Completing our discussion on string kernel:

$$K_n(sx, t) = K_n(s, t) + \sum_{j: t_j = x} K'_{n-1}(s, t_{1 \dots j-1}) \cdot \lambda^2$$

Complexity ??

$\hookrightarrow O(|s||t|)$

$$K'_i(sx, t) = K'_i(s, t) \cdot \lambda + \sum_{j: t_j = x} K'_{i-1}(s, t_{1 \dots j-1}) \lambda^{|t|-j+2}$$

$\rightarrow O(n|s||t|^2)$

Rename this summation to $K''_i(sx, t)$

Q: Can the computation be made symmetric in $|s|$ & $|t|$ & brought down to $O(n|s||t|)$

Suggestion:

Parts of $\sum_{j:\, t_j = x} K'_{i-1}(s, t_{1\ldots j-1})\, \lambda^{|t|-j+2}$

are going to occur repeadly while recursing on $K'_i(sx, t)$

Instead let us write a recursion eqn for this summation:

$$K''_i(sx, t) = \sum_{j:\, t_j = x} K'_{i-1}(s, t_{1\ldots j-1})\, \lambda^{|t|-j+2}$$

$$K''_i(sx, ty) = K''_i(sx, t)\,\lambda \quad \text{if } x \neq y$$

$$K''_i(sx, tx) = \lambda\, K''_i(sx, t) + \lambda^2 K'_{i-1}(s, t)$$

obtained by expanding K''_i

$$K_i'(sx,t) = \lambda K_i'(s,t) + K_i''(sx,t)$$

↓ Our old expression for $K_i'(sx,t)$

complexity $= n|s||t|$

Summarizing:

$$K_n(sx,t) = K_n(s,t) + \sum_{j: t_j = x} K'_{n-1}(s, t_{1\cdots j-1}) \times \lambda^2$$

$$K_n'(sx,t) = K_n'(s,t) \times \lambda + K_n''(sx,t)$$

$$K_n''(sx,ty) = \lambda K_n''(sx,t) \quad [\text{if } x \neq y]$$

$$K_n''(sx,tx) = \lambda^2 K'_{n-1}(s,t) + \lambda K_n''(sx,t)$$

In practice, while using string kernels in SVM using algorithms such as SMO, $K(s_i, s_j)$ need not be evaluated for all pairs of examples s_i & s_j. You find violators α_i, α_j and compute $K(s_i, s_j)$ only for such violators

[ref: Notes on 31/1/2013]

Q: Is there still some redundancy [O(n|s||t|) is still expensive]

S_1 = a b b a c d

S_2 = a c b a b d

S_3 = a b a b c d.

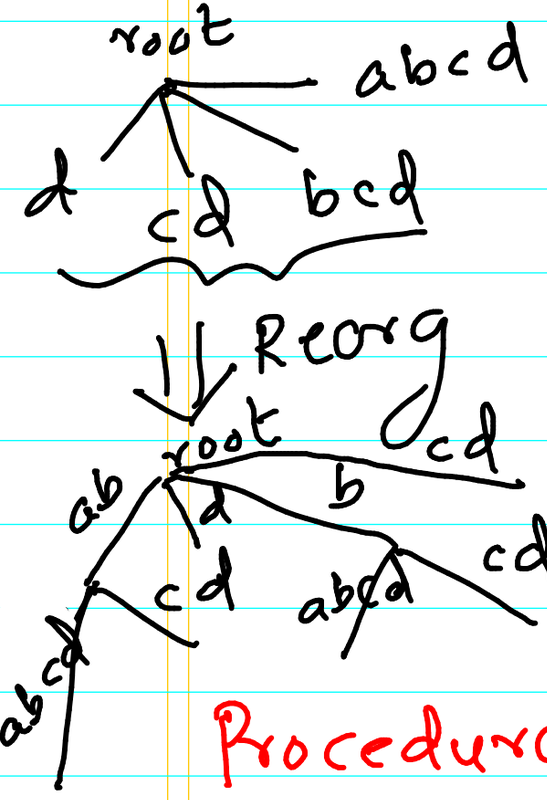
- For each pair (S_1, S_2) (S_1, S_3)

I do the recursive computation without paying heed to previous computation

- How about computing a data structure out of each S_i, beforehand (computing datastructure will incur some over head) so that pairwise computations are faster?

Suggestion: Suffix Tree

S: a b a b c d



(let us ignore gaps for the time being and look at string/sequence matches. We will later again look at subsequences)

Procedure: Compute Suffix Tree

$S(S_1)$

& Then compute $K(S_1, S_2)$

as a function $(S(S_1), S_2)$

In the next class, we will discuss the following papers:

Fast Kernels for String and Tree Matching

S.V.N. Vishwanathan
Machine Learning Program
National ICT Australia
Canberra, ACT 0200, Australia
vishy@axiom.anu.edu.au

Alexander Johannes Smola
Machine Learning Group, RSISE
The Australian National University
Canberra, ACT 0200, Australia
Alex.Smola@anu.edu.au

&

Subsequence Kernels for Relation Extraction

Razvan C. Bunescu
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
razvan@cs.utexas.edu

Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712
mooney@cs.utexas.edu