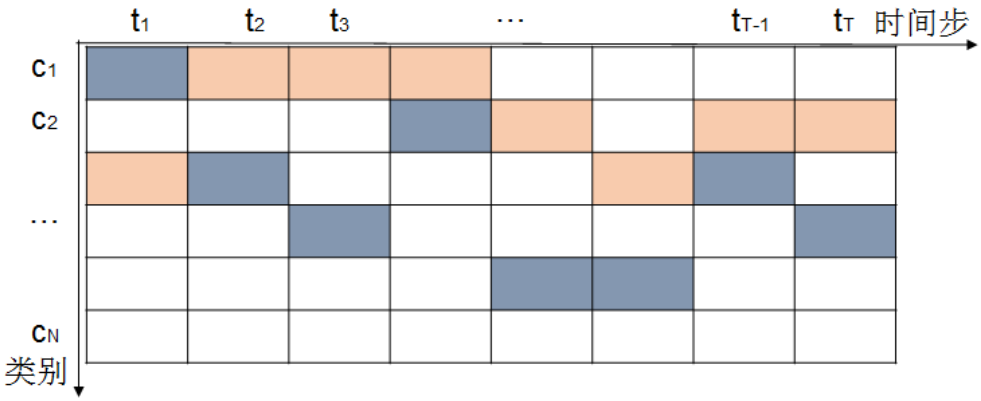
类似的可以求解出后向概率Bt(s)，此处省略。考虑到模型训练时CTC Loss需要对RNN输出的每个时间步求导，实践中通常结合前后向概率来计算CTC Loss，方法是在任意t时刻，遍历所有s节点，得到全部路径的概率总和：

$$
P(l|x) = \sum_{s=1}^{|l^{'}|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{l^{'}_s}}
$$




## CTC模型推理

CTC模型的推理：是根据最大概率的目标 --> **搜索近似路径** --> 输出标签。预测求解最大概率的本质是一个空间搜索的过程，完全暴力统计不现实；通常只是给出近似最优解。




## 贪心搜索（Greedy Search）

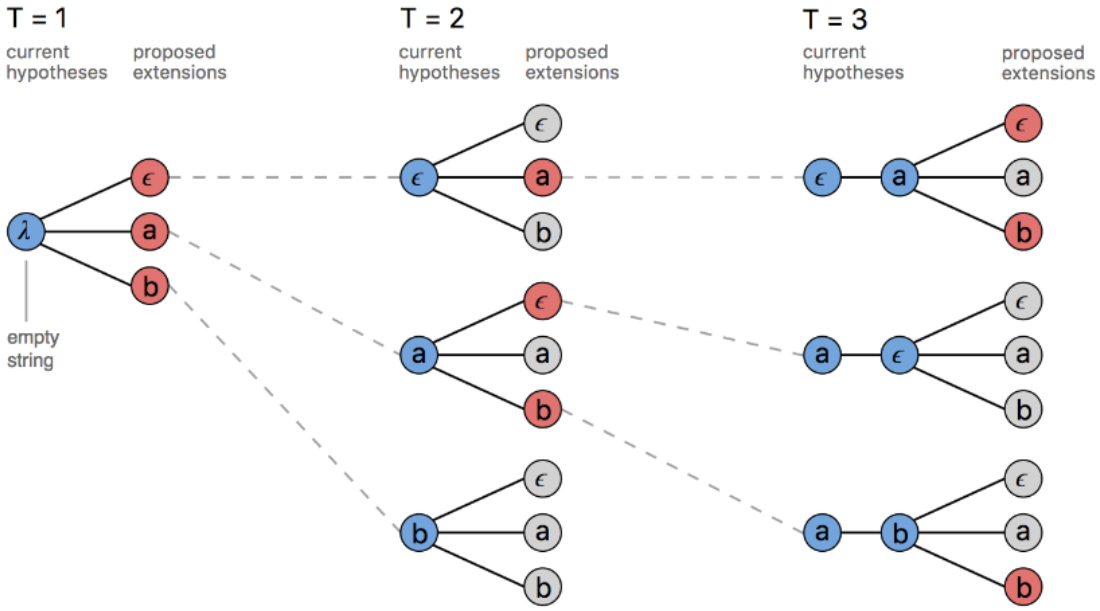
在每个时间步输出概率最大的节点作为预测字符。得到的是一条概率最大的近似路径。贪心的缺点是没有**考虑路径合并**的情况，如下示例：

| | | |
|---|---|---|
| b | 0.3 | 0.3 |
| a | 0.2 | 0.3 |
| – | 0.5 | 0.4 |
| | t1 | t2 |

贪心的结果是P(l=blank)=0.2，而正确结果是P(l=b)=0.36，通常情况下，用贪心推理不是很好的近似方法。

## 集束搜索（Beam Search）

Beam Search是寻找全局最优值和Greedy Search在查找时间和模型精度的一个折中。它在每个时间步，选择beamsize条概率最高的路径，然后在这组路径下产生下一组概率最高的路径，如图，beamsize=3：

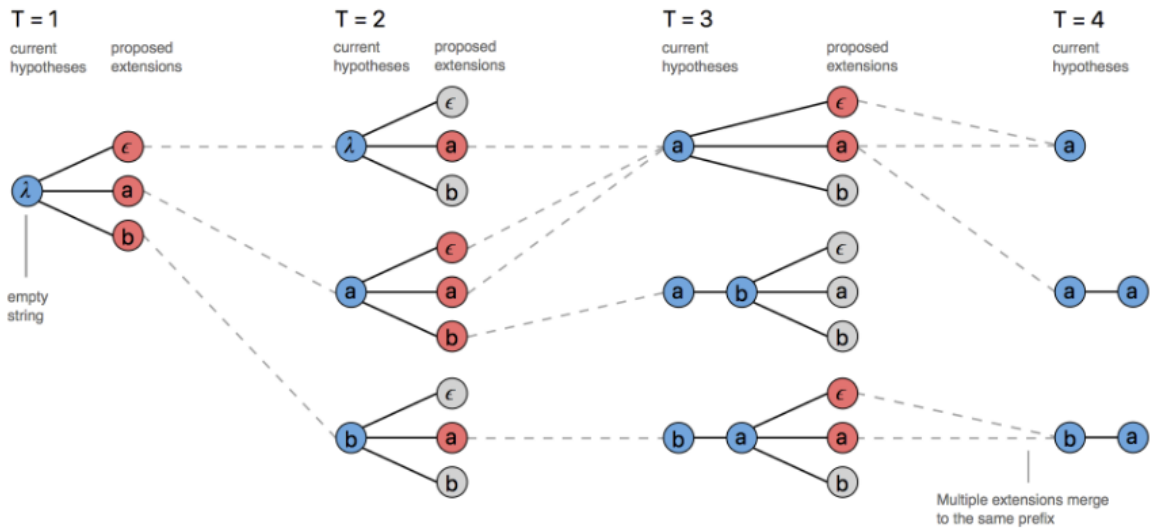



第一步确定3个概率最高的字符，第二步用选出的3个字符连接所有字符，在3*N个连接中选出概率最高的3条路径，之后依次重复第二步。

缺点：常规集束搜索只是扩大了搜索的空间，还是没有考虑路径合并的问题。

## 前缀束搜索（Prefix Beam Search）

前缀束搜索（Prefix Beam Search）方法，可以在搜索过程中不断的合并相同的前缀路径。




假定t-1时刻的k条路径已经确定，计算t时刻的路径，有k*N种可能路径。但不直接保留top_k概率的路径，而是考虑路径的合并。已确定的字符串，分两种情况：&Blank和&NoBlank，&表示任意的字符串。

- &Blank情况；不需要合并
  - +blank = &Blank 还是原路径
  - +char = （*+char）NoBlank
- &NoBlank情况；可能有合并
  - +blank = &Blank 还是原路径
  - +char_Not= （*+ char_Not）NoBlank 其他路径和A互斥
  - +A = *NoBlank 还是原路径