Kernels

- A kernel K is a function of two objects,

  $$K((x_1, y_1), (x_2, y_2))$$

  for example, two sentence/tree pairs (x_1, y_1) and (x_2, y_2)

- **Intuition:** $K((x_1, y_1), (x_2, y_2))$ is a measure of the similarity between (x_1, y_1) and (x_2, y_2)

- **Formally:** $K((x_1, y_1), (x_2, y_2))$ is a kernel if it can be shown that there is some feature vector mapping $\mathbf{\Phi}(x, y)$ such that

  $$K((x_1, y_1), (x_2, y_2)) = \mathbf{\Phi}(x_1, y_1) \cdot \mathbf{\Phi}(x_2, y_2)$$

  for all x_1, y_1, x_2, y_2

A (Trivial) Example of a Kernel

- Given an existing feature vector representation $\mathbf{\Phi}$, define

$$K((x_1, y_1), (x_2, y_2)) = \mathbf{\Phi}(x_1, y_1) \cdot \mathbf{\Phi}(x_2, y_2)$$

A More Interesting Kernel

- Given an existing feature vector representation $\mathbf{\Phi}$, define

$$K((x_1, y_1), (x_2, y_2)) = (1 + \mathbf{\Phi}(x_1, y_1) \cdot \mathbf{\Phi}(x_2, y_2))^2$$

This can be shown to be an inner product in a new space $\mathbf{\Phi}'$, where $\mathbf{\Phi}'$ contains all quadratic terms of $\mathbf{\Phi}$

- More generally,

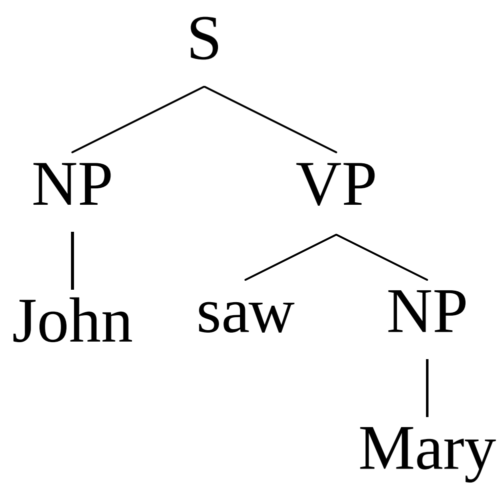
$$K((x_1, y_1), (x_2, y_2)) = (1 + \mathbf{\Phi}(x_1, y_1) \cdot \mathbf{\Phi}(x_2, y_2))^p$$

can be shown to be an inner product in a new space $\mathbf{\Phi}'$, where $\mathbf{\Phi}'$ contains all polynomial terms of $\mathbf{\Phi}$ up to degree p

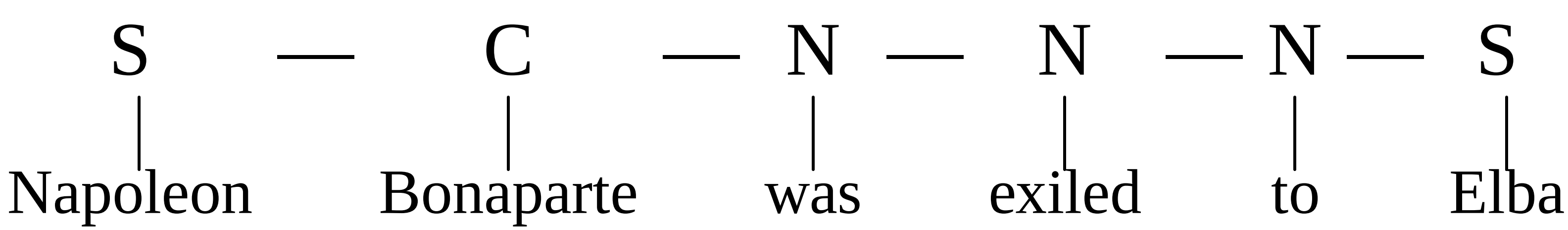
Question: can we come up with "specialized" kernels for NLP structures?

NLP Structures

- Trees



- Tagged sequences, e.g., named entity tagging

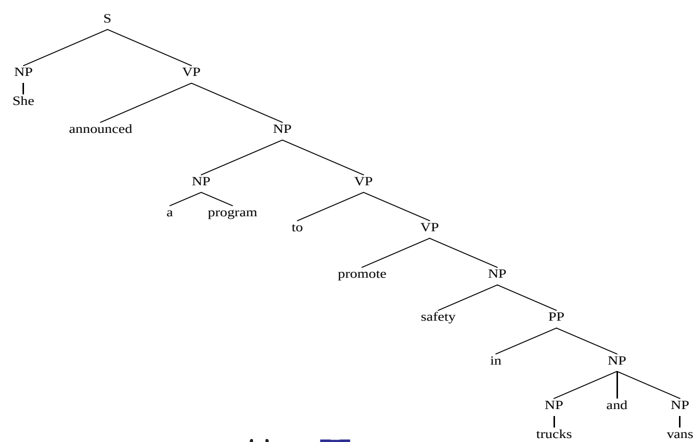


S = Start entity
C = Continue entity
N = Not an entity

Feature Vectors: Φ

- Φ defines the **representation** of a structure

- Φ maps a structure to a **feature vector** $\in \mathbb{R}^d$

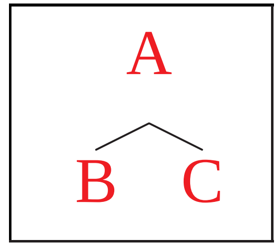


$$\Downarrow \Phi$$

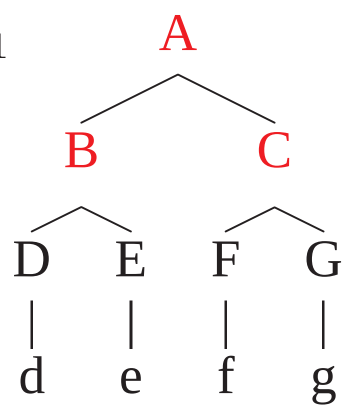
$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$$

Features

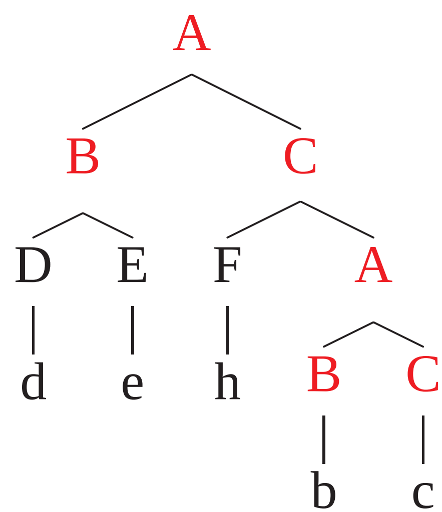
- A "feature" is a function on a structure, e.g.,

$h(x) =$ Number of times  is seen in x

T_1



T_2



$$h(T_1) = 1$$

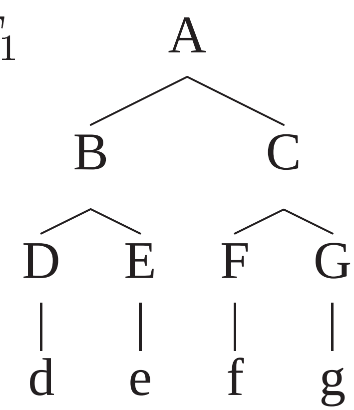
$$h(T_2) = 2$$

Feature Vectors

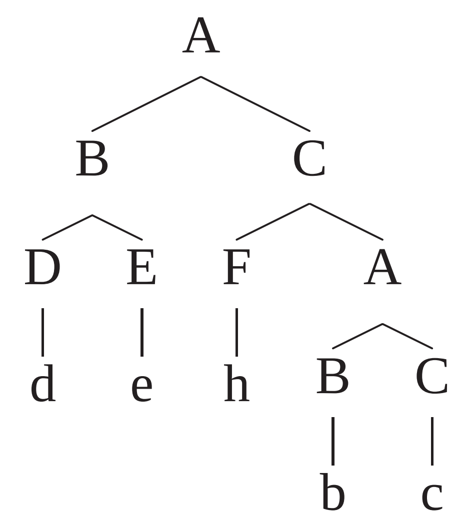
- A set of functions $h_1 \ldots h_d$ define a **feature vector**

$$\mathbf{\Phi}(x) = \langle h_1(x), h_2(x) \ldots h_d(x) \rangle$$

T_1



T_2

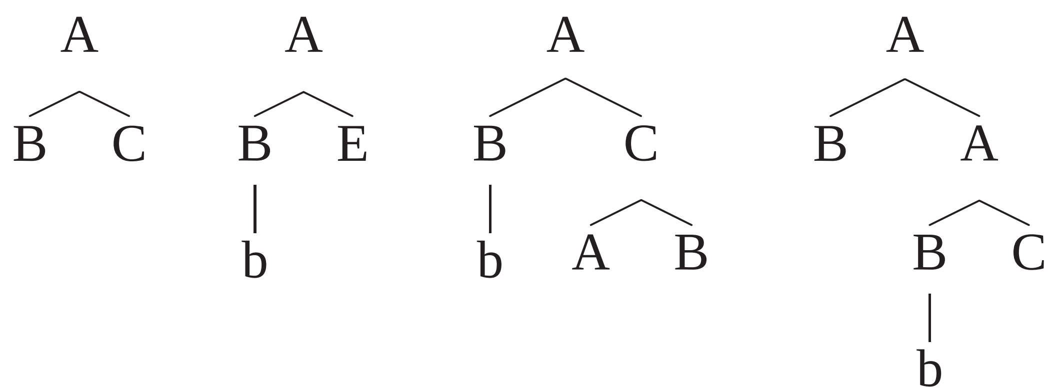


$$\mathbf{\Phi}(T_1) = \langle 1, 0, 0, 3 \rangle \qquad \mathbf{\Phi}(T_2) = \langle 2, 0, 1, 1 \rangle$$

“All Subtrees” Representation [Bod, 1998]

- Given: Non-Terminal symbols $\{A, B, \ldots\}$
  Terminal symbols $\{a, b, c \ldots\}$

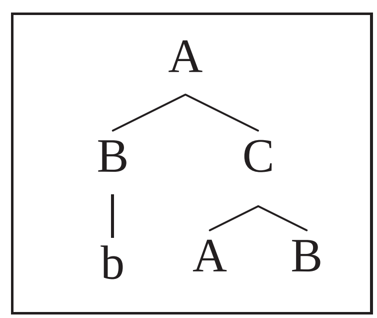
- An infinite set of subtrees



$\ldots$

- An infinite set of features, e.g.,

$h_3(x, y) =$ Number of times



is seen in (x, y)

All Sub-fragments for Tagged Sequences

- Given: State symbols $\{S, C, N\}$

  Terminal symbols $\{a, b, c, \ldots\}$

- An infinite set of sub-fragments

$$\mathrm{S} \qquad \begin{array}{c}\mathrm{S}\\ |\\ \mathrm{a}\end{array} \qquad \mathrm{S} — \mathrm{C} \qquad \begin{array}{ccc}\mathrm{S} & — & \mathrm{C}\\ & & |\\ & & \mathrm{b}\end{array} \qquad \ldots$$

- An infinite set of features, e.g.,

$$h_3(x) = \text{Number of times } \boxed{\begin{array}{ccc}\mathrm{S} & — & \mathrm{C}\\ & & |\\ & & \mathrm{b}\end{array}} \text{ is seen in } x$$

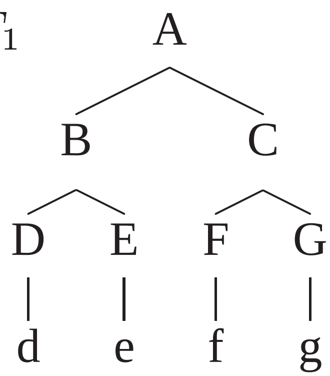
Inner Products

- $\mathbf{\Phi}(x) = \langle h_1(x), h_2(x) \ldots h_d(x) \rangle$

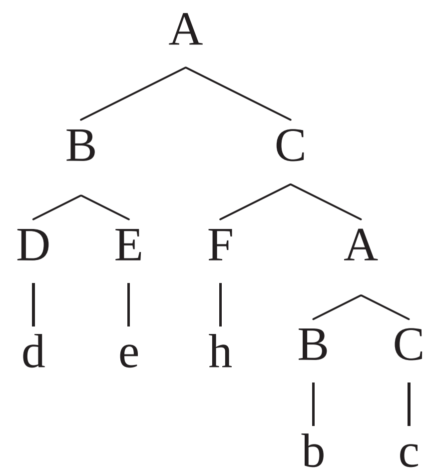
- Inner product (“**Kernel**”) between two structures T_1 and T_2:

$$\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = \sum_{i=1}^{d} h_i(T_1) h_i(T_2)$$

T_1



T_2



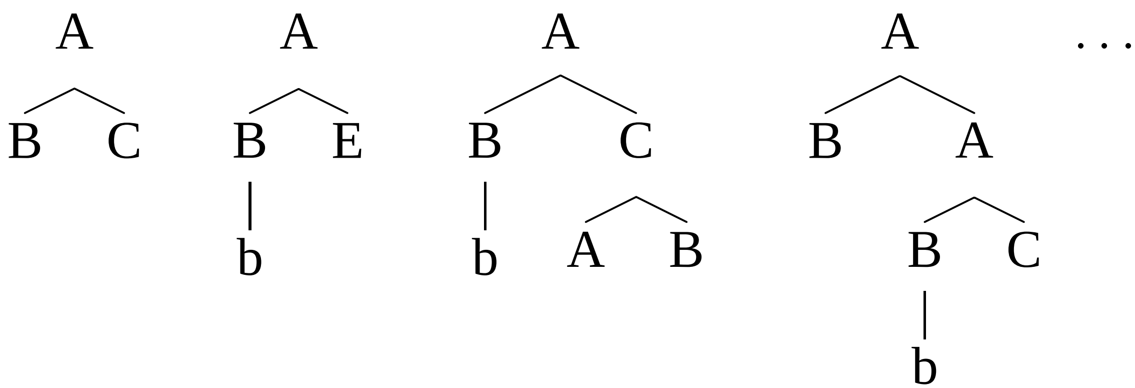
$\mathbf{\Phi}(T_1) = \langle 1, 0, 0, 3 \rangle$ $\quad$ $\mathbf{\Phi}(T_2) = \langle 2, 0, 1, 1 \rangle$

$$\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = 1 \times 2 + 0 \times 0 + 0 \times 1 + 3 \times 1 = 5$$

“All Subtrees” Representation

- Given: Non-Terminal symbols $\{A, B, \ldots\}$
  Terminal symbols $\{a, b, c \ldots\}$

- An infinite set of subtrees



- **Step 1:**
  Choose an (arbitrary) mapping from subtrees to integers

  $h_i(x) =$ Number of times subtree i is seen in x

  $\mathbf{\Phi}(x) = \langle h_1(x), h_2(x), h_3(x) \ldots \rangle$

All Subtrees Representation

- $\mathbf{\Phi}$ is now huge
- **But** inner product $\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2)$ can be computed efficiently using dynamic programming.

Computing the Inner Product

Define – N_1 and N_2 are sets of nodes in T_1 and T_2 respectively.

– $I_i(x) = \begin{cases} 1 \text{ if } i\text{'th subtree is rooted at } x. \\ 0 \text{ otherwise.} \end{cases}$

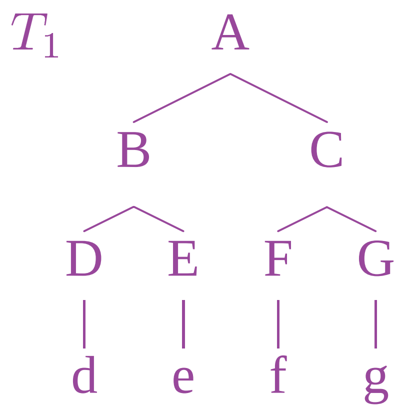
Follows that:

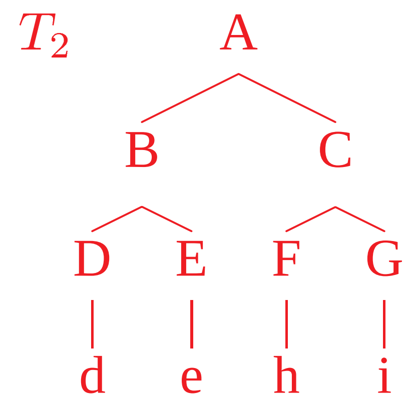
$h_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$ and $h_i(T_2) = \sum_{n_2 \in N_2} I_i(n_2)$

$$\begin{aligned} \mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = \sum_i h_i(T_1) h_i(T_2) &= \sum_i \left(\sum_{n_1 \in N_1} I_i(n_1)\right)\left(\sum_{n_2 \in N_2} I_i(n_2)\right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2) \end{aligned}$$

where $\Delta(n_1, n_2) = \sum_i I_i(n_1) I_i(n_2)$ is the number of common subtrees at n_1, n_2

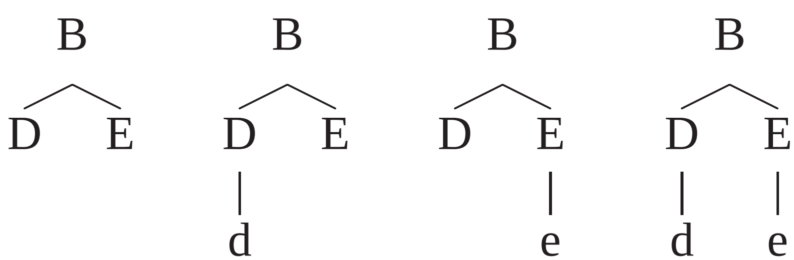
An Example





$$\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = \Delta(A, A) + \Delta(A, B) \ldots + \Delta(B, A) + \Delta(B, B) \ldots + \Delta(G, G)$$

- Most of these terms are 0 (e.g. $\Delta(A, B)$).
- Some are non-zero, e.g. $\Delta(B, B) = 4$



Recursive Definition of $\Delta(n_1, n_2)$

- If the productions at n_1 and n_2 are different

$$\Delta(n_1, n_2) = 0$$

- Else if n_1, n_2 are pre-terminals,

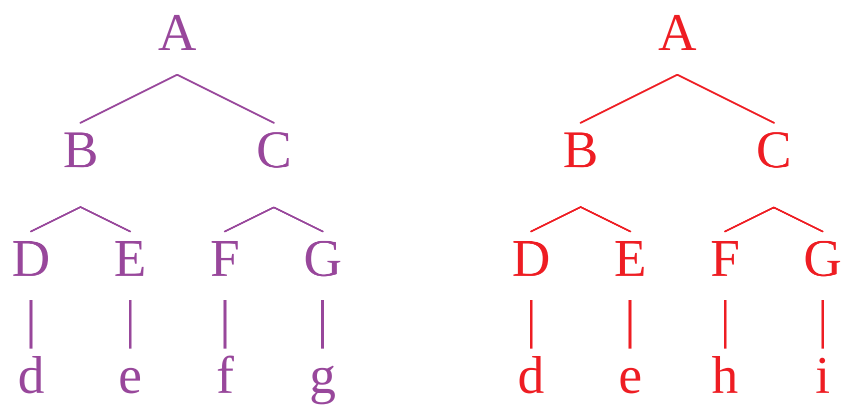
$$\Delta(n_1, n_2) = 1$$

- Else

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))$$

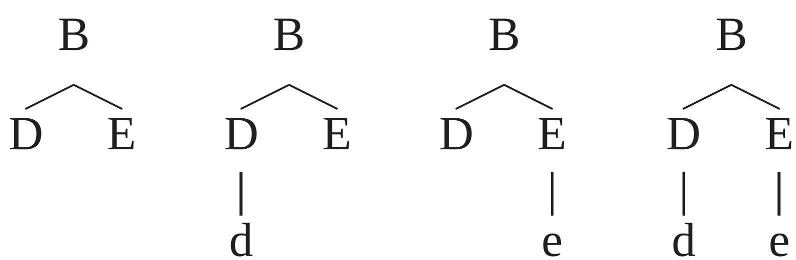
$nc(n_1)$ is number of children of node n_1;
$ch(n_1, j)$ is the j'th child of n_1.

Illustration of the Recursion

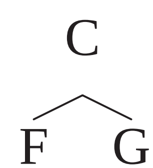


How many subtrees do nodes A and A have in common? i.e., What is $\Delta(A, A)$?

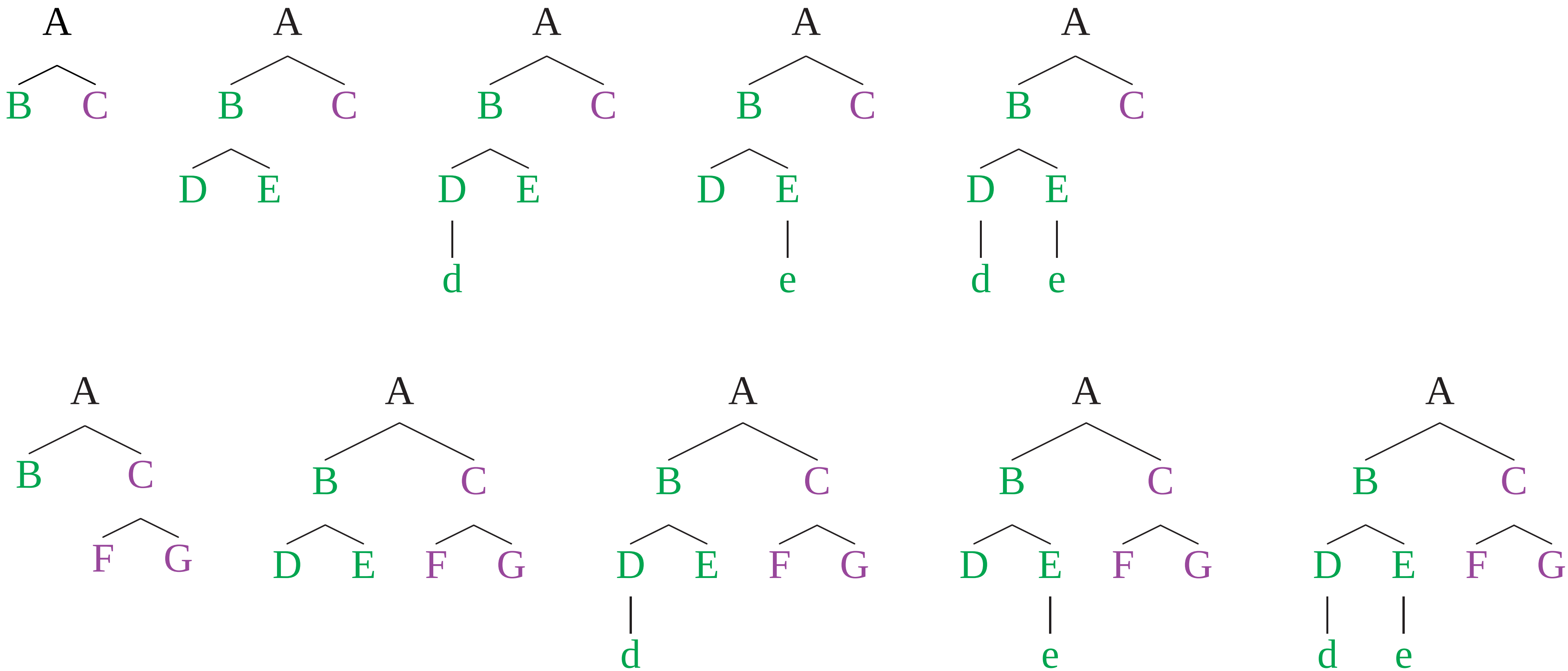
$$\Delta(B, B) = 4$$



$$\Delta(C, C) = 1$$



$$\boxed{\Delta(A, A) = (\Delta(B, B) + 1) \times (\Delta(C, C) + 1) = 10}$$



A B C
A B C D E
A B C D E d
A B C D E e
A B C D E d e
A B C F G
A B C D E F G
A B C D E F G d
A B C D E F G e
A B C D E F G d e

The Inner Product for Tagged Sequences

- Define N_1 and N_2 to be sets of states in T_1 and T_2 respectively.
- By a similar argument,

$$\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2)$$

where $\Delta(n_1, n_2)$ is number of common sub-fragments at n_1, n_2

$$\text{e.g., } T_1 = \begin{array}{ccccccc} \mathrm{A} & — & \mathrm{B} & — & \mathrm{C} & — & \mathrm{D} \\ | & & | & & | & & | \\ \mathrm{a} & & \mathrm{b} & & \mathrm{c} & & \mathrm{d} \end{array} \qquad T_2 = \begin{array}{ccccccc} \mathrm{A} & — & \mathrm{B} & — & \mathrm{C} & — & \mathrm{E} \\ | & & | & & | & & | \\ \mathrm{a} & & \mathrm{b} & & \mathrm{e} & & \mathrm{e} \end{array}$$

$$\mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2) = \Delta(A, A) + \Delta(A, B) \ldots + \Delta(B, A) + \Delta(B, B) \ldots + \Delta(D, E)$$

e.g., $\Delta(B, B) = 4$,

$$\begin{array}{c} \mathrm{B} \\ \\ \\ \end{array} \qquad \begin{array}{c} \mathrm{B} \\ | \\ \mathrm{b} \end{array} \qquad \begin{array}{ccc} \mathrm{B} & — & \mathrm{C} \\ & & \\ & & \end{array} \qquad \begin{array}{ccc} \mathrm{B} & — & \mathrm{C} \\ | & & \\ \mathrm{b} & & \end{array}$$

The Recursive Definition for Tagged Sequences

- Define $N(n)$ = state following n, $W(n)$ = word at state n
- Define $\pi[W(n_1), W(n_2)] = 1$ iff $W(n_1) = W(n_2)$
- Then if labels at n_1 and n_2 are the same,

$$\Delta(n_1, n_2) = (1 + \pi[W(n_1), W(n_2)]) \times (1 + \Delta(N(n_1), N(n_2)))$$

---

e.g., $T_1 = \begin{array}{ccccccc} \mathrm{A} & — & \mathrm{B} & — & \mathrm{C} & — & \mathrm{D} \\ | & & | & & | & & | \\ \mathrm{a} & & \mathrm{b} & & \mathrm{c} & & \mathrm{d} \end{array}$ $T_2 = \begin{array}{ccccccc} \mathrm{A} & — & \mathrm{B} & — & \mathrm{C} & — & \mathrm{E} \\ | & & | & & | & & | \\ \mathrm{a} & & \mathrm{b} & & \mathrm{e} & & \mathrm{e} \end{array}$

$$\begin{aligned} \Delta(A, A) &= (1 + \pi[a, a]) \times (1 + \Delta(B, B)) \\ &= (1 + 1) \times (1 + 4) = 10 \end{aligned}$$

Refinements of the Kernels

- Include log probability from the baseline model:

  $\mathbf{\Phi}(T_1)$ is representation under all sub-fragments kernel

  $L(T_1)$ is log probability under baseline model

  New representation $\mathbf{\Phi}'$ where

  $$\mathbf{\Phi}'(T_1) \cdot \mathbf{\Phi}'(T_2) = \beta L(T_1)L(T_2) + \mathbf{\Phi}(T_1) \cdot \mathbf{\Phi}(T_2)$$

  (includes $L(T_1)$ as an additional component with weight $\sqrt{\beta}$)

- Allows the perceptron to use original ranking as default

Refinements of the Kernels

- Downweighting larger sub-fragments

$$\sum_{i=1}^{d} \lambda^{SIZE_i} h_i(T_1) h_i(T_2)$$

where $0 < \lambda \leq 1$,
$SIZE_i$ is number of states/rules in i'th fragment

- Simple modification to recursive definitions, e.g.,

$$\Delta(n_1, n_2) = (1 + \pi[W(n_1), W(n_2)]) \times (1 + \lambda \Delta(N(n_1), N(n_2))$$

Refinement of the Tagging Kernel

- Sub-fragments sensitive to spelling features (e.g., Capitalization)

• Define $\pi[x, y] = 1$ if x and y are identical,
$\pi[x, y] = 0.5$ if x and y share same capitalization features

$$\Delta(n_1, n_2) = (1+\pi[W(n_1), W(n_2)])\times(1+\lambda\Delta(N(n_1), N(n_2))$$

- Sub-fragments now include capitalization features

N — N — S
exiled to Elba

N — N — S
exiled to Cap

N — N — S
No cap to Cap

N — N — S
No cap No cap Cap

Experimental Results

Parsing Wall Street Journal

MODEL	$\leq$ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	2 CBs
CO99	88.1%	88.3%	1.06	64.0%	85.1%
VP	88.6%	88.9%	0.99	66.5%	86.3%

VP gives 5.1% relative reduction in error (CO99 = my thesis parser)

Named Entity Tagging on Web Data

	P	R	F
Max-Ent	84.4%	86.3%	85.3%
Perc.	86.1%	89.1%	87.6%
Improvement	10.9%	20.4%	15.6%

VP gives 15.6% relative reduction in error

Summary

- For any representation $\mathbf{\Phi}(x)$,
  Efficient computation of $\mathbf{\Phi}(x) \cdot \mathbf{\Phi}(y) \Rightarrow$
  Efficient learning through kernel form of the perceptron

- Dynamic programming can be used to calculate $\mathbf{\Phi}(x) \cdot \mathbf{\Phi}(y)$ under "all sub-fragments" representations

- Several refinements of the inner products:
  - Including probabilities from baseline model
  - Downweighting larger sub-fragments
  - Sensitivity to spelling features