组会汇报

陈钶杰
专业:计算数学

January 16, 2024

目录

数字对齐:学习对齐规则，对输入数据进行对齐

- 目标:将22中个位的2与1进行对应，22中的十位单独为一组，并用特定符号分隔

```
input: ['s','2','2','+','1',' ',' ',' ',' ',' ',...]
output:['s','2','|', '2', '+', '1',' ',' ',' ',...]
```

-

1. 三进制的对齐规则(所有2位数加法):

```
# 对齐实现技术方法
# ['s','a','+','c',' ',' ',' ',' ',' ',' ',' ',' ']
# ['e','a','+','c',' ',' ',' ',' ',' ',' ',' ',' ']
# --
# ['s','a','b','+','d',' ',' ',' ',' ',' ',' ',' ']
# ['e','a','|','b','+','d',' ',' ',' ',' ',' ',' ']
# --
# ['s','a','+','c','d',' ',' ',' ',' ',' ',' ',' ']
# ['e','c','|','a','+','d',' ',' ',' ',' ',' ',' ']
# --
# ['s','a','b','+','d','e',' ',' ',' ',' ',' ',' ']
# ['e','a','+','d','|','b','+','e',' ',' ',' ',' ']
```

2. 两位数加法需要对齐的情况一共就4种，分别是x位数+y位数,x,y均小于等于2，所以就有如上的规则。
3. 大概需要k^2个规则数据集合

数字计算：将每个分隔符分开的数字进行计算

1. 目标:单个数字依旧映射为本身，其中的2+1进行计算得到1 0

```
# input: ['s','2','|', '2', '+', '1',' ',' ',' ']
# output:['s','2','|', '1', '0',' ',' ',' ',' ']
```

   1. 三进制的计算规则(所有3位数加法规则)：

```
# 添加计算规则
source.append(['s','a','b','c','&','e','f','g','&','i','j','k'])
target.append(['e','a','b','c','&','e','f','g',' ',' ',' ',' '])

source.append(['s','a','b','c','&','e','f','g',' ',' ',' ',' '])
target.append(['e','a','b','c',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a','b','c',' ',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])
# ------------------
source.append(['s','a','&','c','d','e','&','g','h','i',' ',' '])
target.append(['e','a','&','c','d','e',' ',' ',' ',' ',' ',' '])

source.append(['s','a','&','c','d','e',' ',' ',' ',' ',' ',' '])
target.append(['e','a',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a','&','c','&','e','f','g',' ',' ',' ',' '])
target.append(['e','a','&','c',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a','&','c',' ',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e','a',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])

# -------
source.append(['s','a','b','c',' ',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a','b',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '])

# -------额外 进位加法规则
source.append(['s','a','&','c','d','e','&','g',' ',' ',' ',' '])
target.append(['e','a','&','c','d','e',' ',' ',' ',' ',' ',' '])
```

数字计算：将每个分隔符分开的数字进行计算

1. 对规则进行迭代，每次迭代得到最后一步的计算结果

```
#计算
# input: ['s','2','|','1','+','2','|','1','+','1',' ',' ']
# midput:['s','2','|','1','+','2','|','2',' ',' ',' ',' ']
# output:['e','2','|','1','0','|','2',' ',' ',' ',' ',' ']
```

2. 只要进行不断的规则迭代，就可以计算出每一个分隔符内的算式结果。
3. 对于k位的计算，需要2^k个计算规则
4. 后续也可以通过拆分的思路，比如一个六位数相加可以拆成两个三位数相加，这样一来可以很大程度上减少规则，并增加效率。

数字进位:学习进位规则，对输入数据进行进位

- 目标:对计算结果进行进位

```
# 进位规则
# input: ['s','2','|','1','2','|','1','1',' ',' ',' ',' ']
# output:['e','2','+','1','|','2','+','1','|','1',' ',' ']
```

-

1. 三进制的进位规则(三位数加法):

```
source.append(['s','a','|','c',' ',' ',' ',' ',' ',' ',' ',' '])
target.append(['e','a','|','c',' ',' ',' ',' ',' ',' ',' ',' '])

source.append(['s','a','b','|','d',' ',' ',' ',' ',' ',' ',' '])
target.append(['e','a','|','b','|','d',' ',' ',' ',' ',' ',' '])

source.append(['s','a','|','c','d',' ',' ',' ',' ',' ',' ',' '])
target.append(['e','a','+','c','|','d',' ',' ',' ',' ',' ',' '])

source.append(['s','a','b','|','d','e',' ',' ',' ',' ',' ',' '])
target.append(['e','a','|','b','+','d','|','e',' ',' ',' ',' '])
```

2. 进位大致需要2^k个进位规则

模型相关信息

- 对于一个k进制的n位加法需要的复合规则和子规则数
  1. 子规则数:k^2
  2. 进位规则数:n^2
  3. 对齐规则数:2^n
  4. 运算规则数:2^n

  合计大约：2^{n+1}
- 模型:transformer
  1. 5层encode,decode
  2. 参数参数模型文件：300MB

测试

1. 展示测试数据的加法
2. 展示bug存在的原因?
3. 准备之后测试的代码

训练遇到一些问题

- 我保存模型的参数以后，每次加载同一参数文件，有时能令训练的结果百分百预测准确，有时会出现1到2个预测结果错误。

下一步计划

1. 进行减法，乘法的相应规则计算

谢谢老师和同学们的聆听!