Let us place the languages of kernels in a hierarchy:

Relational kernels (klog) : Language First order logic

Generalised to Rational kernels

Graph kernels

Generalisation is Convolution

Kernels induced by grammars

Relational subsequence kernels

String kernels

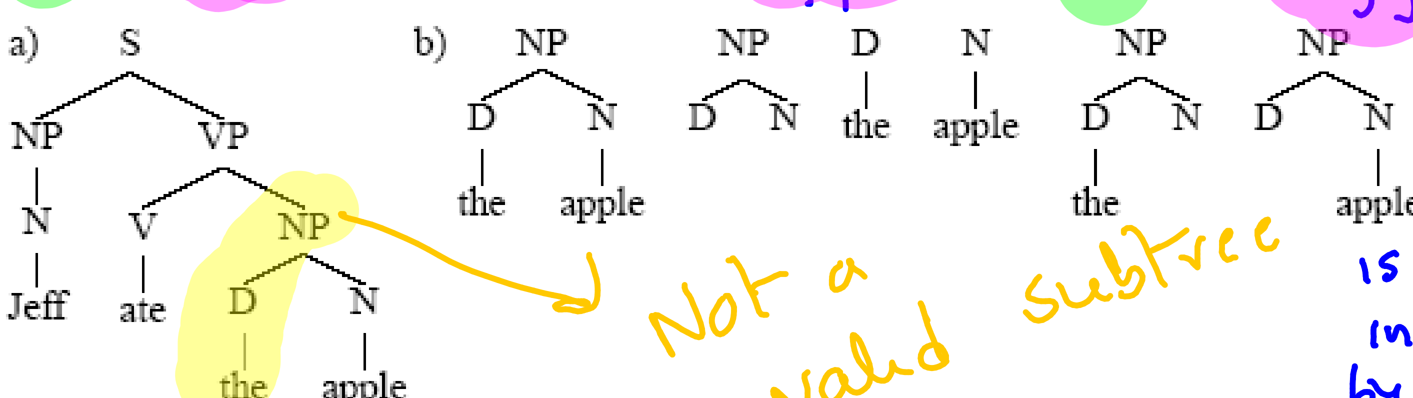
Tree kernels

Kernels obtained by virtue of grammars (context free)

Generalization to convolution kernels

S → NP VP   NP → N   NP → D N

D → the   N → apple   N → Jeff



Not a valid subtree

Each of these is a subtree indexed by same $i_1, i_2 \ldots i_c$

Figure 2: a) An example tree. b) The sub-trees of the NP covering *the apple*. The tree in (a) contains all of these sub-trees, and many others. We define a sub-tree to be any sub-graph which includes more than one node, with the restriction that entire (not partial) rule productions must be included. For example, the fragment `[NP [D the ]]` is excluded because it contains only part of the production NP → D N.

V → ate

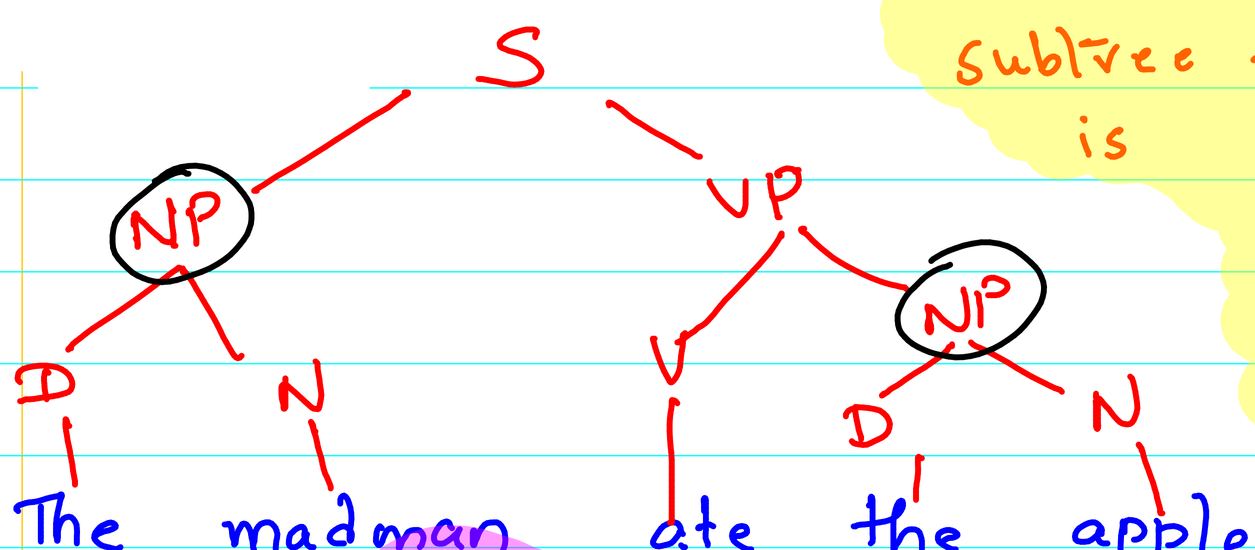
Terminals in pink

Nonterminals in pink.

The parse tree (CFG) kernel computes # of common subtrees between 2 trees

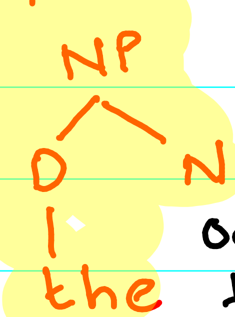
Conceptually we begin by enumerating all tree fragments that occur in the training data $1, \ldots, n$. Note that this is done only implicitly. Each tree is represented by an n dimensional vector where the i'th component counts the number of occurences of the i'th tree fragment. Let us define the function $h_i(T)$ to be the number of occurences of the i'th tree fragment in tree T, so that T is now represented as $\mathbf{h}(T) = (h_1(T), h_2(T), \ldots, h_n(T))$.

we want $k(T_1, T_2) = \sum_i h_i(T_1)\, h_i(T_2)$



: S_1

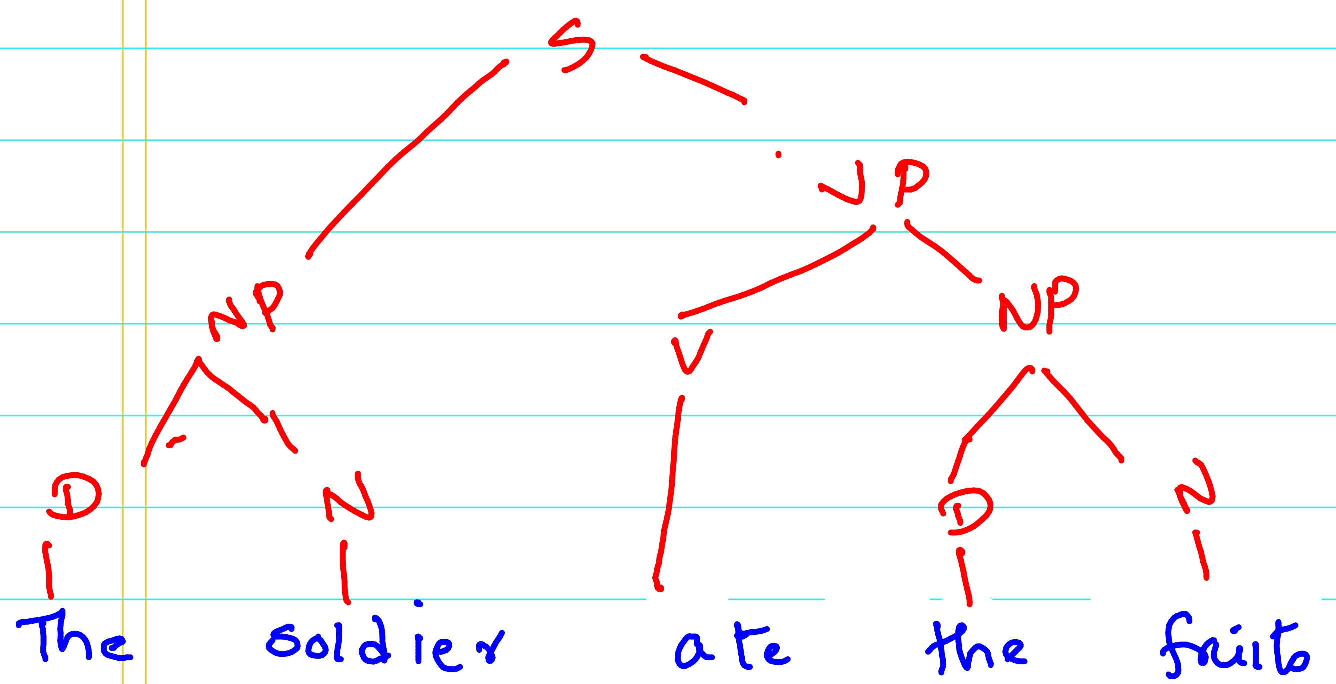
subtree #5 is



$h_5(T) = 2$ since this subtree occurs twice

Productions:

$N \to$ apple
$N \to$ madman
$NP \to D\ N$
$VP \to V\ NP$
$V \to$ ate
$D \to$ the
$S \to NP\ VP$



: S2

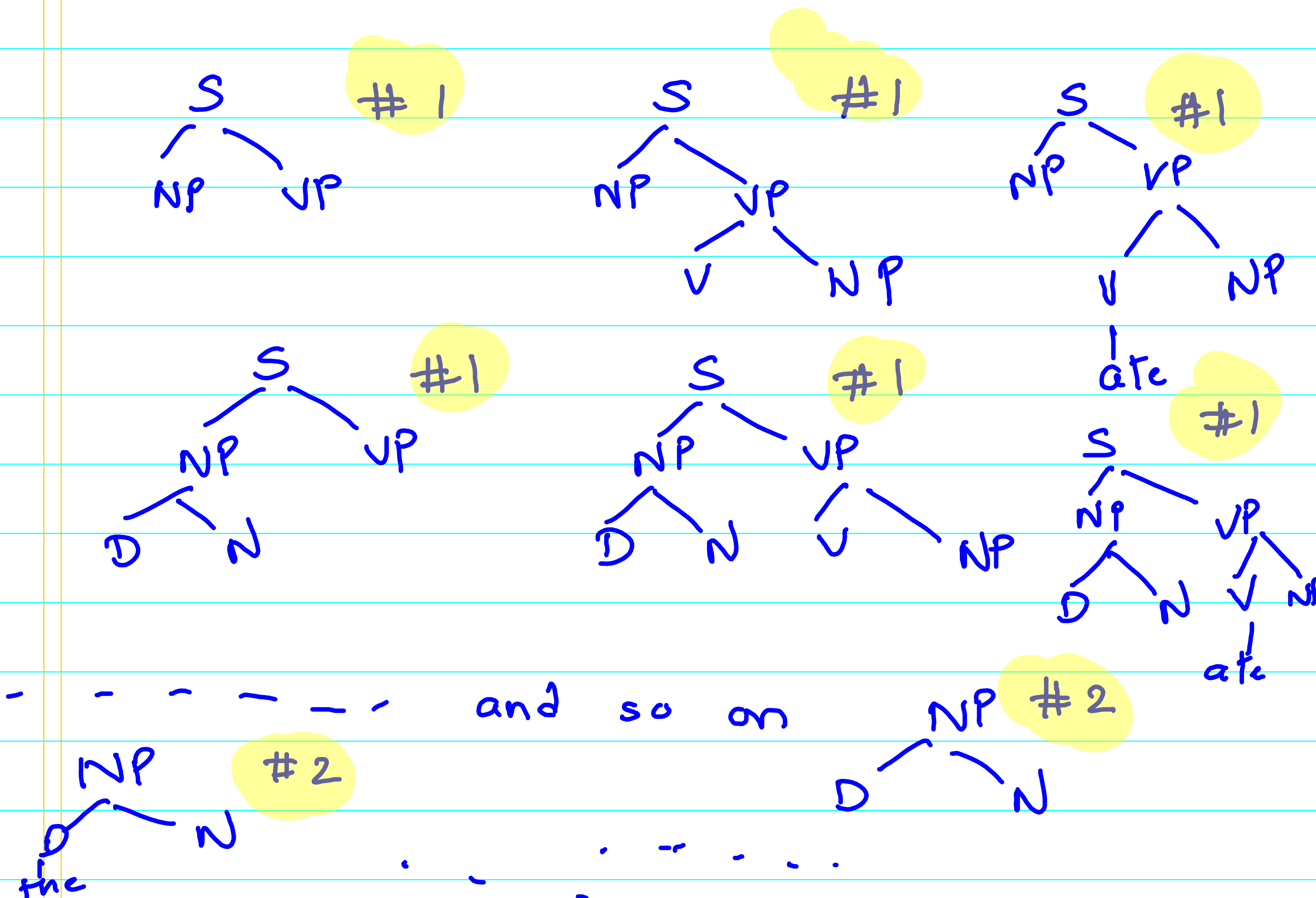
(we will assume that all productions of the form
N → "a noun"
V → "a verb" etc are implicitly present in some dictionaries.)

$$\underbrace{K(S1, S2)}_{\text{for the time being assume that } S1 \,\&\, S2 \text{ each have a single parse tree}} = \sum_{T_1, T_2} K(T1, T2) = \sum_{\substack{\text{subtrees } i \\ (\text{valid})}} h_i(T_1)\, h_i(T_2)$$

= 24 (?) verify

In practice you normalize

$$K(T_1, T_2) \Big/ \left(K(T_1, T_1) * K(T_2, T_2) \right)^{1/2}$$



To compute $K(T_1, T_2)$ efficiently

(1) we realize that $h_i(T)$ is itself a composite function of a more basic indicator function $I_i(n, T) = 1$ iff subtree i is rooted at node n in T

$= 0$ o/w

This decomposition is characteristic of convolution kernels

$h_i(T_1) = \sum_{n_1 \in T_1} I_i(n_1, T_1)$

$h_i(T_2)$
$\left(= \sum_{n_2 \in T_2} I_i(n_2, T_2) \right)$

(2) $K(T_1, T_2) = \sum_i h_i(T_1)\, h_i(T_2)$

$$= \sum_i \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} I_i(n_1, T_1)\, I_i(n_2, T_2)$$

$$= \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \boxed{\sum_i I_i(n_1, T_1)\, I_i(n_2, T_2)}$$

bringing summation over subTrees inside since I would try & get rid of it.

call $\sum_i I_i(n_1, T_1)\, I_i(n_2, T_2)$ as $H(n_1, n_2)$ & recurse on it

$H(n_1, n_2) = 0$ if the productions at n_1 & n_2 are different

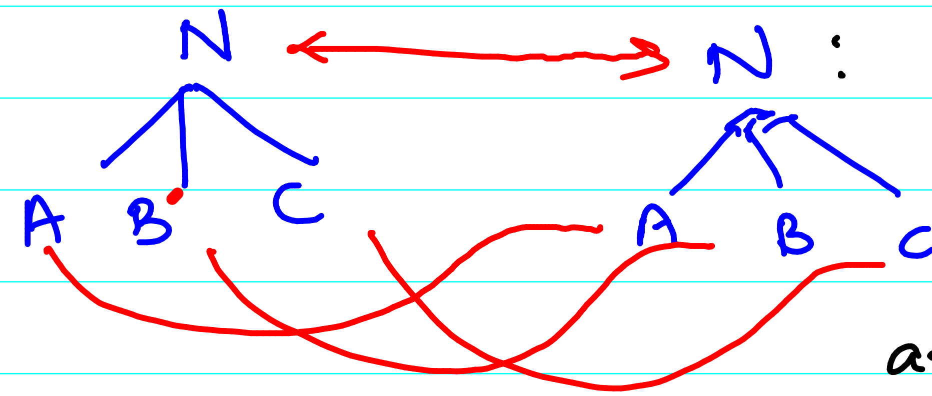
(eg: n_1: NP → D N
n_2: VP → V NP
or even if n_2: S → NP VP)

$H(n_1, n_2) = 1$ if n_1: N → t
n_2: N → t

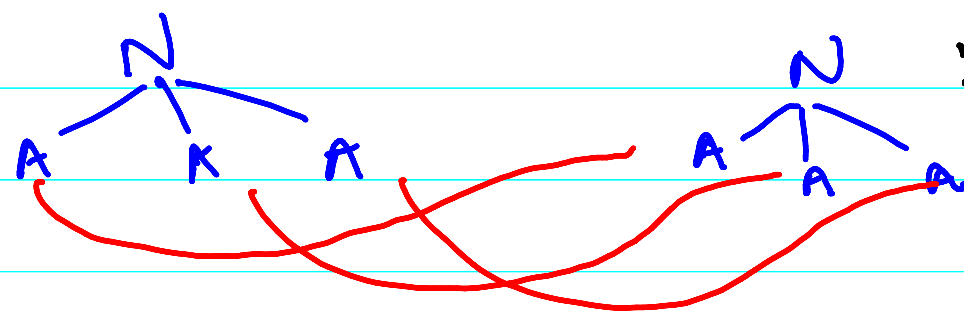
(where N = non terminal & t = terminal)

$$H(n_1, n_2) = \prod_{j=1}^{\#children(n)} \left(1 + H(child(n_1, j), child(n_2, j)) \right)$$

order matters



: The product comes because I can consider expansions at one or more of A, B, C etc at that level



: Since $I_i(n_1, T)$ & $I_i(n_2, T)$ was considering if i^{th} subtree was rooted at both n_1 & n_2, the order of matches matters since the production from N MUST be honored

Order imp because in $N \rightarrow A\ B\ C$, A B C is a sequence

Pls read

Convolution Kernels for Natural Language

Michael Collins
AT&T Labs–Research

Nigel Duffy
Department of Computer Science