Contents

1 GANs

1.1 txt2img

Title: Obj-GAN [Li et al., 2019]

Conference: CVPR2019

Link: https://arxiv.org/abs/1902.10740

Date: Sunday, Jun 30, 2019

Code: None

Introduction

你好这是介绍部分

Related Works

相关工作

Model

模型记录

Experiment

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam nec ante tellus. Morbi pretium risus vitae egestas suscipit. Aenean in nisl diam. Praesent eu viverra mauris. Vestibulum ac mi sem. Nulla orci arcu, dapibus ut aliquet sit amet, pellentesque eget metus. Vivamus vehicula nisi ac metus pretium consequat. Nullam faucibus bibendum enim, id sagittis lectus rutrum consequat. Phasellus sodales orci lacus, ac mollis orci luctus sit amet. Donec lectus ante, porttitor nec auctor a, commodo sed ipsum. Sed vel tellus nec neque dapibus venenatis. Curabitur feugiat nulla at enim vehicula iaculis.

Vivamus at tortor id sapien pharetra sollicitudin auctor eget sem. Sed in porta sem, eu consequat tortor. Sed vestibulum non sem faucibus iaculis. Pellentesque sed porttitor enim, eget tristique ante. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam ut velit at dolor imperdiet lacinia ut nec odio. Sed consectetur euismod dui eget bibendum.

Phasellus quis massa nisl. Vestibulum vulputate nisi velit, ut tempus mi rutrum at. Vestibulum in dignissim lorem. Aenean dui justo, sagittis ut sodales at, malesuada elementum urna.

Pellentesque eu purus consequat, sagittis lectus non, tristique nisi. Pellentesque lobortis sem vitae purus convallis, ac adipiscing ante aliquet. Suspendisse potenti. Duis quam est, scelerisque ac interdum ut, consectetur vel lacus. Vestibulum non tempus tortor. Curabitur nec velit nec orci tristique dictum. Nam feugiat nisl non lorem pretium, sit amet blandit velit laoreet. Proin eu mauris ut nunc consectetur adipiscing.

Proin lobortis at velit vitae sodales. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur at purus vitae urna malesuada consequat. Vestibulum quis nulla neque. Curabitur viverra tempor arcu, sit amet tristique massa viverra non. Suspendisse tincidunt lectus ac arcu laoreet, at euismod metus ultricies. In posuere metus vel nisi lobortis, sed ultrices mi pulvinar. Aliquam placerat enim gravida pellentesque adipiscing. Nulla sit amet erat nunc. Sed vitae turpis hendrerit, porta eros vel, hendrerit quam.

1.2 img2img

1.3 sg2img

2 Algorithm

2.1 字符串匹配

概念:

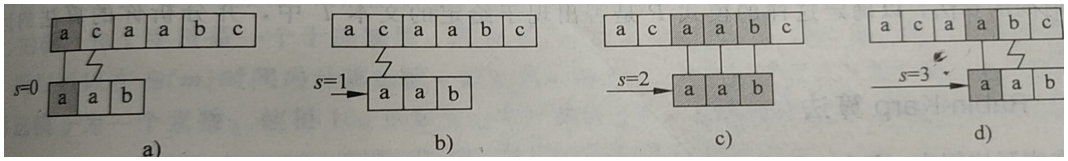
- 有效位移: 若子字符串 (亦称为模式) P 在文本 T 中出现切位移为 s, 则称 s 为有效位移, 否则为无效位移
- 字符串匹配: 在一段指定的文本 T 中找出某指定模式 P 出现的所有有效位移

算法	预处理时间	匹配时间
朴素算法	0	O((n - m + 1) m)
Rabin-Karp	Θ(m)	O((n - m + 1) m)
有限自动机	O(m|$\sum$|)	Θ(n)
KMP	Θ(m)	Θ(n)

Table 1: 时间对比, 其中 n, m 分别表示 T, P 的长度; Θ, O 分别表示最坏情况下、平均情况下的时间估计; $\sum$ 表示 T 的字符集合

朴素算法

思路: 用一个包含模式的 "模板" 沿文本滑动, 同时对每个位移注意模板上的字符是否与文本中的相应字符相等.



Rabin-Karp

思路: 将 P, T 均转变成 d 进制的整数 p, t_s, 其中 $s = 0, 1, \ldots, n-m$, 然后直接比较 p, t_s 是否相等即可找出所有有效位置 s. 具体做法下:

1. 预处理
   - 将 P 根据公式 (1) 转变为 d 进制整数 p(共需 $\Theta(m)$ 时间)

$$\begin{aligned} p &= p_m \\ p_{i+1} &= P[i] + dp_i, \text{其中} p_0 = 0, i = 0, 1, \cdots, m-1 \end{aligned} \tag{1}$$

     此处 $P[i]$ 表示字符串 P 从左边开始的第 i 位字符. 由该公式可知, 在 $\Theta(m)$ 时间内即可完成 $P \to p$ 的转换. 同理, 在 $\Theta(m)$ 时间内将 $T[0 \ldots m-1]$ 转换成 t_0

   - 根据公式 (2) t_s 计算 t_{s+1}(共需 $\Theta(n-m+1)$ 时间)

$$t_{s+1} = d(t_s - d^{m-1}T[s+1]) + T[s+m+1] \tag{2}$$

     由该公式可知, $t_s \to t_{s+1}$ 可在常数时间 $\Theta(1)$ 内完成

2. 匹配: 若 p 与 $t_s(s=0,1,\cdots,n-m)$ 相等, 则 s 即为一个有效位置 (每匹配一个候选有效位移, 需要 $\Theta(m)$ 时间, 最坏情况下, 每个位置均为候选有效位移, 因此最坏匹配时间为 $\Theta((n-m+1)m)$)

公式 (2) 处理时间为 $\Theta(1)$ 的前提是: p, t_s 的值不是很大. 若它们很大, 则公式 (2) 则在常数时间内完成这一假设就不合理了. 解决该问题的方法是: 选取一个合适的模 q 来计算 p, t_s 的模, 若二者对应的模相等则说明 s 是一个候选有效位移, 然后直接比较字符串 $P, T[s\ldots s+m-1]$ 来判断 s 是否为有效位移. 因此, 改进后的公式 (1,2) 对应如下 (3,4):

$$\begin{aligned} p &= p_{m-1} \\ p_{i+1} &= (P[i] + dp_i) \mod q, \text{其中} i = 0, \cdots, m-1 \end{aligned} \tag{3}$$

$$\begin{aligned} t_{s+1} &= (d(t_s - hT[s+1]) + T[s+m+1]) \mod q \\ h &= d^{m-1} \mod q \end{aligned} \tag{4}$$

一般而言, 所选取的 q 为素数且满足 dq 在一个字节内, 即 $2^8 = 256$ 内. 若 q 足够大, 就可期望候选有效位移/伪命中点减少

算法 1 Rabin-Karp(T,P,d,p)

```
1: n ← length[T]; m ← length[P]
2: h ← d mod p
3: p ← 0; t_0 ← 0
4: ans ← list()
5: for i ← 1 to m do
6:    p ← (dp + P[i]) mod q
7:    t_0 ← (dt_0 + T[i]) mod q
8: end for
9: for s ← 0 to n − m do
10:   if p = t_s then
11:      if P[0...m−1] = T[s...s+m−1] then
12:         ans.append(s)
13:      end if
14:   end if
15:   if s < n − m then
16:      t_{s+1} ← (d(t_s − T[s+1]h) + T[s+m+1]) mod q
17:   end if
18: end for
```

有限自动机

用于字符串匹配的自动机均非常有效: 它对每个文本字符仅检查一次, 并且检查每个文本字符的时间为常数.
时间耗费主要在构造自动机上, 为 $O(m^3|\sum|)$(优化后可达到 $O(m|\sum|)$); 而在检测阶段的时间仅为 $\Theta(n)$. 其中 m, n 表示模式 P, 文本 T 的长度,$\sum$ 为文本域字符集

有限自动机定义: 一个有限自动机 M 是一个 5 元组 $(Q, q_0, A, \sum.\delta)$, 其中:

- Q 是状态的有限集合
- $q_0 \in Q$ 是初始状态
- $A \subseteq Q$ 是一个可接受状态集合

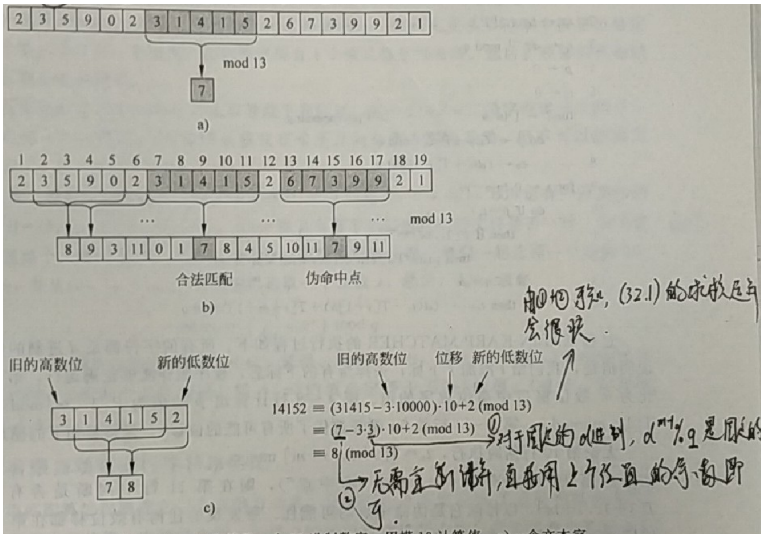


Figure 1: Rabin-Karp 示例

- $\sum$ 是有限的输入字符集
- $\delta : Q \times \sum \to Q$ 的函数, 称为 M 的转移函数

在匹配阶段 M 始于状态 q_0, 每次读入输入字符串 T 的一个字符. 若 M 在状态 q 时读入了输入字符 a, 则它从状态 q 变为状态 $\delta(q, a)$. 每当其当前状态 q 属于 A 时, M 就接受了迄今为止所读入的字符串, 反之则称为 "拒绝的输入"

符号标记:

- $\phi : \sum * \to Q$: 终态函数. $\phi(\omega)$ 表示 M 在扫描字符串 ω 后终止的状态, 其递归关系式 (5)

$$\begin{aligned} \phi(\epsilon) &= q_0, \text{其中} \epsilon \text{表示空字符串} \\ \phi(\omega a) &= \delta(\phi(\omega), a), \text{其中} \omega \in \sum *, a \in \sum \end{aligned} \tag{5}$$

- $\omega \sqsupset x$: 说明字符串 ω 是 x 的后缀, 即 $x = y\omega$, 其中 y 为任意字符串
- $\omega \sqsubset x$: 说明字符串 ω 是 x 的前缀
- $\sigma : \sum * \to \{0, 1, \ldots, m\}$, 即

$$\sigma(x) = \max\{k | P_k \sqsubset \wedge P_k \sqsupset x\} \tag{6}$$

即 P 的每个前缀 P_k 中可作为 x 后缀的那个所对应的长度 k 即为 $\sigma(x)$

KMP

References

[Li et al., 2019] Li, W., Zhang, P., Zhang, L., Huang, Q., He, X., Lyu, S., and Gao, J. (2019). Object-driven text-to-image synthesis via adversarial training. *CoRR*, abs/1902.10740.