MACHINE LEARNING

Kernel Methods

Alessandro Moschitti

Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it

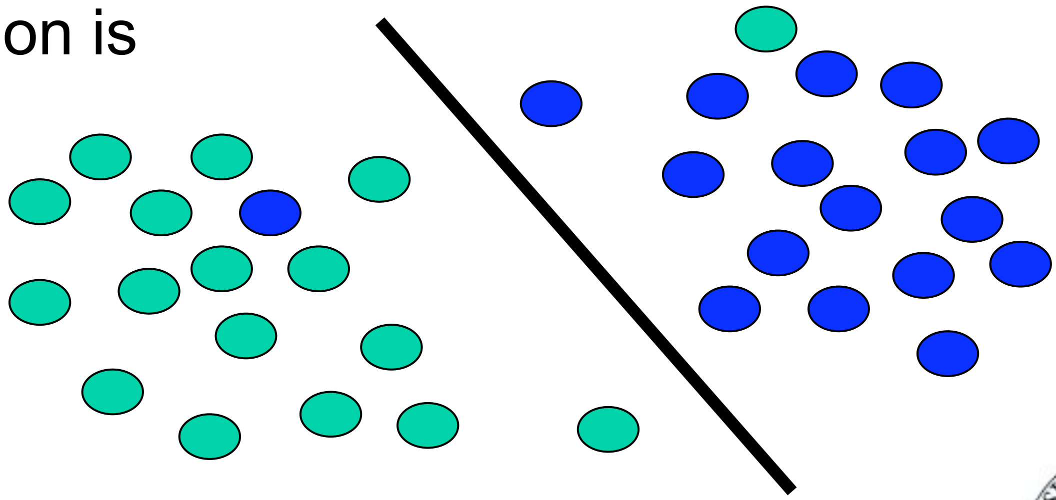
Linear Classifier

- The equation of a hyperplane is

$$f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x},\vec{w} \in \Re^n, b \in \Re$$

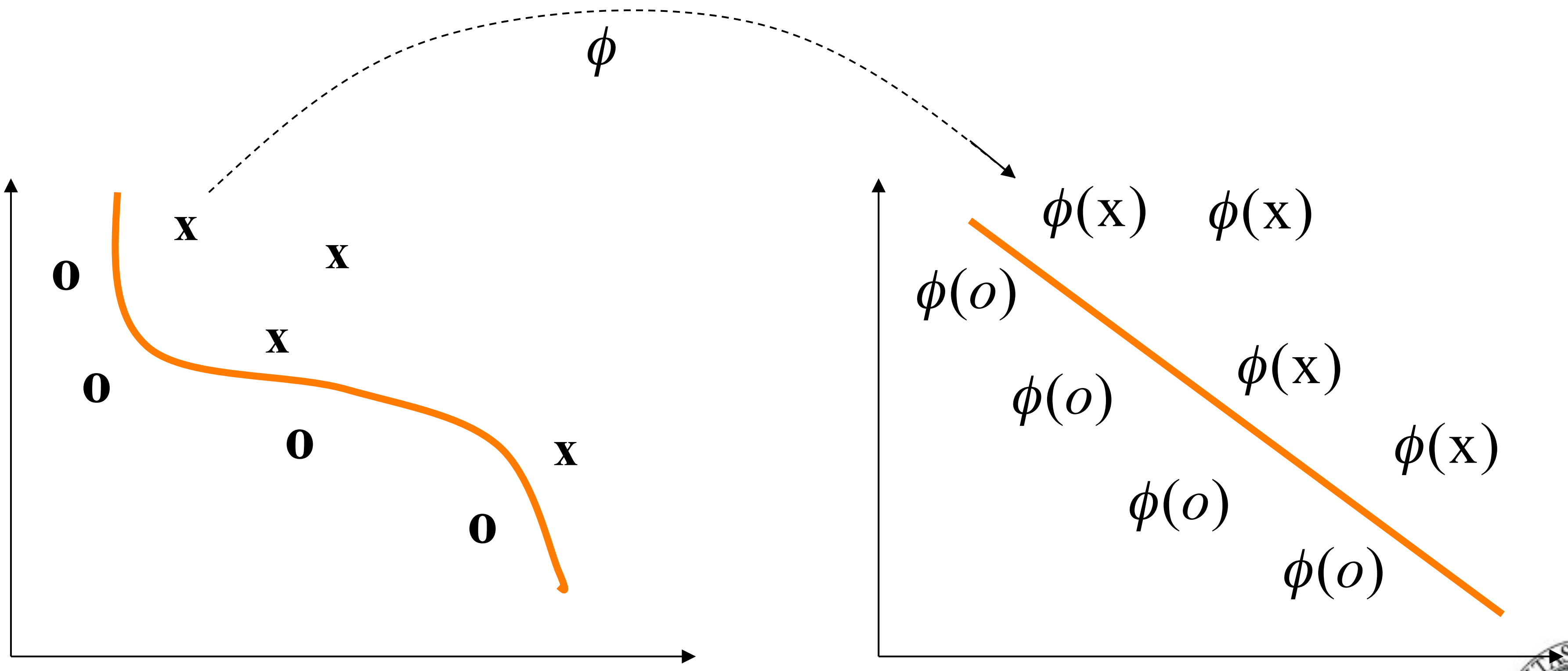
- $\vec{x}$ is the vector representing the classifying example
- $\vec{w}$ is the gradient of the hyperplane
- The classification function is

$$h(x) = \text{sign}(f(x))$$



The main idea of Kernel Functions

- Mapping vectors in a space where they are linearly separable $\vec{x} \rightarrow \phi(\vec{x})$



A mapping example

- Given two masses m_1 and m_2, one is constrained
- Apply a force f_a to the mass m_1
- Experiments
  - Features m_1, m_2 and f_a
- We want to learn a classifier that tells when a mass m_1 will get far away from m_2
- If we consider the Gravitational Newton Law

$$f(m_1, m_2, r) = C\frac{m_1 m_2}{r^2}$$

- we need to find when $f(m_1, m_2, r) < f_a$

A mapping example (2)

$$\vec{x} = (x_1, \ldots, x_n) \rightarrow \phi(\vec{x}) = (\phi_1(\vec{x}), \ldots, \phi_n(\vec{x}))$$

- The gravitational law is not linear so we need to change space

$$(f_a, m_1, m_2, r) \rightarrow (k, x, y, z) = (\ln f_a, \ln m_1, \ln m_2, \ln r)$$

- As

$$\ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2\ln r = c + x + y - 2z$$

- We need the hyperplane

$$\ln f_a - \ln m_1 - \ln m_2 + 2\ln r - \ln C = 0$$

$(\ln m_1, \ln m_2, -2\ln r) \cdot (x,y,z) - \ln f_a + \ln C = 0$, we can decide without error if the mass will get far away or not

A kernel-based Machine Perceptron training

$$\vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \le i \le l} \| \vec{x}_i \|$$

do

for i = 1 to ℓ

if $y_i(\vec{w}_k \cdot \vec{x}_i + b_k) \le 0$ then

$$\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$$

$$b_{k+1} = b_k + \eta y_i R^2$$

k = k + 1

endif

endfor

while an error is found

return k, $(\vec{w}_k, b_k)$

Dual Representation for Classification

- Each step of perceptron only training data is added with a certain weight

$$\vec{w} = \sum_{j=1..\ell} \alpha_j y_j \vec{x}_j$$

- So the classification function

$$\mathrm{sgn}(\vec{w} \cdot \vec{x} + b) = \mathrm{sgn}\left(\sum_{j=1..\ell} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b \right)$$

- Note that data only appears in the scalar product

Dual Representation for Learning

- as well as the updating function

$$\text{if } y_i\left(\sum_{j=1..\ell}\alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

- The learning rate η only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix $\eta = 1$.

Dual Perceptron algorithm and Kernel functions

- We can rewrite the classification function as

$$h(x) = \text{sgn}(\vec{w}_\phi \cdot \phi(\vec{x}) + b_\phi) = \text{sgn}(\sum_{j=1..\ell} \alpha_j y_j \phi(\vec{x}_j) \cdot \phi(\vec{x}) + b_\phi) =$$

$$= \text{sgn}(\sum_{i=1..\ell} \alpha_j y_j k(\vec{x}_j, \vec{x}) + b_\phi)$$

- As well as the updating function

$$\text{if } y_i \left(\sum_{j=1..\ell} \alpha_j y_j k(\vec{x}_j, \vec{x}_i) + b_\phi \right) \leq 0 \text{ allora } \alpha_i = \alpha_i + \eta$$

- The learning rate η does not affect the algorithm so we set it to $\eta = 1$.

Dual optimization problem of SVMs

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x_i} \cdot \vec{x_j} + \frac{1}{C} \delta_{ij} \right)$$

$$\alpha_i \geq 0, \quad \forall i = 1, .., m$$

$$\textstyle\sum_{i=1}^{m} y_i \alpha_i = 0$$

Kernels in Support Vector Machines

- In Soft Margin SVMs we maximize:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x_i} \cdot \vec{x_j} + \frac{1}{C} \delta_{ij} \right)$$

- By using kernel functions we rewrite the problem as:

$$\begin{cases} maximize \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(k(o_i, o_j) + \frac{1}{C} \delta_{ij} \right) \\ \alpha_i \geq 0, \quad \forall i = 1, .., m \\ \sum_{i=1}^{m} y_i \alpha_i = 0 \end{cases}$$

Kernel Function Definition

Def. 2.26 *A kernel is a function k, such that* $\forall\ \vec{x}, \vec{z} \in X$

$$k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

where ϕ *is a mapping from* X *to an (inner product) feature space.*

- Kernels are the product of mapping functions such as

$$\vec{x} \in \Re^n, \quad \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_m(\vec{x})) \in \Re^m$$

The Kernel Gram Matrix

- With KM-based learning, the **sole** information used from the training data set is the Kernel Gram Matrix

$$K_{training} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \dots & k(\mathbf{x}_1,\mathbf{x}_m) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \dots & k(\mathbf{x}_2,\mathbf{x}_m) \\ \dots & \dots & \dots & \dots \\ k(\mathbf{x}_m,\mathbf{x}_1) & k(\mathbf{x}_m,\mathbf{x}_2) & \dots & k(\mathbf{x}_m,\mathbf{x}_m) \end{bmatrix}$$

- If the kernel is valid, K is symmetric definite-positive .

Valid Kernels

Def. B.11 *Eigen Values*
Given a matrix $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$, *an egeinvalue* λ *and an egeinvector* $\vec{x} \in \mathbb{R}^n - \{\vec{0}\}$ *are such that*

$$\mathbf{A}\vec{x} = \lambda\vec{x}$$

Def. B.12 *Symmetric Matrix*
A square matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is symmetric iff* $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ *for* $i \neq j$ $i = 1, .., m$ *and* $j = 1, .., n$, *i.e. iff* $\mathbf{A} = \mathbf{A}'$.

Def. B.13 *Positive (Semi-) definite Matrix*
A square matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).*

Valid Kernels cont'd

Proposition 2.27 *(Mercer's conditions)*
Let X be a finite input space with $K(\vec{x}, \vec{z})$ a symmetric function on X. Then $K(\vec{x}, \vec{z})$ is a kernel function if and only if the matrix

$$k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

is positive semi-definite (has non-negative eigenvalues).

- If the matrix is positive semi-definite then we can find a mapping ϕ implementing the kernel function

Mercer's Theorem (finite space)

- Let us consider $\mathrm{K} = \left(K(\vec{x}_i, \vec{x}_j)\right)_{i,j=1}^{n}$
- K symmetric $\Rightarrow \exists\, \mathrm{V}$: $\mathrm{K} = \mathrm{V}\Lambda\mathrm{V}'$ for Takagi factorization of a complex-symmetric matrix, where:
  - Λ is the diagonal matrix of the eigenvalues λ_t of K
  - $\vec{\mathrm{v}}_t = \left(v_{ti}\right)_{i=1}^{n}$ are the eigenvectors, i.e. the columns of V
- Let us assume lambda values non-negative

$$\phi : \vec{x}_i \rightarrow \left(\sqrt{\lambda_t}\, v_{ti}\right)_{t=1}^{n} \in \Re^n,\ i = 1,..,n$$

Mercer's Theorem (sufficient conditions)

- Therefore

$$\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j) = \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} = \left(\mathrm{V} \Lambda \mathrm{V}'\right)_{ij} = \mathrm{K}_{ij} = K(\vec{x}_i, \vec{x}_j)$$

- which implies that K is a kernel function

Mercer's Theorem (necessary conditions)

- Suppose we have negative eigenvalues λ_s and eigenvectors $\vec{\mathrm{v}}_s$ the following point

$$\vec{z} = \sum_{i=1}^{n} v_{si}\,\Phi(\vec{x}_i) = \sum_{i=1}^{n} v_{si}\left(\sqrt{\lambda_t}\,v_{ti}\right)_t = \sqrt{\Lambda}\mathrm{V}'\vec{\mathrm{v}}_s$$

- has the following norm:

$$\left\|\vec{z}\right\|^2 = \vec{z}\cdot\vec{z} = \sqrt{\Lambda}\mathrm{V}'\vec{\mathrm{v}}_s\sqrt{\Lambda}\mathrm{V}'\vec{\mathrm{v}}_s = \vec{\mathrm{v}}_s'\,\mathrm{V}\sqrt{\Lambda}\sqrt{\Lambda}\mathrm{V}'\vec{\mathrm{v}}_s =$$

$$\vec{\mathrm{v}}_s'\,K\vec{\mathrm{v}}_s = \vec{\mathrm{v}}_s'\,\lambda_s\,\vec{\mathrm{v}}_s = \lambda_s\left\|\vec{\mathrm{v}}_s\right\|^2 < 0$$

this contradicts the geometry of the space.

Is it a valid kernel?

- It may not be a kernel so we can use ***M´·M***

Proposition B.14 *Let* $\boldsymbol{A}$ *be a symmetric matrix. Then* A *is positive (semi-) definite iff for any vector* $\vec{x} \neq 0$

$$\vec{x}'\boldsymbol{A}\vec{x} > 0 \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $\boldsymbol{A}$ in $\boldsymbol{M}'\boldsymbol{M}$, then $\boldsymbol{A}$ is semi-definite positive matrix as

$$\vec{x}'\boldsymbol{A}\vec{x} = \vec{x}'\boldsymbol{M}'\boldsymbol{M}\vec{x} = (\boldsymbol{M}\vec{x})'(\boldsymbol{M}\vec{x}) = \boldsymbol{M}\vec{x} \cdot \boldsymbol{M}\vec{x} = ||\boldsymbol{M}\vec{x}||^2 \geq 0.$$

Valid Kernel operations

- $k(x,z) = k_1(x,z)+k_2(x,z)$
- $k(x,z) = k_1(x,z)*k_2(x,z)$
- $k(x,z) = \alpha\, k_1(x,z)$
- $k(x,z) = f(x)f(z)$
- $k(x,z) = k_1(\phi(x),\phi(z))$
- $k(x,z) = x'Bz$

Basic Kernels for unstructured data

- Linear Kernel
- Polynomial Kernel
- Lexical kernel
- String Kernel

Linear Kernel

- In Text Categorization documents are word vectors

$$\Phi(d_x) = \vec{x} = (0,..,1,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,1)$$

buy acquisition stocks sell market

$$\Phi(d_z) = \vec{z} = (0,..,1,..,0,..,1,..,0,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$

buy company stocks sell

- The dot product $\vec{x} \cdot \vec{z}$ counts the number of features in common
- This provides a sort of *similarity*

Feature Conjunction (polynomial Kernel)

- The initial vectors are mapped in a higher space

$$\Phi(<x_1, x_2>) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- More expressive, as (x_1x_2) encodes

  `Stock+Market` vs. **`Downtown+Market`** features

- We can smartly compute the scalar product as

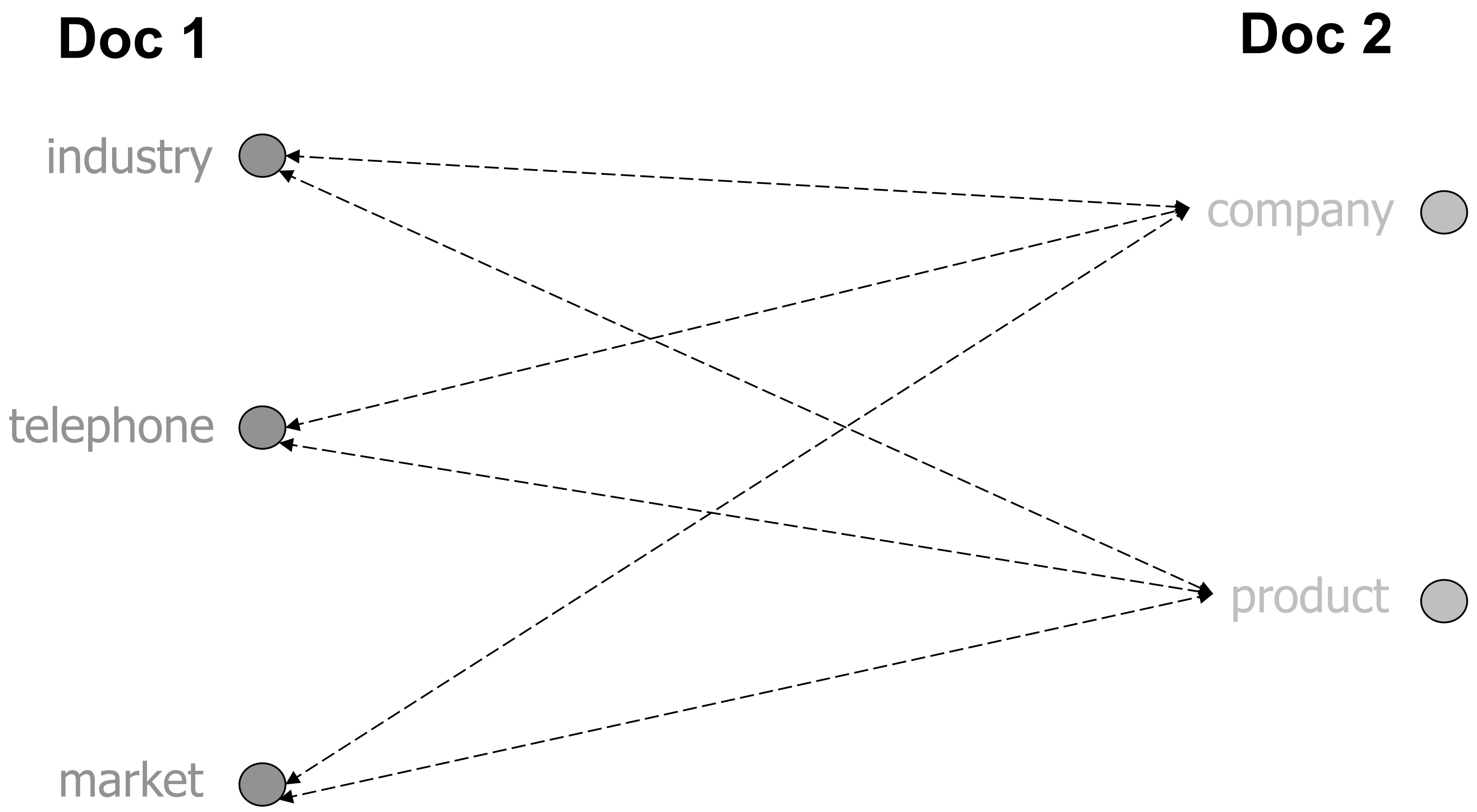
$$\Phi(\vec{x}) \cdot \Phi(\vec{z}) =$$

$$= (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) =$$

$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 =$$

$$= (x_1z_1 + x_2z_2 + 1)^2 = (\vec{x} \cdot \vec{z} + 1)^2 = K_{Poly}(\vec{x}, \vec{z})$$

Document Similarity



Lexical Semantic Kernel [CoNLL 2005]

- The document similarity is the SK function:

$$SK(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} s(w_1, w_2)$$

- where *s* is any similarity function between words, e.g. WordNet [Basili et al.,2005] similarity or LSA [Cristianini et al., 2002]
- Good results when training data is small

Using character sequences

$$\phi(\text{"}bank\text{"}) = \vec{x} = (0,..,1,..,0,..,1,..,0,......1,..,0,..,1,..,0,..,1,..,0)$$

bank ank bnk bk b

$$\phi(\text{"}rank\text{"}) = \vec{z} = (1,..,0,..,0,..,1,..,0,......0,..,1,..,0,..,1,..,0,..,1)$$

rank ank rnk rk r

- $\vec{x} \cdot \vec{z}$ counts the number of common substrings

$$\vec{x} \cdot \vec{z} = \phi(\text{"}bank\text{"}) \cdot \phi(\text{"}rank\text{"}) = k(\text{"}bank\text{"},\text{"}rank\text{"})$$

String Kernel

- Given two strings, the number of matches between their substrings is evaluated
- E.g. Bank and Rank
  - B, a, n, k, Ba, Ban, Bank, Bk, an, ank, nk,..
  - R, a , n , k, Ra, Ran, Rank, Rk, an, ank, nk,..
- String kernel over sentences and texts
- Huge space but there are efficient algorithms

Formal Definition

$$s = s_1, .., s_{|s|}$$

$$\vec{I} = (i_1, ..., i_{|u|}) \qquad u = s[\vec{I}]$$

$$\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})}, \text{ where } \quad l(\vec{I}) = i_{|u|} - i_1 + 1$$

$$K(s,t) = \sum_{u\in\Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u\in\Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{J})} =$$

$$= \sum_{u\in\Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{I})+l(\vec{J})}, \text{ where } \quad \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Kernel between Bank and Rank

B, a, n, k, Ba, Ban, Bank, an, ank, nk, Bn, Bnk, Bk and ak are the substrings of *Bank*.

R, a, n, k, Ra, Ran, Rank, an, ank, nk, Rn, Rnk, Rk and ak are the substrings of *Rank*.

An example of string kernel computation

- $\phi_{\mathsf{a}}(\mathsf{Bank}) = \phi_{\mathsf{a}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_{\mathsf{n}}(\mathsf{Bank}) = \phi_{\mathsf{n}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_{\mathsf{k}}(\mathsf{Bank}) = \phi_{\mathsf{k}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(4-4+1)} = \lambda,$
- $\phi_{\mathsf{an}}(\mathsf{Bank}) = \phi_{\mathsf{an}}(\mathsf{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(3-2+1)} = \lambda^2,$
- $\phi_{\mathsf{ank}}(\mathsf{Bank}) = \phi_{\mathsf{ank}}(\mathsf{Rank}) = \lambda^{(i_3 - i_1 + 1)} = \lambda^{(4-2+1)} = \lambda^3,$
- $\phi_{\mathsf{nk}}(\mathsf{Bank}) = \phi_{\mathsf{nk}}(\mathsf{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(4-3+1)} = \lambda^2$
- $\phi_{\mathsf{ak}}(\mathsf{Bank}) = \phi_{\mathsf{ak}}(\mathsf{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(4-2+1)} = \lambda^3$

$$K(\mathsf{Bank}, \mathsf{Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3) \cdot (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)$$
$$= 3\lambda^2 + 2\lambda^4 + 2\lambda^6$$

Efficient Evaluation

- Dynamic Programming technique
- Evaluate the spectrum string kernels
  - Substrings of size *p*
- Sum the contribution of the different spectra

Efficient Evaluation

Given two sequences s_1a and s_2b, we define:

$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r]),$$

$s_1[1:i]$ and $s_2[1:r]$ are their subsequences from 1 to i and 1 to r.

$$SK_p(s_1a, s_2b) = \begin{cases} \lambda^2 \times D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

D_p satisfies the recursive relation:

$$D_p(k, l) = SK_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) + \\ + \lambda D_p(k-1, l) - \lambda^2 D_p(k-1, l-1)$$

An example: SK("Gatta","Cata")

- First, evaluate the SK with size p=1, i.e. "a", "a","t","t","a","a"
- Store this in the table

$SK_{p=1}$	g	a	t	t	a
c	0	0	0	0	0
a	0	λ^2	0	0	λ^2
t	0	0	λ^2	λ^2	0
a	0	λ^2	0	0	λ^2

Evaluating DP2

- Evaluate the weight of the string of size p in case a character will be matched
- This is done by multiplying the double summation by the number of substrings of size p-1

$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r])$$

Evaluating the Predictive DP on strings of size 2 (second row)

- Let's consider substrings of size 2 and suppose that:
  - we have matched the first "a"
  - we will match the next character that we will add to the two strings
- We compute the weights of matches above at different string positions with some not-yet known character "?"
- If the match occurs immediately after "a" the weight will be λ^{1+1} x $\lambda^{1+1} = \lambda^4$ and we store just λ^2 in the DP entry in ["a","a"]

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4+\lambda^2$	$\lambda^5+\lambda^3+\lambda^2$

Evaluating the DP wrt different positions (second row)

- If the match for "gatta" occurs after "t" the weight will be λ^{1+2} (x λ^2 = λ^5) since the substring for it will be with "a□?"
- We write such prediction in the entry ["a","t"]
- Same rationale for a match after the second "t": we have the substring "a□□?" (matching with "a?" from "catta") for a weight of λ^{3+1} (x λ^2)

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4 + \lambda^2$	$\lambda^5 + \lambda^3 + \lambda^2$

Evaluating the DP wrt different positions (third row)

- If the match occurs after "t" of "cata", the weight will be λ^{2+1} (x $\lambda^2 = \lambda^5$) since it will be with the string "a□?", with a weight of λ^3
- If the match occurs after "t" of both "gatta" and "cata", there are two ways to compose substring of size two: "a□?" with weight λ^4 or "t?" with weight $\lambda^2 \Longrightarrow$ the total is $\lambda^2+\lambda^4$

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4 + \lambda^2$	$\lambda^5 + \lambda^3 + \lambda^2$

Evaluating the DP wrt different positions (third row)

- The final case is a match after the last “t” of both “cat” and “gatta”
- There are three possible substrings of “gatta”:
  - “a□□?”, “t□?”, “t?” for “gatta” with weight λ^3 , λ^2 or λ, respectively.
- There are two possible substrings of “cata”
  - “a□?”, “t?” with weight λ^2 and λ
  - Their match gives weights: λ^5 , λ^3, λ^2 $\Rightarrow$ by summing: $\lambda^5 + \lambda^3 + \lambda^2$

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4 + \lambda^2$	$\lambda^5 + \lambda^3 + \lambda^2$

Evaluating SK of size 2 using DP2

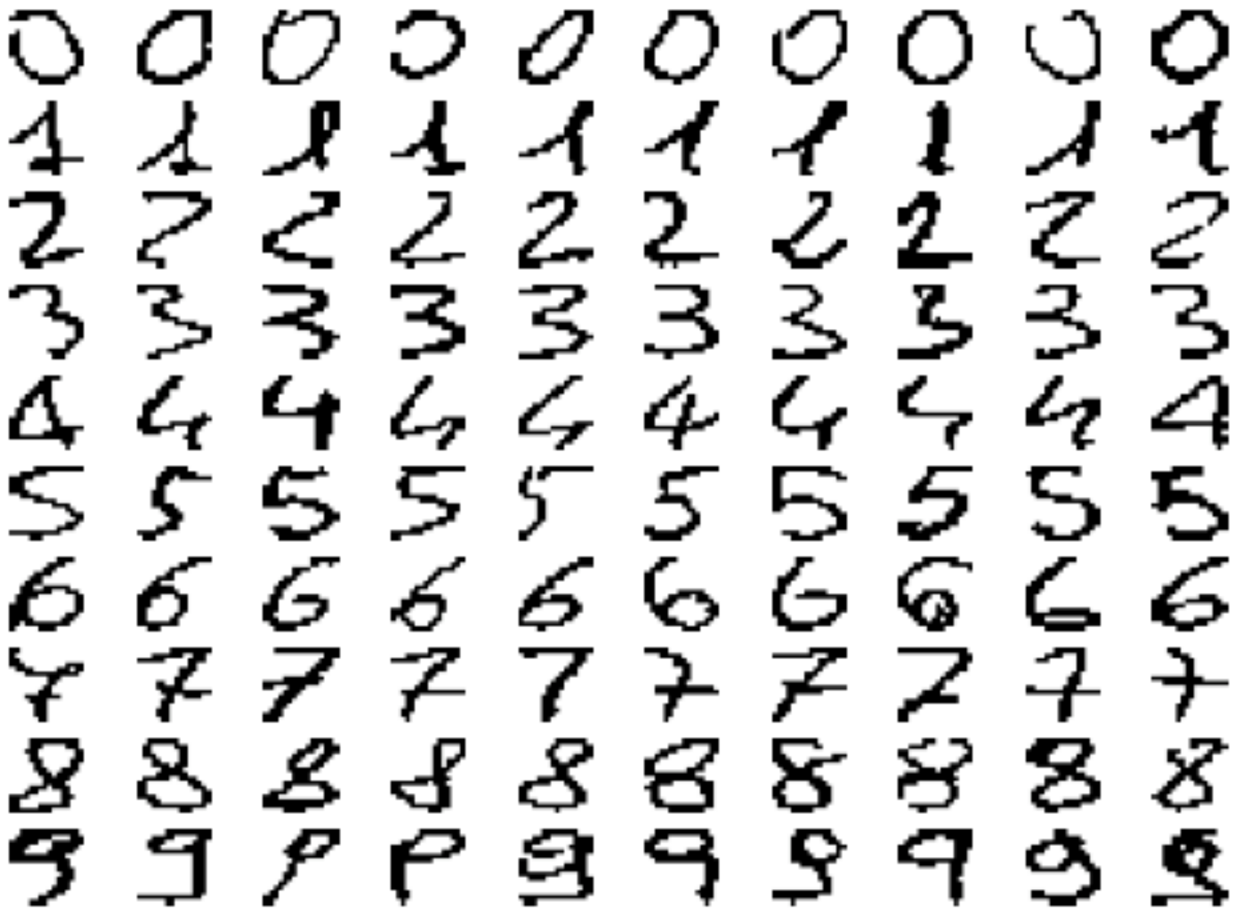
$$SK_p(s_1 a, s_2 b) = \begin{cases} \lambda^2 \times D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad \textit{otherwise}. \end{cases}$$

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4 + \lambda^2$	$\lambda^5 + \lambda^3 + \lambda^2$

$SK_{p=2}$	g	a	t	t	a
c	0	0	0	0	0
a	0	0	0	0	0
t	0	0	λ^4	λ^5	0
a	0	0	0	0	$\lambda^7 + \lambda^5 + \lambda^4$

- The number (weight) of substrings of size 2 between “gat” and “cat” is $\lambda^4 = \lambda^2$ ([“a”,”a”] entry of DP) x λ^2(cost of one character), where a = “t” and b = “t”.
- Between “gatta” and “cata” is $\lambda^7 + \lambda^5 + \lambda^4$, i.e the matches of “a□□a”, “t□a”, “ta” with “a□a” and “ta”.

String Kernels for OCR



Pixel Representation

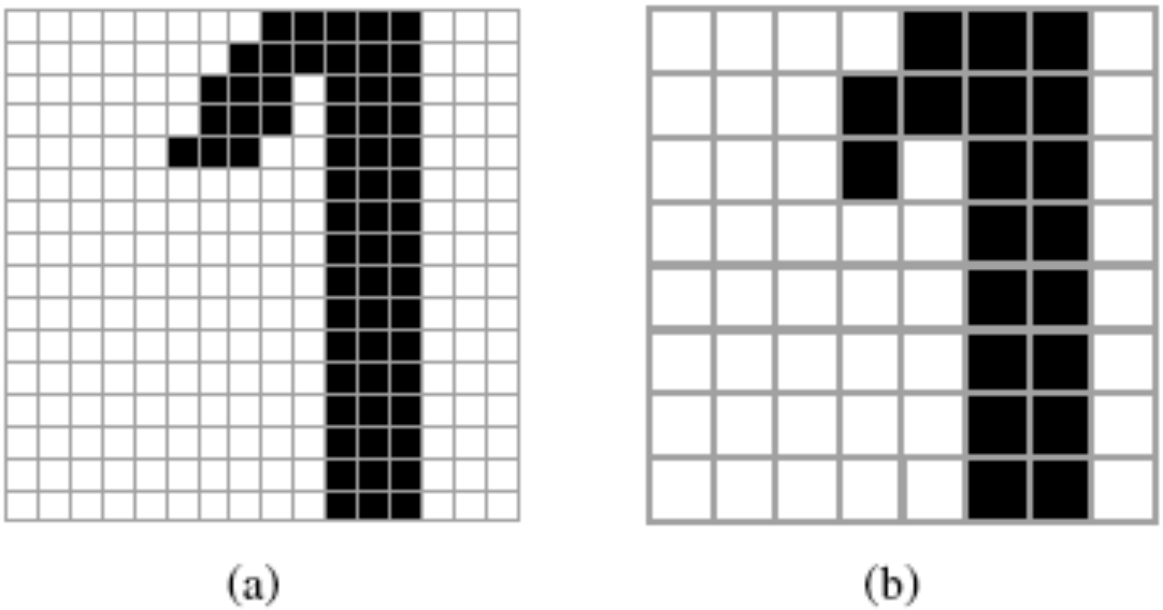


Figure 6: Resampling of an image from 16×16 to 8×8 format

Sequence of bits

L1	000**111**00
.	00**1111**00
.	00**1011**00
.	0000**11**00
	0000**11**00
L8	0000**11**00

$$SK(im_a, im_b) = \sum_{i=1..8} SK(L_a^i, L_b^i)$$

Results

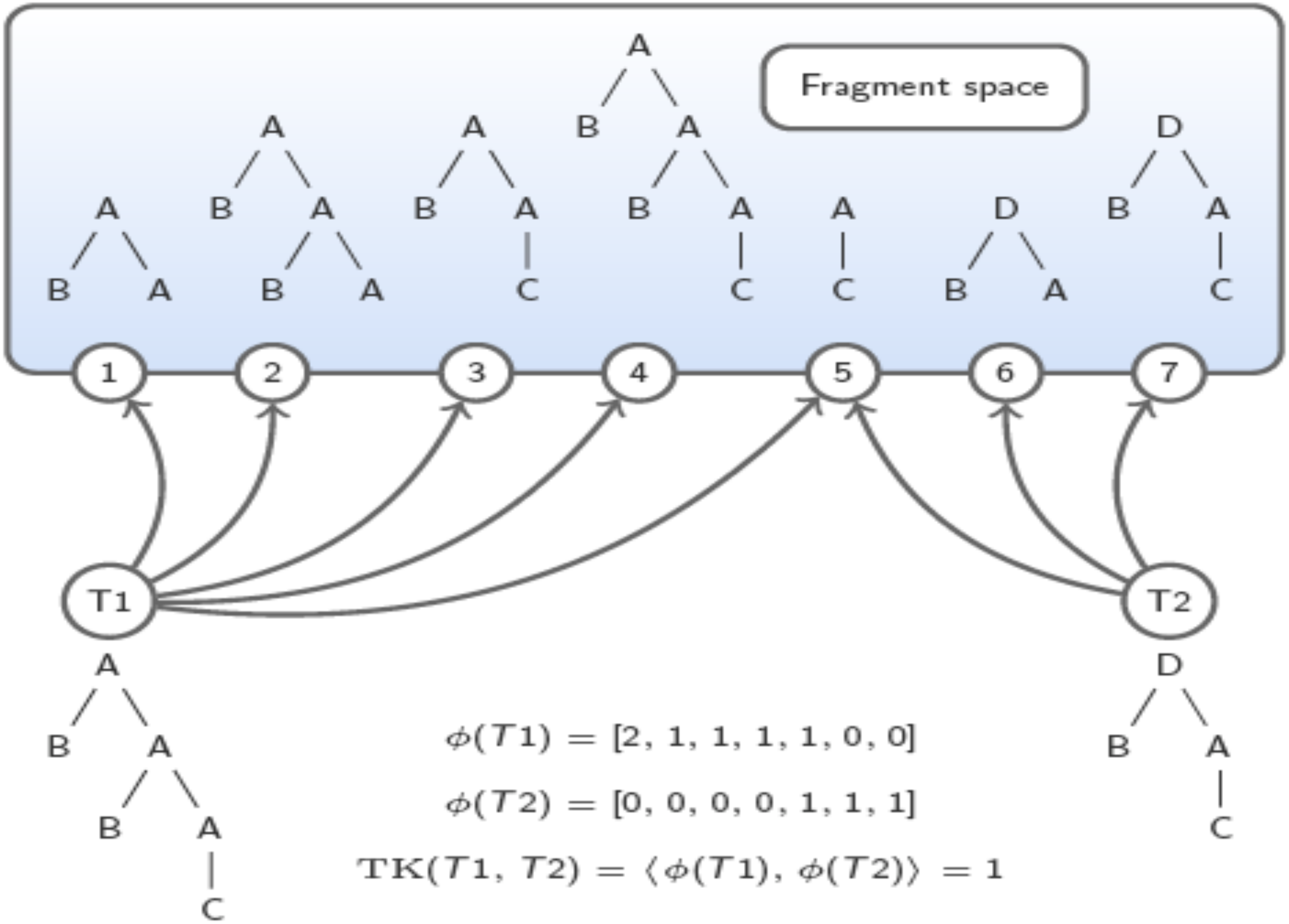
- Using columns+rows+diagonals

Digit	Precision	Recall	F1
0	97.78	97.78	97.78
1	95.45	93.33	94.38
2	93.62	97.78	95.65
3	93.33	93.33	93.33
4	97.83	100.00	98.90
5	97.67	93.33	95.45
6	100.00	97.78	98.88
7	91.84	100.00	95.74
8	93.18	91.11	92.13
9	93.02	88.89	90.91
Multiclass accuracy	**95.33**		

Tree kernels

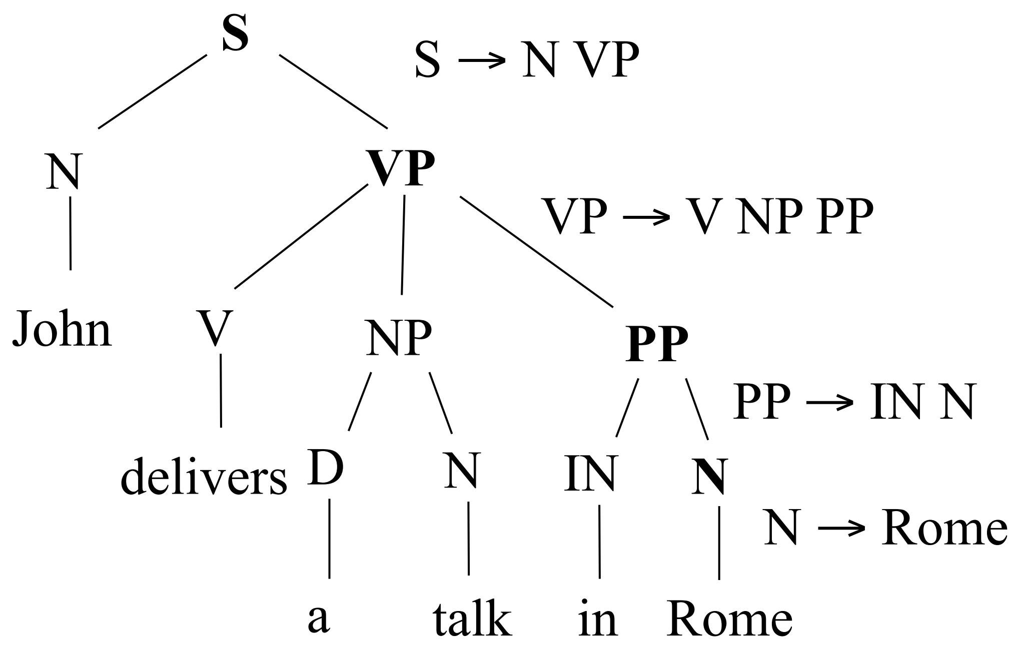
- Subtree, Subset Tree, Partial Tree kernels
- Efficient computation

Main Idea of Tree Kernels

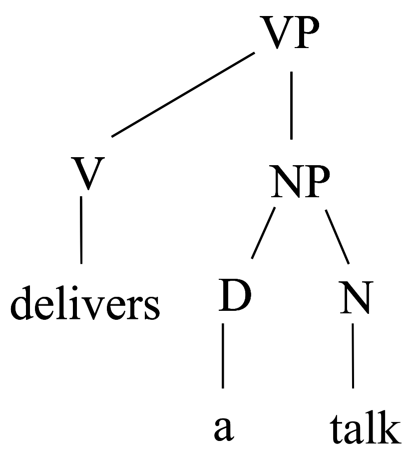


Example of a syntactic parse tree

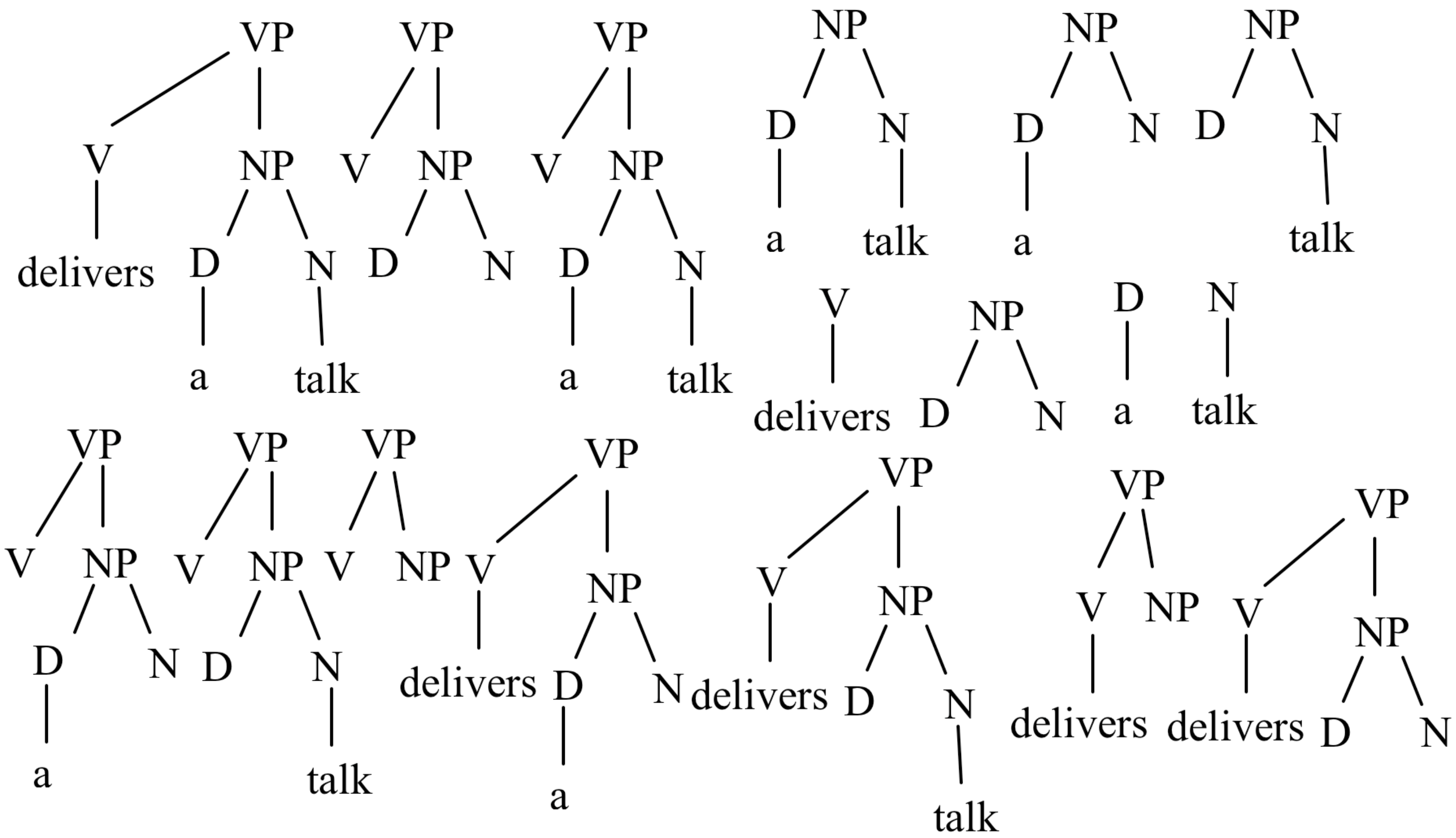
- "John delivers a talk in Rome"



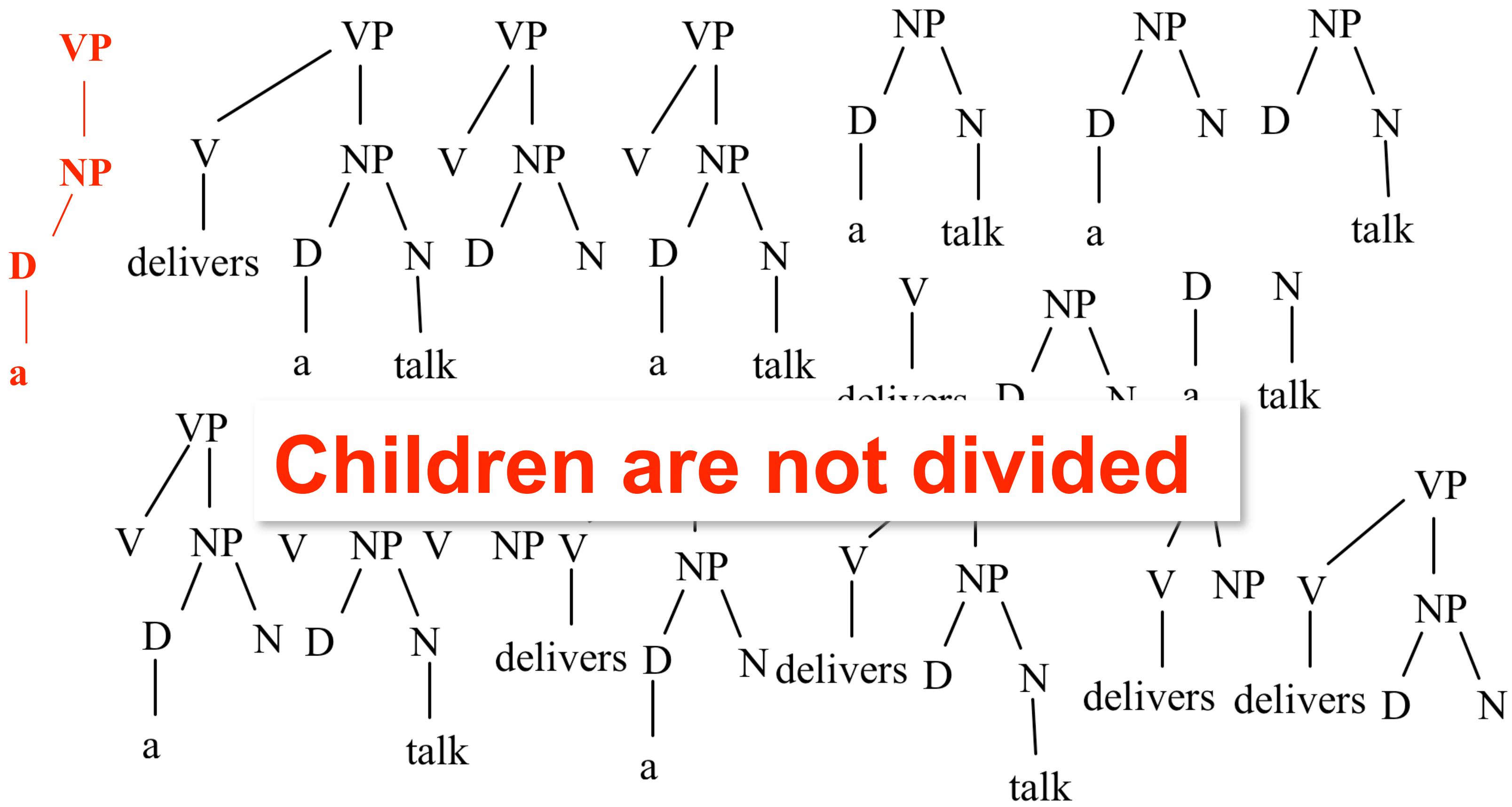
The Syntactic Tree Kernel (STK)
[Collins and Duffy, 2002]



The overall fragment set

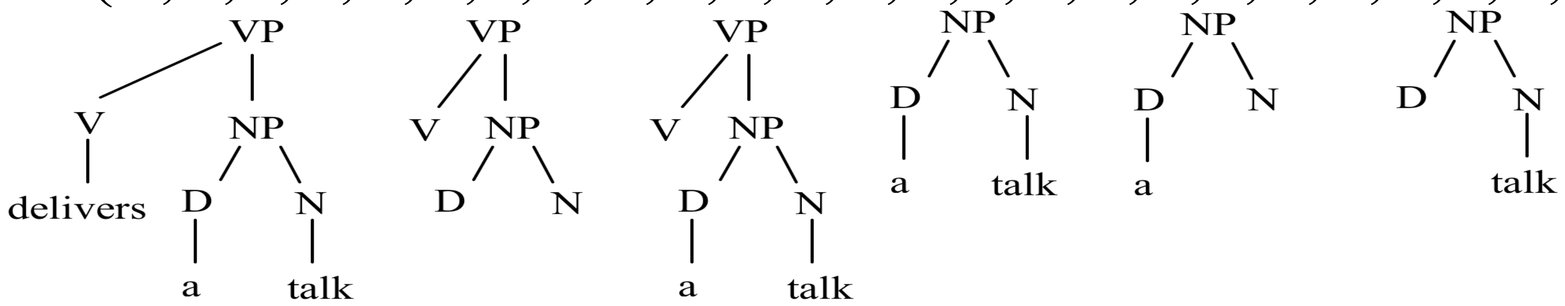


The overall fragment set

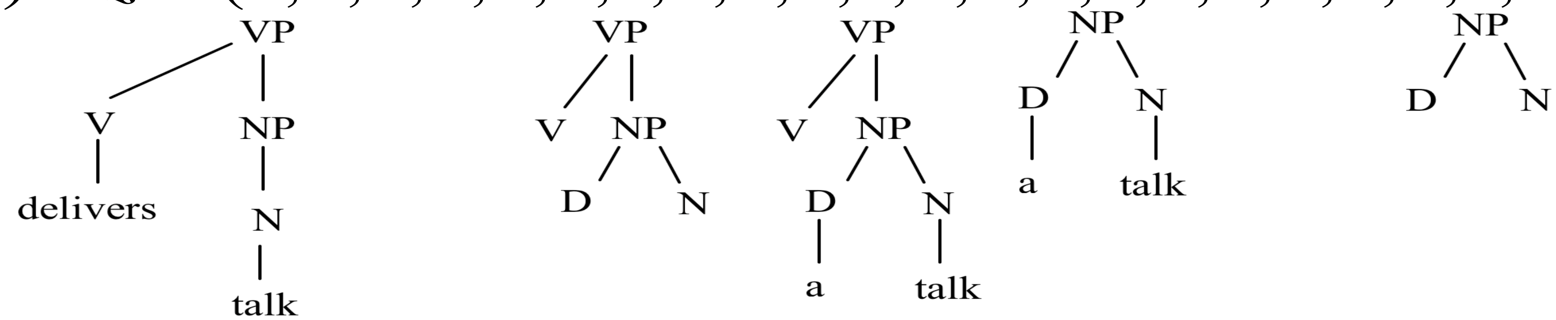


Explicit kernel space

$$\phi(T_x) = \vec{x} = (0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0)$$



$$\phi(T_z) = \vec{z} = (1,..,0,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$



- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

Efficient evaluation of the scalar product

$$\vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) =$$

$$= \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z)$$

Efficient evaluation of the scalar product

$$\vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) =$$

$$= \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z)$$

- [Collins and Duffy, ACL 2002] evaluate Δ in $O(n^2)$:

$\Delta(n_x, n_z) = 0,$ if the productions are different else

$\Delta(n_x, n_z) = 1,$ if pre-terminals else

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

Other Adjustments

- Decay factor

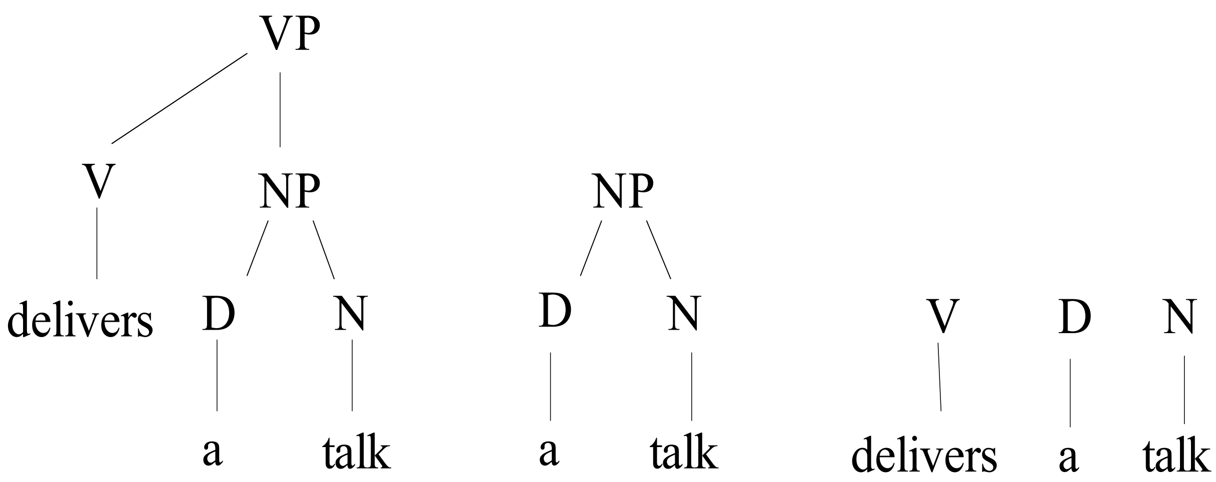
$$\Delta(n_x, n_z) = \lambda, \quad \text{if pre-terminals else}$$

$$\Delta(n_x, n_z) = \lambda \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))$$

- Normalization

$$K'(T_x, T_z) = \frac{K(T_x, T_z)}{\sqrt{K(T_x, T_x) \times K(T_z, T_z)}}$$

SubTree (ST) Kernel [Vishwanathan and Smola, 2002]



Evaluation

- Given the equation for STK

$$\Delta(n_x, n_z) = 0, \text{ if the productions are different else}$$

$$\Delta(n_x, n_z) = 1, \text{ if pre-terminals else}$$

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

Evaluation

- Given the equation for STK

$$\Delta(n_x, n_z) = 0, \text{ if the productions are different else}$$

$$\Delta(n_x, n_z) = 1, \text{ if pre-terminals else}$$

$$\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (\Delta(ch(n_x, j), ch(n_z, j)))$$

Fast Evaluation of STK [Moschitti, EACL 2006]

$$K(T_x, T_z) = \sum_{\langle n_x, n_z \rangle \in NP} \Delta(n_x, n_z)$$

$$NP = \left\{ \langle n_x, n_z \rangle \in T_x \times T_z : \Delta(n_x, n_z) \neq 0 \right\} =$$

$$= \left\{ \langle n_x, n_z \rangle \in T_x \times T_z : P(n_x) = P(n_z) \right\},$$

where $P(n_x)$ and $P(n_z)$ are the production rules used at nodes n_x and n_z

```
function Evaluate_Pair_Set(Tree T1, T2) returns NODE_PAIR_SET;
LIST L1,L2;
NODE_PAIR_SET Np;
begin
    L1 = T1.ordered_list;
    L2 = T2.ordered_list; /*the lists were sorted at loading time*/
    n1 = extract(L1); /*get the head element and*/
    n2 = extract(L2); /*remove it from the list*/
    while (n1 and n2 are not NULL)
        if (production_of(n1) > production_of(n2))
            then n2 = extract(L2);
            else if (production_of(n1) < production_of(n2))
                then n1 = extract(L1);
                else
                    while (production_of(n1) == production_of(n2))
                        while (production_of(n1) == production_of(n2))
                            add(⟨n1, n2⟩, Np);
                            n2=get_next_elem(L2); /*get the head element
                            and move the pointer to the next element*/
                        end
                        n1 = extract(L1);
                        reset(L2); /*set the pointer at the first element*/
                    end
    end
    return Np ;
end
```

Running Time Complexity

- We order the production rules used in T_x and T_z, at loading time
- At learning time we may evaluate NP in

$|T_x| + |T_z|$ *running time*

- If T_x and T_z are generated by only one production rule $\Rightarrow$ $O(|T_x| \times |T_z|)\ldots$

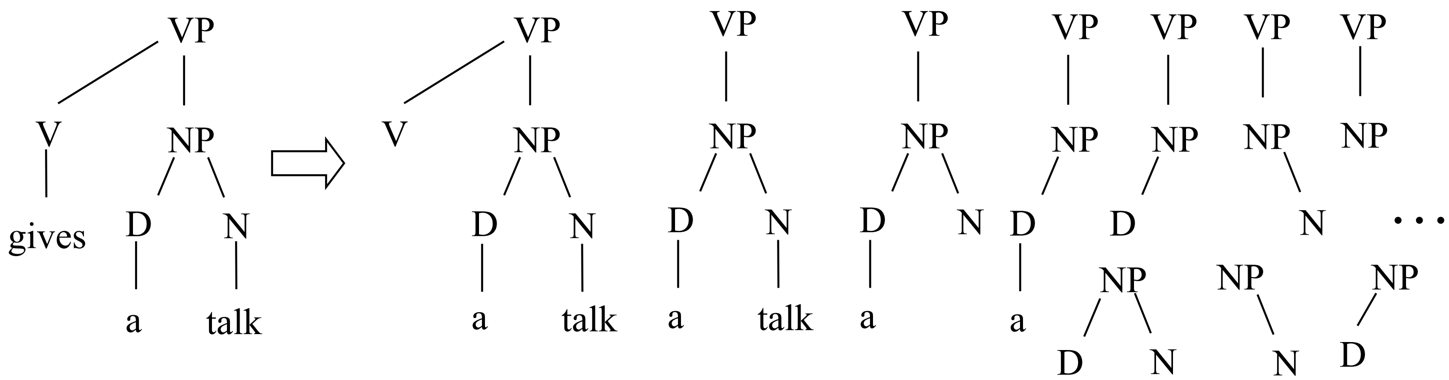
Running Time Complexity

- We order the production rules used in T_x and T_z, at loading time
- At learning time we may evaluate NP in

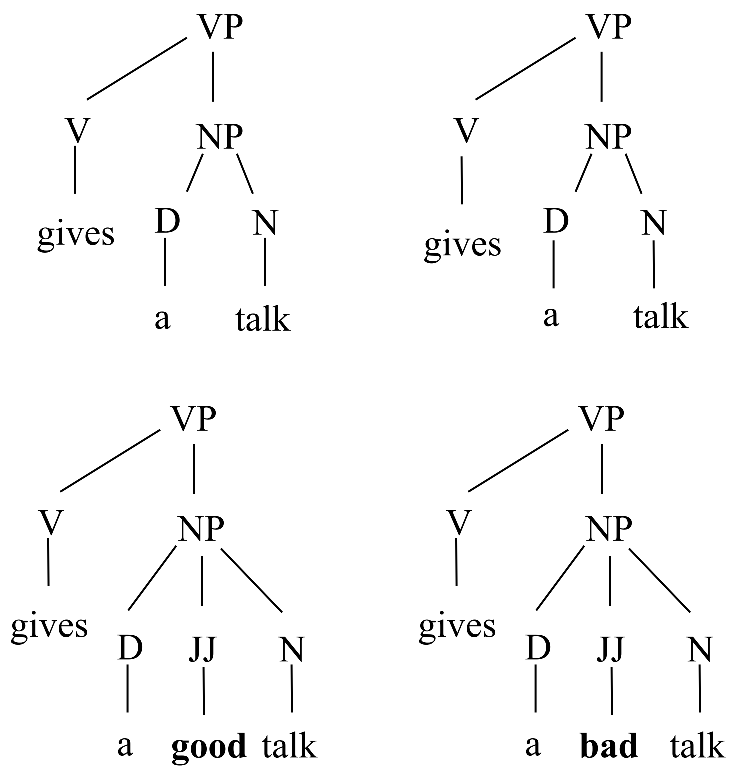
  $|T_x|+|T_z|$ *running time*
- If T_x and T_z are generated by only one production rule $\Rightarrow$ $O(|T_x|\times|T_z|)$…***Very Unlikely!!!!***

Labeled Ordered Tree Kernel

- STK satisfies the constraint “remove 0 or all children at a time”.
- If we relax such constraint we get more general substructures [Kashima and Koyanagi, 2002]



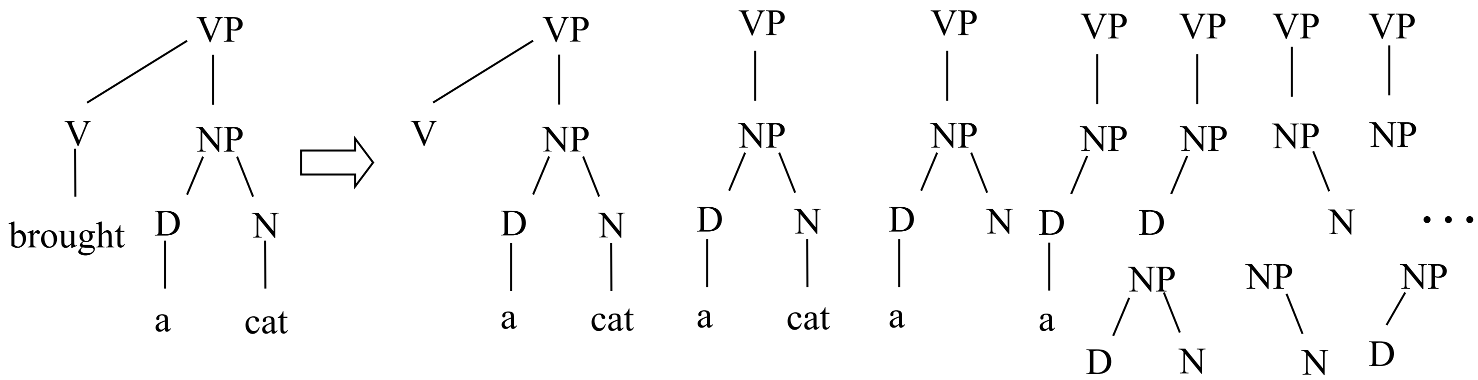
Weighting Problems



- Both matched pairs give the same contribution.
- Gap based weighting is needed.
- A novel efficient evaluation has to be defined

Partial Trees, [Moschitti, ECML 2006]

- STK + String Kernel with weighted gaps on Nodes' children



Partial Tree Kernel

- if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

- else

$$\Delta(n_1, n_2) = 1 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])$$

- By adding two decay factors we obtain:

$$\mu\Big(\lambda^2 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \lambda^{d(\vec{J}_1) + d(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])\Big)$$

Efficient Evaluation (1)

- In [Taylor and Cristianini, 2004 book], sequence kernels with weighted gaps are factorized with respect to different subsequence sizes.
- We treat children as sequences and apply the same theory

$$\Delta(n_1, n_2) = \mu\big(\lambda^2 + \textstyle\sum_{p=1}^{lm} \Delta_p(c_{n_1}, c_{n_2})\big),$$

Given the two child sequences $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ (a and b are the last children), $\Delta_p(s_1 a, s_2 b) =$

$$\Delta(a,b) \times \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r])$$

D_p

Efficient Evaluation (2)

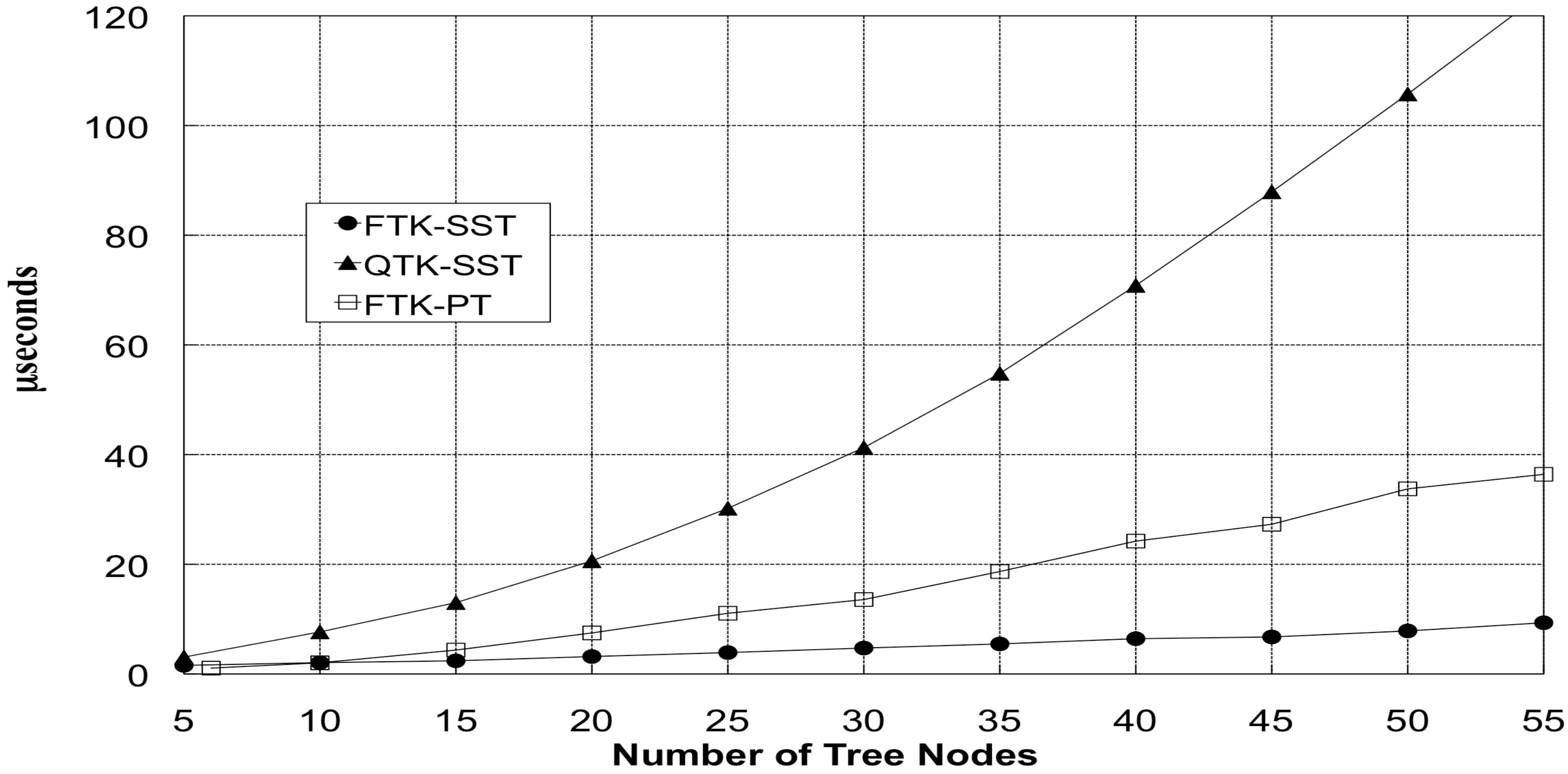
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

$$\begin{aligned} D_p(k,l) \quad &= \Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) \\ &+ \lambda D_p(k-1, l) + \lambda^2 D_p(k-1, l-1). \end{aligned}$$

- The complexity of finding the subsequences is $O(p|s_1||s_2|)$
- Therefore the overall complexity is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ where ρ is the maximum branching factor ($p = \rho$)

Running Time of Tree Kernel Functions



SVM-light-TK Software

- Encodes ST, STK and combination kernels in SVM-light [Joachims, 1999]
- Available at http://dit.unitn.it/~moschitt/
- Tree forests, vector sets
- The new SVM-Light-TK toolkit will be released asap (email me to have the current version)

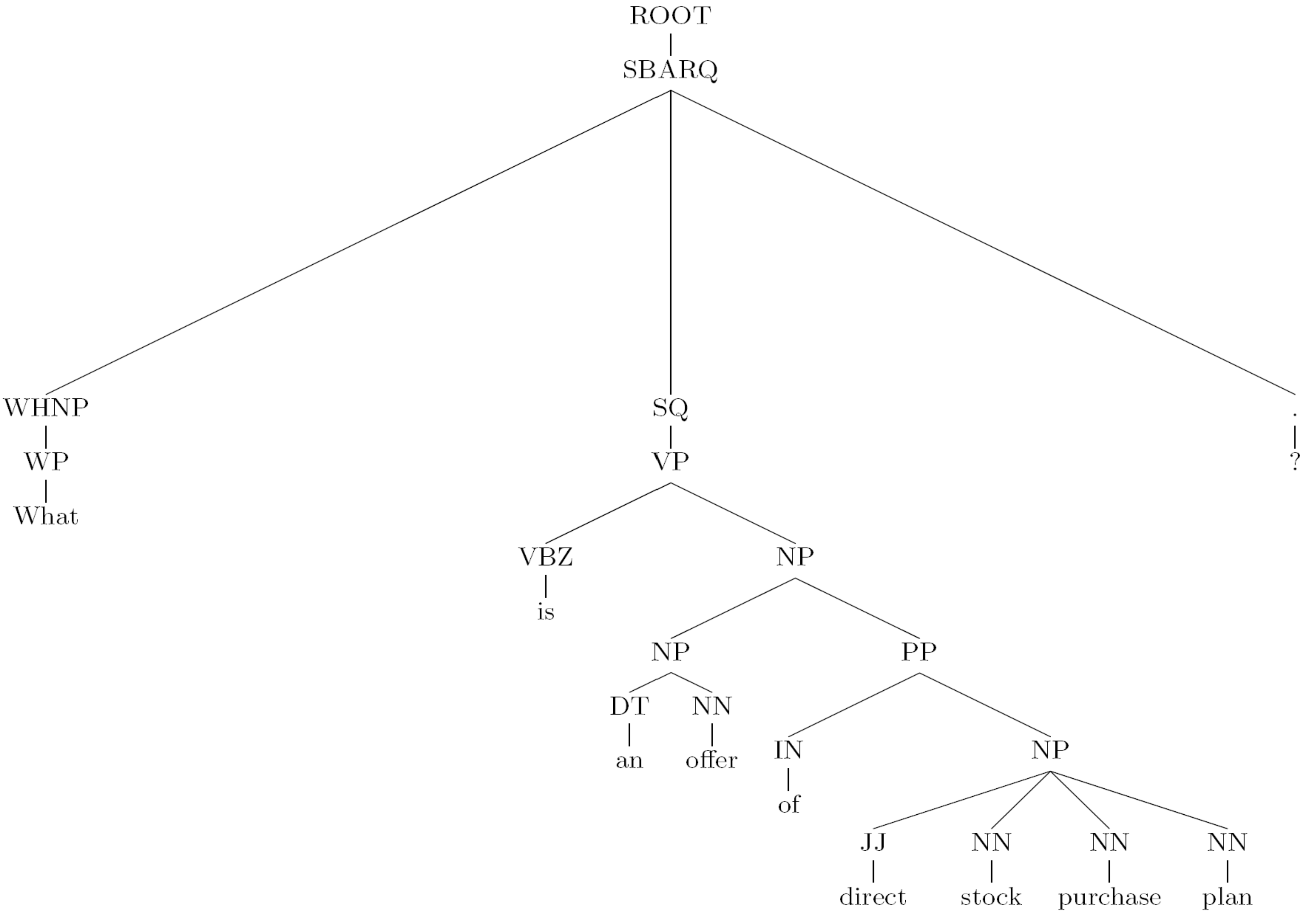
Practical Example on Question Classification

- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: How did Bob Marley die?
- **Numeric**: When was Martin Luther King Jr. born?
- **Organization**: What company makes Bentley cars?

Question Classifier based on Tree Kernels

- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Cross-validation (10-folds)
- Using the whole question parse trees
  - Constituent parsing
  - Example

"What is an offer of direct stock purchase plan ?"


ROOT
SBARQ
WHNP
WP
What
SQ
VP
VBZ
is
NP
NP
DT
an
NN
offer
PP
IN
of
NP
JJ
direct
NN
stock
NN
purchase
NN
plan
.
?

Data Format

- **"What does HTML stand for?"**
- 1 |**BT**| (SBARQ (WHNP (WP What))(SQ (AUX does)(NP (NNP S.O.S.))(VP (VB stand)(PP (IN for))))(. ?))|**ET**|

Trees + Feature Vectors

- **"What does HTML stand for?"**
- 1 |**BT**| (SBARQ (WHNP (WP What))(SQ (AUX does)(NP (NNP S.O.S.))(VP (VB stand)(PP (IN for))))(. ?))|**ET**|

  2:1 21:1.4421347148614654E-4 23:1 31:1 36:1 39:1 41:1 46:1 49:1 52:1 66:1 152:1 246:1 333:1 392:1 **|EV|**

Basic Commands

- Training and classification
  - `./svm_learn -t 5 train.dat model`
  - `./svm_classify test.dat model`

Conclusions

- Dealing with noisy and errors of NLP modules require robust approaches
- SVMs are robust to noise and Kernel methods allows for:
  - Syntactic information via STK
  - Shallow Semantic Information via PTK
  - Word/POS sequences via String Kernels
- When the IR task is complex, syntax and semantics are essential

⇒ Great improvement in Q/A classification

- SVM-Light-TK: an efficient tool to use them

SVM-light-TK Software

- Encodes ST, SST and combination kernels in SVM-light [Joachims, 1999]
- Available at http://dit.unitn.it/~moschitt/
- Tree forests, vector sets
- New extensions: the PT kernel will be released asap

Data Format

- **“What does Html stand for?”**
- 1 |**BT**| (SBARQ (WHNP (WP What))(SQ (AUX does)(NP (NNP S.O.S.))(VP (VB stand)(PP (IN for))))(. ?))

|**BT**| (***BOW*** (What *)(does *)(S.O.S. *)(stand *)(for *)(? *))

|**BT**| (***BOP*** (WP *)(AUX *)(NNP *)(VB *)(IN *)(. *))

|**BT**| (***PAS*** (ARG0 (R-A1 (What *)))(ARG1 (A1 (S.O.S. NNP)))(ARG2 (rel stand)))

|**ET**| 1:1 21:2.742439465642236E-4 23:1 30:1 36:1 39:1 41:1 46:1 49:1 66:1 152:1 274:1 333:1

|BV| 2:1 21:1.4421347148614654E-4 23:1 31:1 36:1 39:1 41:1 46:1 49:1 52:1 66:1 152:1 246:1 333:1 392:1 **|EV|**

Basic Commands

- Training and classification
  - **./svm_learn -t 5 -C T train.dat model**
  - **./svm_classify test.dat model**
- Learning with a vector sequence
  - **./svm_learn -t 5 -C V train.dat model**
- Learning with the sum of vector and kernel sequences
  - **./svm_learn -t 5 -C + train.dat model**

Custom Kernel

- Kernel.h
- **double custom_kernel(KERNEL_PARM *kernel_parm, DOC *a, DOC *b);**
- **if(a->num_of_trees && b->num_of_trees && a->forest_vec[i]!=NULL && b->forest_vec[i]!=NULL){**// Test if one the i-th tree of instance a and b is an empty tree

Custom Kernel: tree-kernel

- k1= // summation of tree kernels

```
tree_kernel(kernel_parm, a, b, i, i)/
```

Evaluate tree kernel between the two i-th trees.

```
sqrt(tree_kernel(kernel_parm, a, a, i, i) *
tree_kernel(kernel_parm, b, b, i, i));
```

Normalize respect to both i-th trees.

Custom Kernel: Polynomial kernel

- **`if(a->num_of_vectors && b->num_of_vectors && a->vectors[i]!=NULL && b->vectors[i]!=NULL){`** `Check if the i-th vectors are empty.`
- **`k2=   // summation of vectors basic_kernel(kernel_parm, a, b, i, i)/`** `Compute standard kernel (selected according to the "second_kernel" parameter).`

Custom Kernel: Polynomial kernel

- ```
  sqrt(
  basic_kernel(kernel_parm, a, a, i, i) *
  basic_kernel(kernel_parm, b, b, i, i)
  ); //normalize vectors
  ```
- `return k1+k2;`
  ```

Conclusions

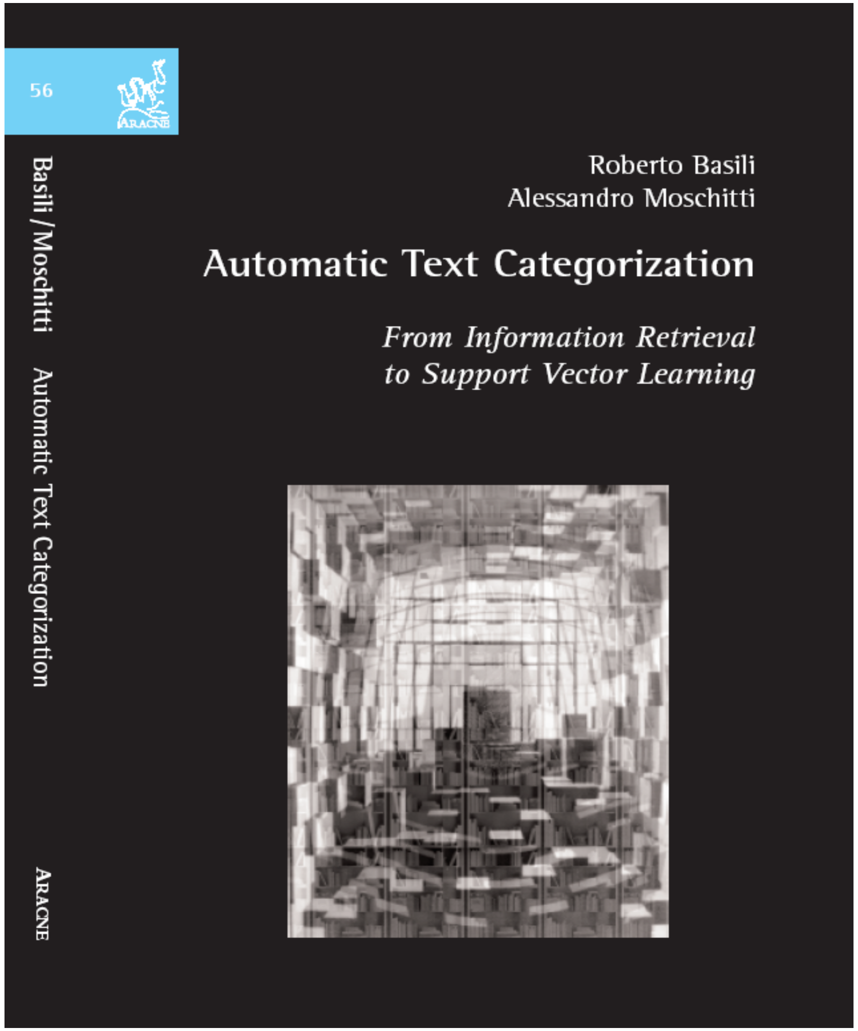
- Kernel methods and SVMs are useful tools to design language applications
- Kernel design still require some level of expertise
- Engineering approaches to tree kernels
  - Basic Combinations
  - Canonical Mappings, e.g.
    - Node Marking
  - Merging of kernels in more complex kernels
- State-of-the-art in SRL and QC
- An efficient tool to use them

Thank you

References

- Alessandro Moschitti, Silvia Quarteroni, Roberto Basili and Suresh Manandhar, *Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification*, Proceedings of the 45th Conference of the Association for Computational Linguistics (ACL), Prague, June 2007.
- Alessandro Moschitti and Fabio Massimo Zanzotto, *Fast and Effective Kernels for Relational Learning from Texts*, Proceedings of The 24th Annual International Conference on Machine Learning (ICML 2007), Corvallis, OR, USA.
- Daniele Pighin, Alessandro Moschitti and Roberto Basili, *RTV: Tree Kernels for Thematic Role Classification*, Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval-4), English Semantic Labeling, Prague, June 2007.
- Stephan Bloehdorn and Alessandro Moschitti, *Combined Syntactic and Semanitc Kernels for Text Classification*, to appear in the 29th European Conference on Information Retrieval (ECIR), April 2007, Rome, Italy.
- Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti, *Efficient Kernel-based Learning for Trees*, to appear in the IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Honolulu, Hawaii, 2007

An introductory book on SVMs, Kernel methods and Text Categorization



References

- Roberto Basili and Alessandro Moschitti, *Automatic Text Categorization: from Information Retrieval to Support Vector Learning*, Aracne editrice, Rome, Italy.
- Alessandro Moschitti, *Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees*. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
- Alessandro Moschitti, Daniele Pighin, and Roberto Basili, *Tree Kernel Engineering for Proposition Re-ranking*, In Proceedings of Mining and Learning with Graphs (MLG 2006), Workshop held with ECML/PKDD 2006, Berlin, Germany, 2006.
- Elisa Cilia, Alessandro Moschitti, Sergio Ammendola, and Roberto Basili, *Structured Kernels for Automatic Detection of Protein Active Sites*. In Proceedings of Mining and Learning with Graphs (MLG 2006), Workshop held with ECML/PKDD 2006, Berlin, Germany, 2006.

References

- Fabio Massimo Zanzotto and Alessandro Moschitti, *Automatic learning of textual entailments with cross-pair similarities*. In Proceedings of COLING-ACL, Sydney, Australia, 2006.
- Alessandro Moschitti, *Making tree kernels practical for natural language learning*. In Proceedings of the Eleventh International Conference on European Association for Computational Linguistics, Trento, Italy, 2006.
- Alessandro Moschitti, Daniele Pighin and Roberto Basili. *Semantic Role Labeling via Tree Kernel joint inference*. In Proceedings of the 10th Conference on Computational Natural Language Learning, New York, USA, 2006.
- Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin and Roberto Basili, *Semantic Tree Kernels to classify Predicate Argument Structures*. In Proceedings of the the 17th European Conference on Artificial Intelligence, Riva del Garda, Italy, 2006.

References

- Alessandro Moschitti and Roberto Basili, *A Tree Kernel approach to Question and Answer Classification in Question Answering Systems*. In Proceedings of the Conference on Language Resources and Evaluation, Genova, Italy, 2006.
- Ana-Maria Giuglea and Alessandro Moschitti, *Semantic Role Labeling via FrameNet, VerbNet and PropBank*. In Proceedings of the Joint 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Sydney, Australia, 2006.
- Roberto Basili, Marco Cammisa and Alessandro Moschitti, *Effective use of wordnet semantics via kernel-based learning*. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005), Ann Arbor(MI), USA, 2005

References

- Alessandro Moschitti, Ana-Maria Giuglea, Bonaventura Coppola and Roberto Basili. *Hierarchical Semantic Role Labeling*. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005 shared task), Ann Arbor(MI), USA, 2005.
- Roberto Basili, Marco Cammisa and Alessandro Moschitti, *A Semantic Kernel to classify texts with very few training examples*. In Proceedings of the Workshop on Learning in Web Search, at the 22nd International Conference on Machine Learning (ICML 2005), Bonn, Germany, 2005.
- Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin and Roberto Basili. *Engineering of Syntactic Features for Shallow Semantic Parsing*. In Proceedings of the ACL05 Workshop on Feature Engineering for Machine Learning in Natural Language Processing, Ann Arbor (MI), USA, 2005.

References

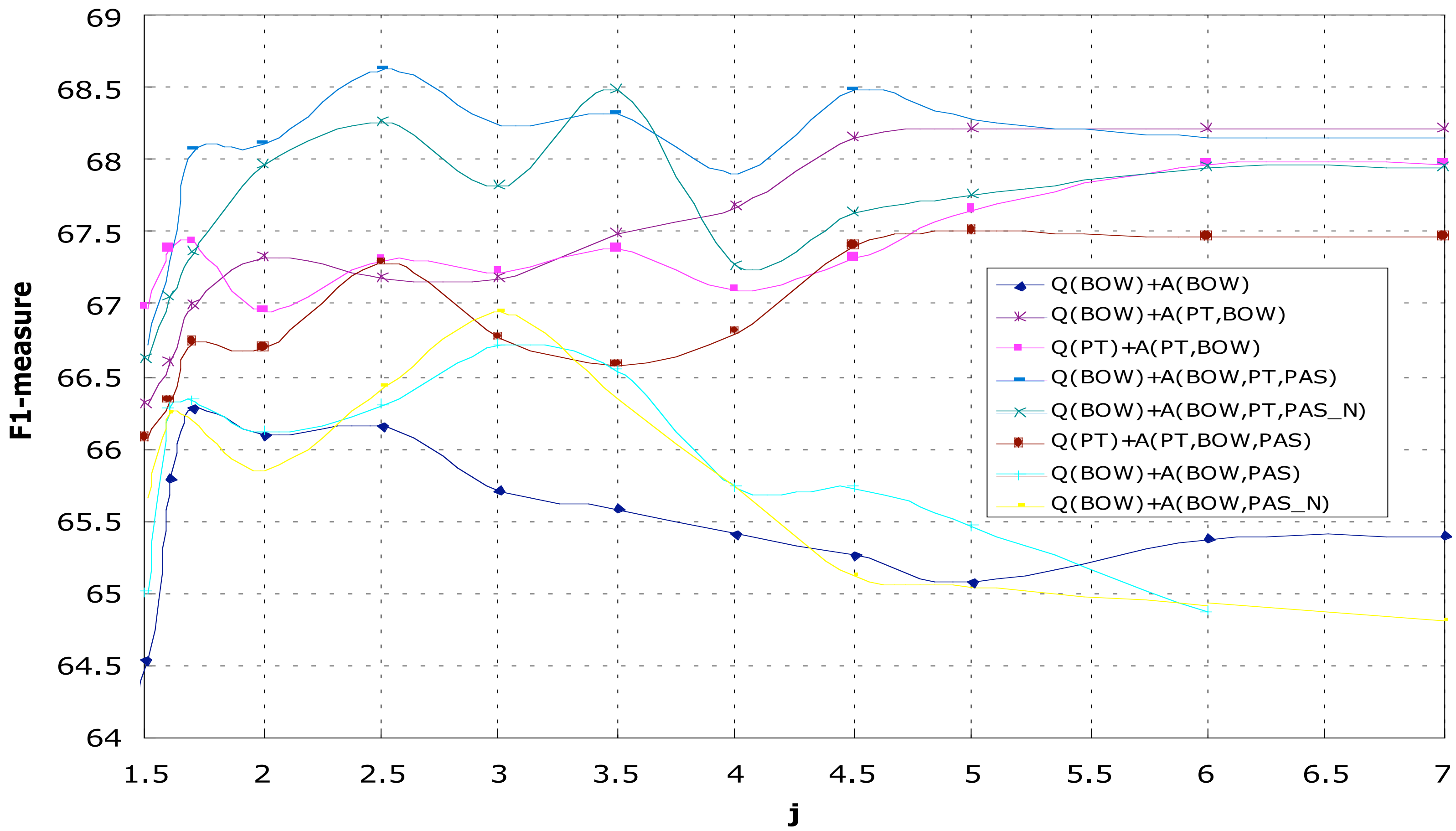
- Alessandro Moschitti, *A study on Convolution Kernel for Shallow Semantic Parsing*. In proceedings of ACL-2004, Spain, 2004.
- Alessandro Moschitti and Cosmin Adrian Bejan, *A Semantic Kernel for Predicate Argument Classification.* In proceedings of the CoNLL-2004, Boston, MA, USA, 2004.
- M. Collins and N. Duffy, *New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron*. In ACL02, 2002.
- S.V.N. Vishwanathan and A.J. Smola. *Fast kernels on strings and trees*. In Proceedings of Neural Information Processing Systems, 2002.

References

- *AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods)* N. Cristianini and J. Shawe-Taylor Cambridge University Press
- Xavier Carreras and Llu´ıs M`arquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *proceedings of CoNLL'05*.
- Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *to appear in Machine Learning Journal*.

```
function Evaluate_Pair_Set(Tree T1, T2) returns NODE_PAIR_SET;
LIST L1,L2;
NODE_PAIR_SET Np;
begin
    L1 = T1.ordered_list;
    L2 = T2.ordered_list; /*the lists were sorted at loading time*/
    n1 = extract(L1); /*get the head element and*/
    n2 = extract(L2); /*remove it from the list*/
    while (n1 and n2 are not NULL)
        if (production_of(n1) > production_of(n2))
            then n2 = extract(L2);
            else if (production_of(n1) < production_of(n2))
                then n1 = extract(L1);
                else
                    while (production_of(n1) == production_of(n2))
                        while (production_of(n1) == production_of(n2))
                            add(⟨n1, n2⟩, Np);
                            n2=get_next_elem(L2); /*get the head element
                            and move the pointer to the next element*/
                        end
                        n1 = extract(L1);
                        reset(L2); /*set the pointer at the first element*/
                    end
    end
    return Np ;
end
```

The Impact of SSTK in Answer Classification



Mercer's conditions (1)

Def. B.11 *Eigen Values*
Given a matrix $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$, *an egeinvalue* λ *and an egeinvector* $\vec{x} \in \mathbb{R}^n - \{\vec{0}\}$ *are such that*

$$\mathbf{A}\vec{x} = \lambda\vec{x}$$

Def. B.12 *Symmetric Matrix*
A square matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is symmetric iff* $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ *for* $i \neq j$ $i = 1, .., m$ *and* $j = 1, .., n$, *i.e. iff* $\mathbf{A} = \mathbf{A}'$.

Def. B.13 *Positive (Semi-) definite Matrix*
A square matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).*

Mercer's conditions (2)

Proposition 2.27 *(Mercer's conditions)*
Let X be a finite input space with $K(\vec{x},\vec{z})$ a symmetric function on X. Then $K(\vec{x},\vec{z})$ is a kernel function if and only if the matrix

$$k(\vec{x},\vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

is positive semi-definite (has non-negative eigenvalues).

- If the Gram matrix: $G = k(\vec{x}_i, \vec{x}_j)$
  is positive semi-definite there is a mapping ϕ that produces the target kernel function

The lexical semantic kernel is not always a kernel

- It may not be a kernel so we can use ***M´·M***, where ***M*** is the initial similarity matrix

Proposition B.14 *Let* $\boldsymbol{A}$ *be a symmetric matrix. Then* A *is positive (semi-) definite iff for any vector* $\vec{x} \neq 0$

$$\vec{x}' \boldsymbol{A} \vec{x} > \lambda \vec{x} \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $\boldsymbol{A}$ in $M'M$, then $\boldsymbol{A}$ is semi-definite positive matrix as

$$\vec{x}' \boldsymbol{A} \vec{x} = \vec{x}' \boldsymbol{M}' \boldsymbol{M} \vec{x} = (\boldsymbol{M} \vec{x})'(\boldsymbol{M} \vec{x}) = \boldsymbol{M} \vec{x} \cdot \boldsymbol{M} \vec{x} = ||\boldsymbol{M} \vec{x}||^2 \geq 0.$$

Efficient Evaluation (1)

- In [Taylor and Cristianini, 2004 book], sequence kernels with weighted gaps are factorized with respect to different subsequence sizes.
- We treat children as sequences and apply the same theory

$$\Delta(n_1, n_2) = \mu\left(\lambda^2 + \sum_{p=1}^{lm} \Delta_p(c_{n_1}, c_{n_2})\right),$$

Given the two child sequences $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ (a and b are the last children), $\Delta_p(s_1 a, s_2 b) =$

$$\Delta(a, b) \times \underbrace{\sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r])}_{D_p}$$