COMPUTATIONAL MODELS FOR DATA ANALYSIS

Kernel Methods

Alessandra Giordani

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

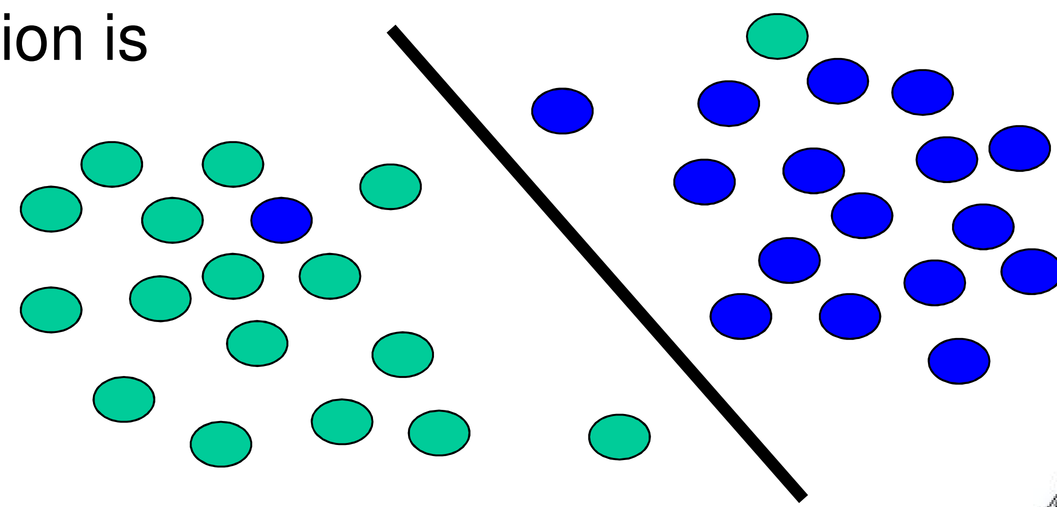
Linear Classifier

- The equation of a hyperplane is

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

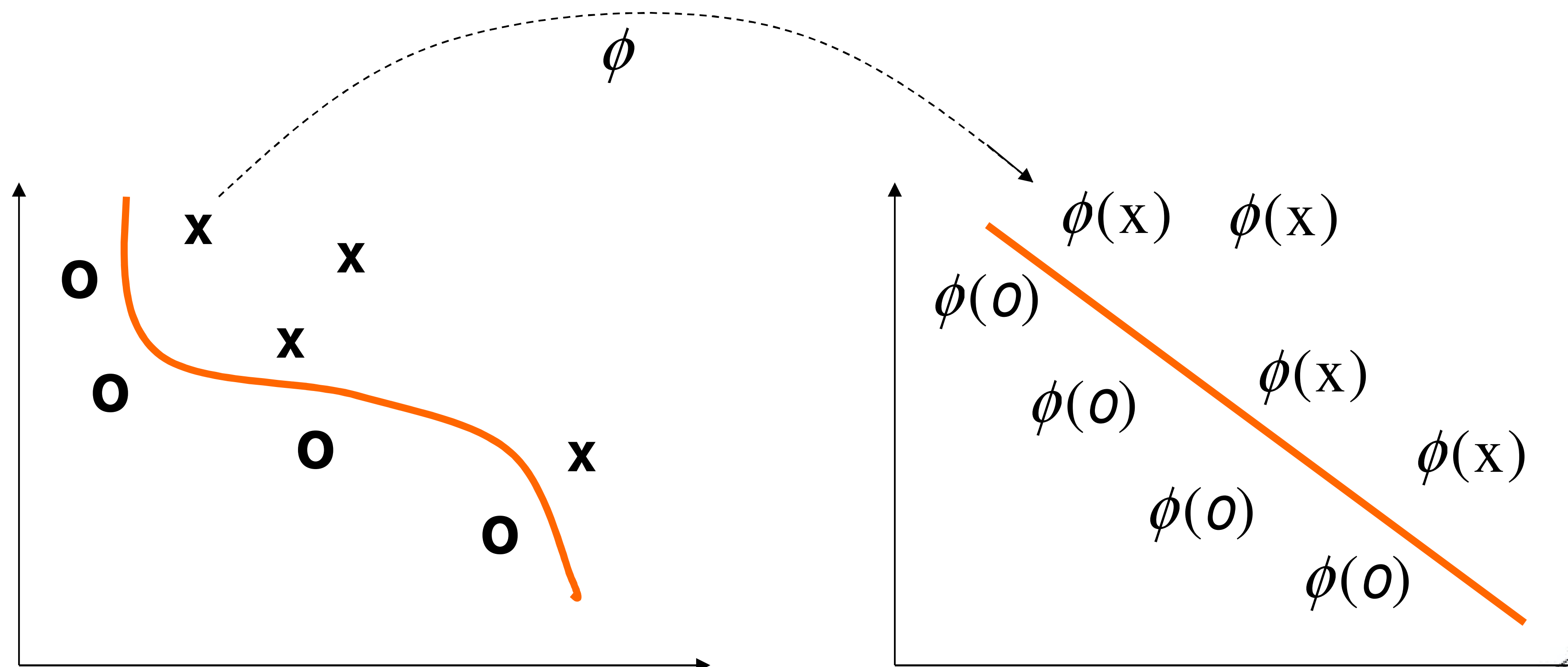
- $\vec{x}$ is the vector representing the classifying example
- $\vec{w}$ is the gradient of the hyperplane
- The classification function is

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



The main idea of Kernel Functions

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



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)

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

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

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

- 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, $(\vec{w}_k, b_k)$

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

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

The Kernel Gram Matrix

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

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

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

Valid Kernels

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

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

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

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

Mercer's condition

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

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 $\square_s$ 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:

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

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

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

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

- More expressive, as $(x_1 x_2)$ encodes

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

- We can smartly compute the scalar product as

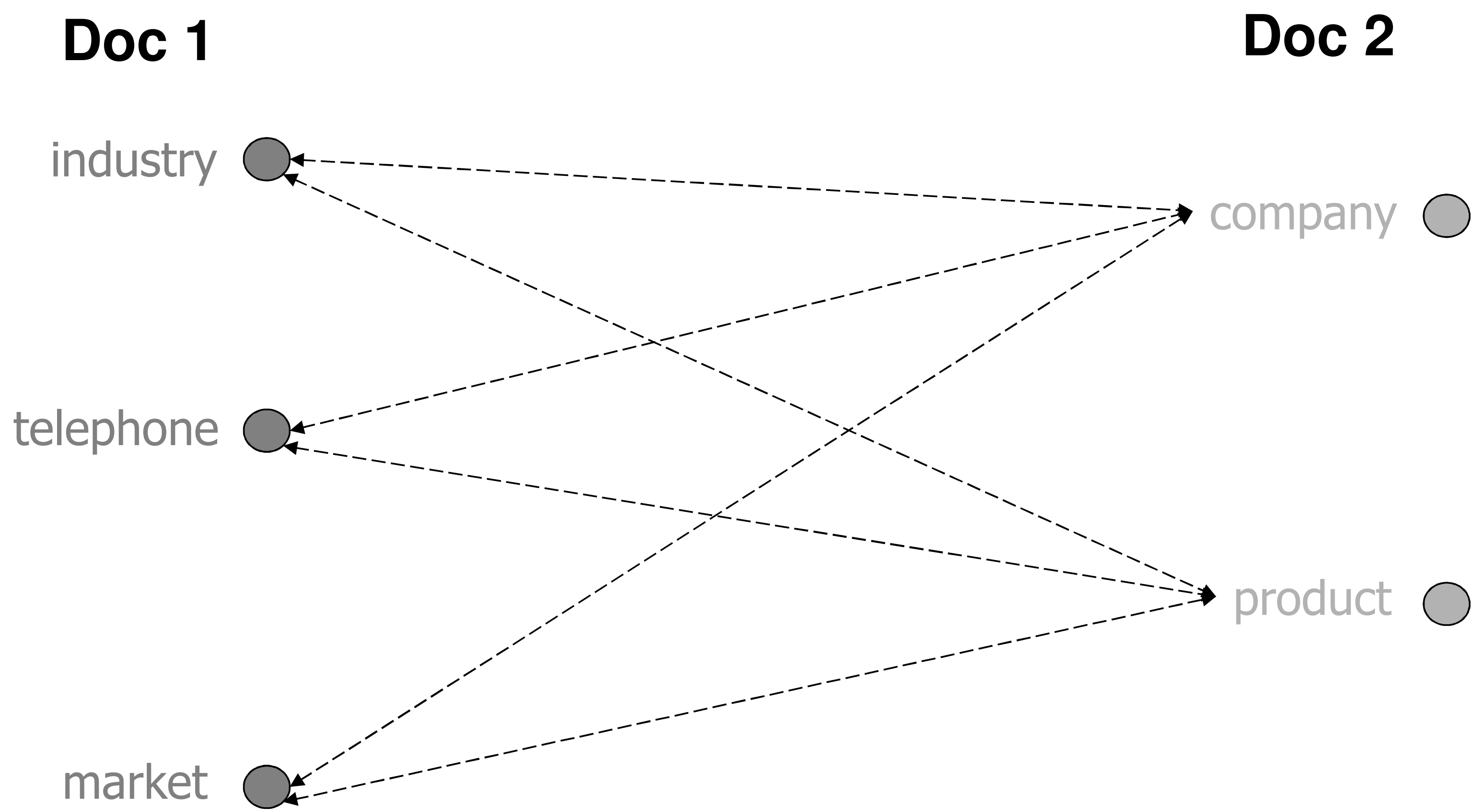
$$\Phi(\vec{x}) \cdot \Phi(\vec{z}) =$$

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

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 + 2x_1 z_1 + 2x_2 z_2 + 1 =$$

$$= (x_1 z_1 + x_2 z_2 + 1)^2 = (\vec{x} \cdot \vec{z} + 1)^2 = K_{Poly}(\vec{x}, \vec{z})$$

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 $Bank$.

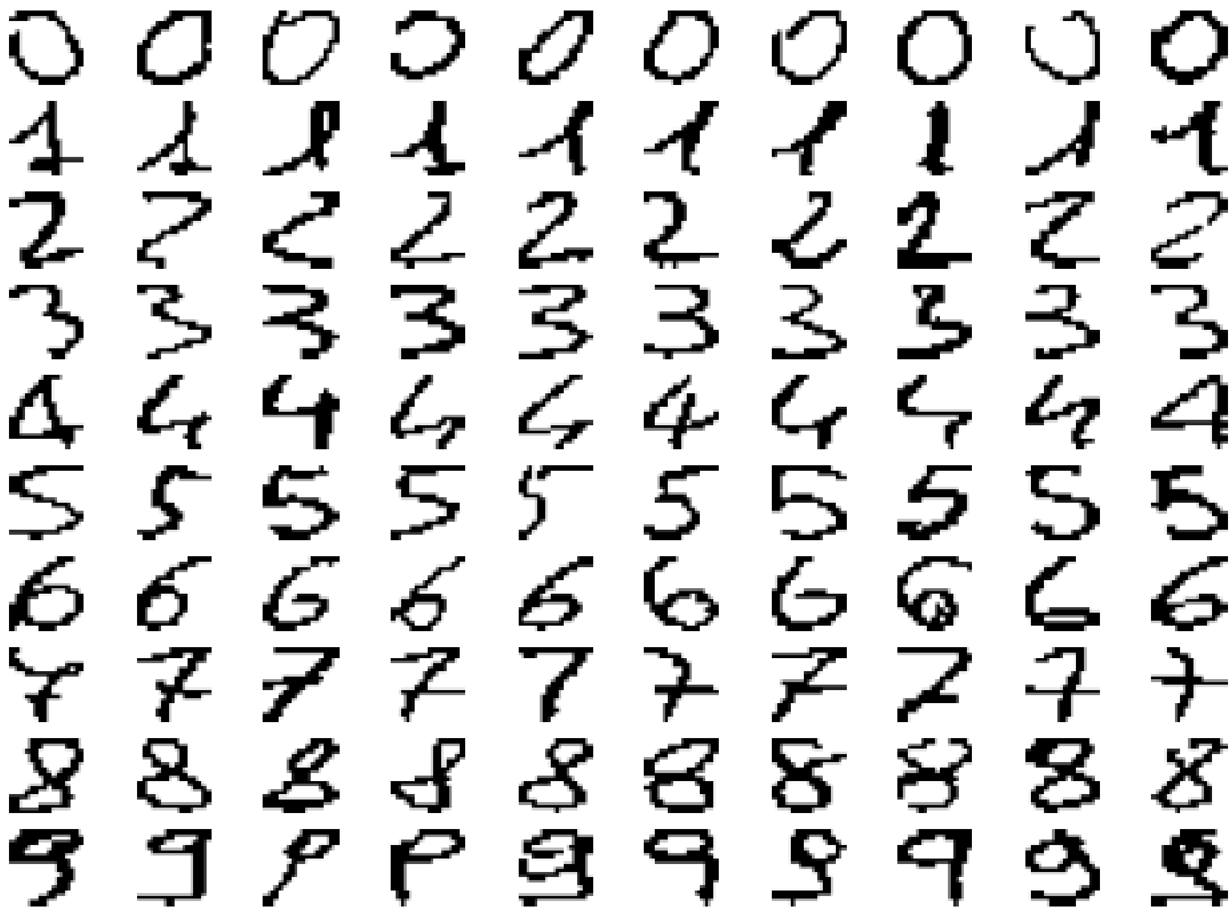
R, a, n, k, Ra, Ran, Rank, an, ank, nk, Rn, Rnk, Rk and ak are the substrings of $Rank$.

An example of string kernel computation

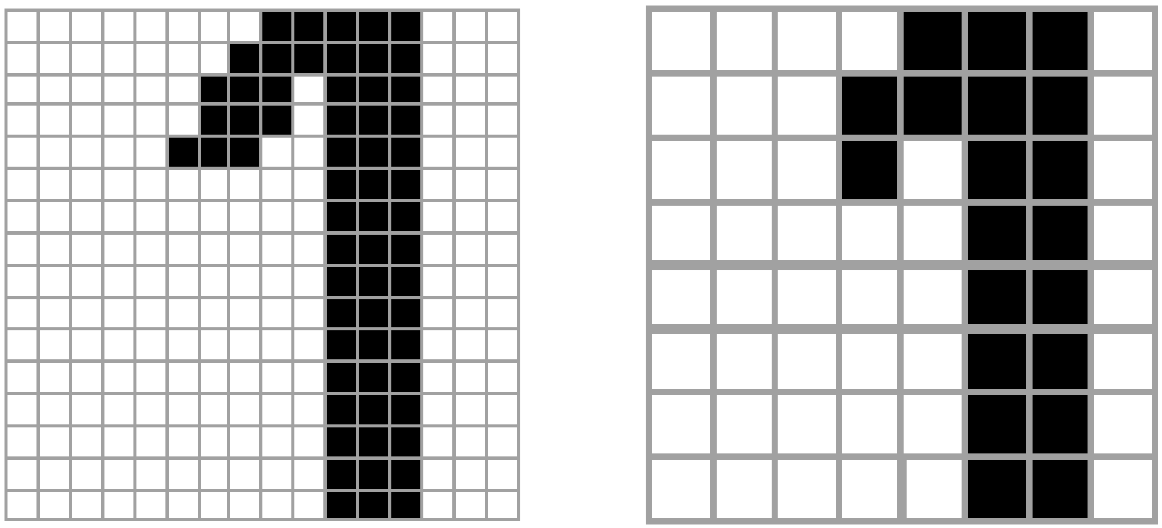
- $\phi_{\mathsf{a}}(\mathsf{Bank}) = \phi_{\mathsf{a}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_{\mathsf{n}}(\mathsf{Bank}) = \phi_{\mathsf{n}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_{\mathsf{k}}(\mathsf{Bank}) = \phi_{\mathsf{k}}(\mathsf{Rank})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

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

String Kernels for OCR



Pixel Representation



(a) (b)

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

Sequence of bits

L1 000**111**00
. 00**1111**00
. 00**1**0**11**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

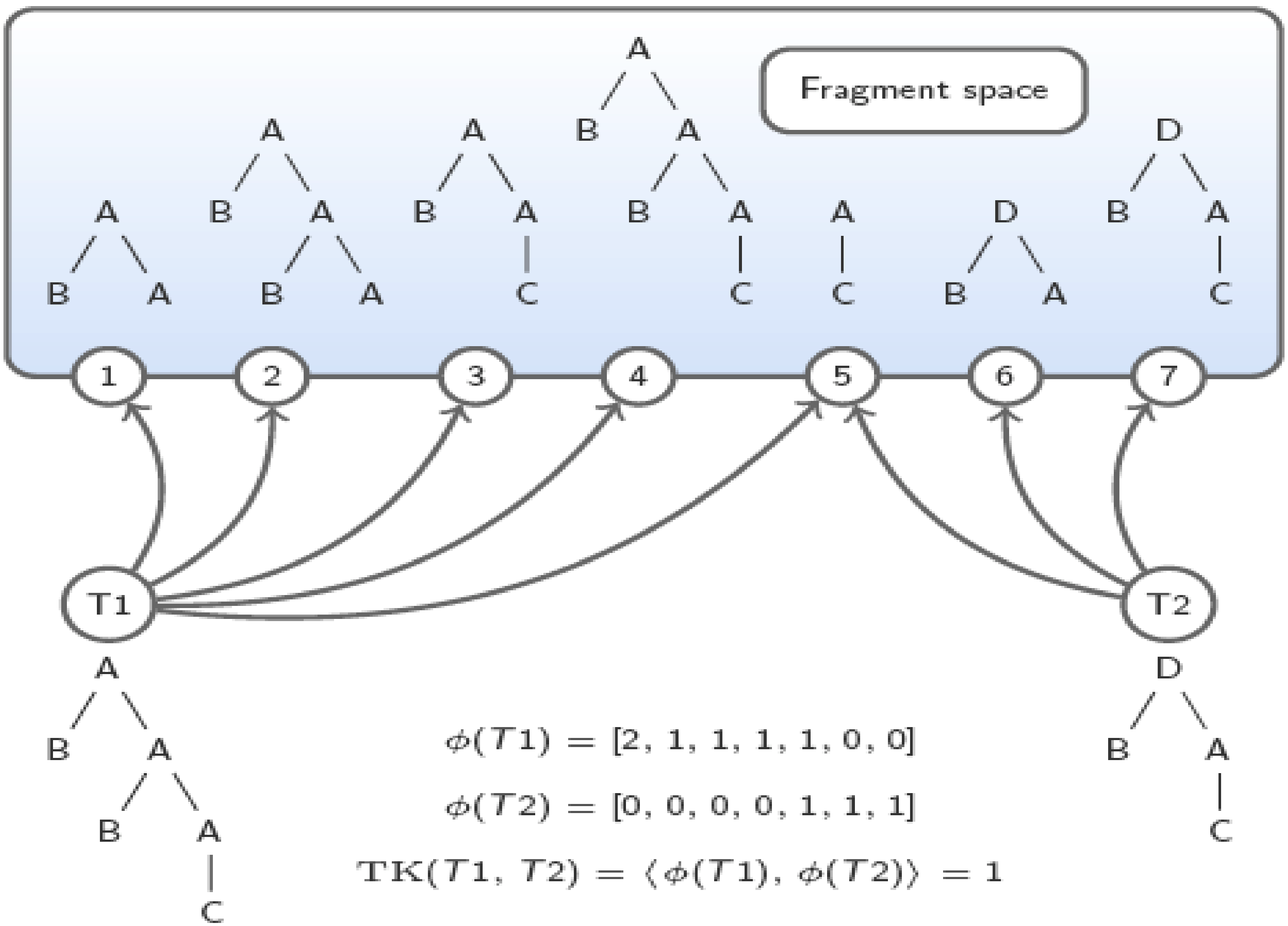
- Using columns+rows+diagonals

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



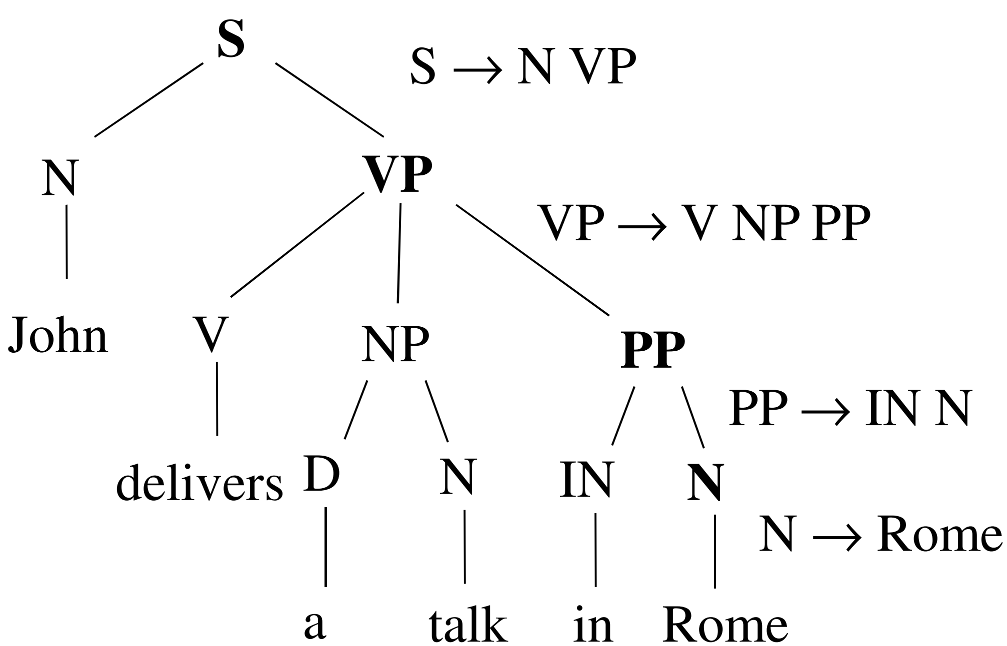
$$\phi(T1) = [2, 1, 1, 1, 1, 0, 0]$$

$$\phi(T2) = [0, 0, 0, 0, 1, 1, 1]$$

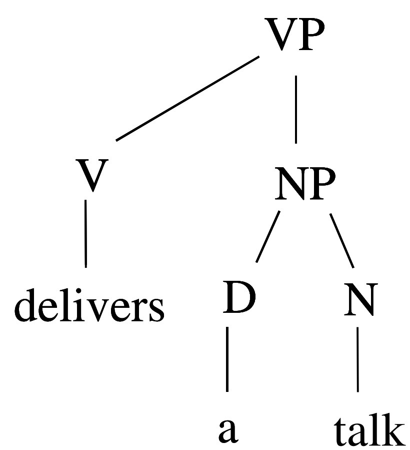
$$\mathrm{TK}(T1, T2) = \langle \phi(T1), \phi(T2) \rangle = 1$$

Example of a syntactic parse tree

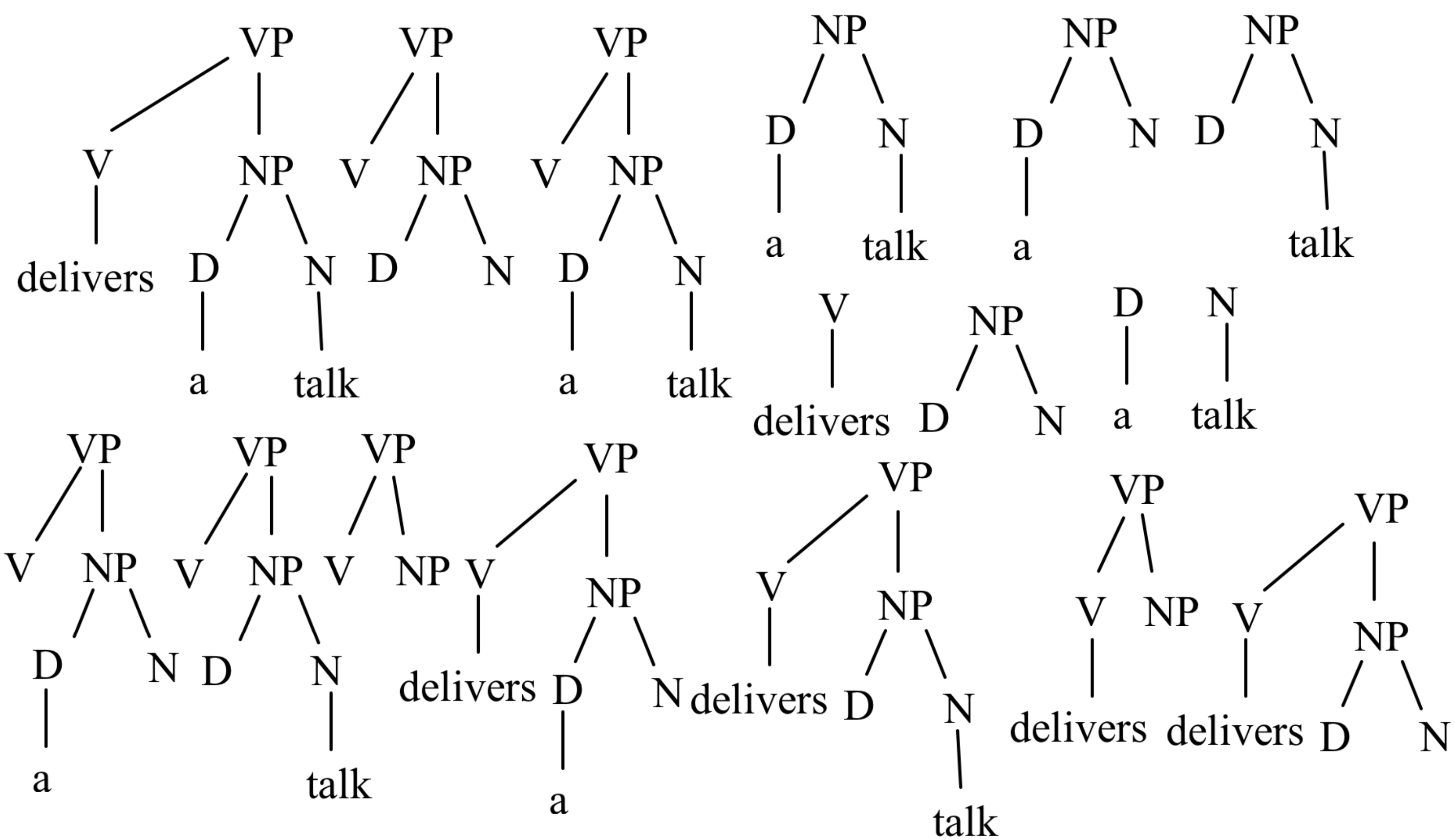
- "John delivers a talk in Rome"



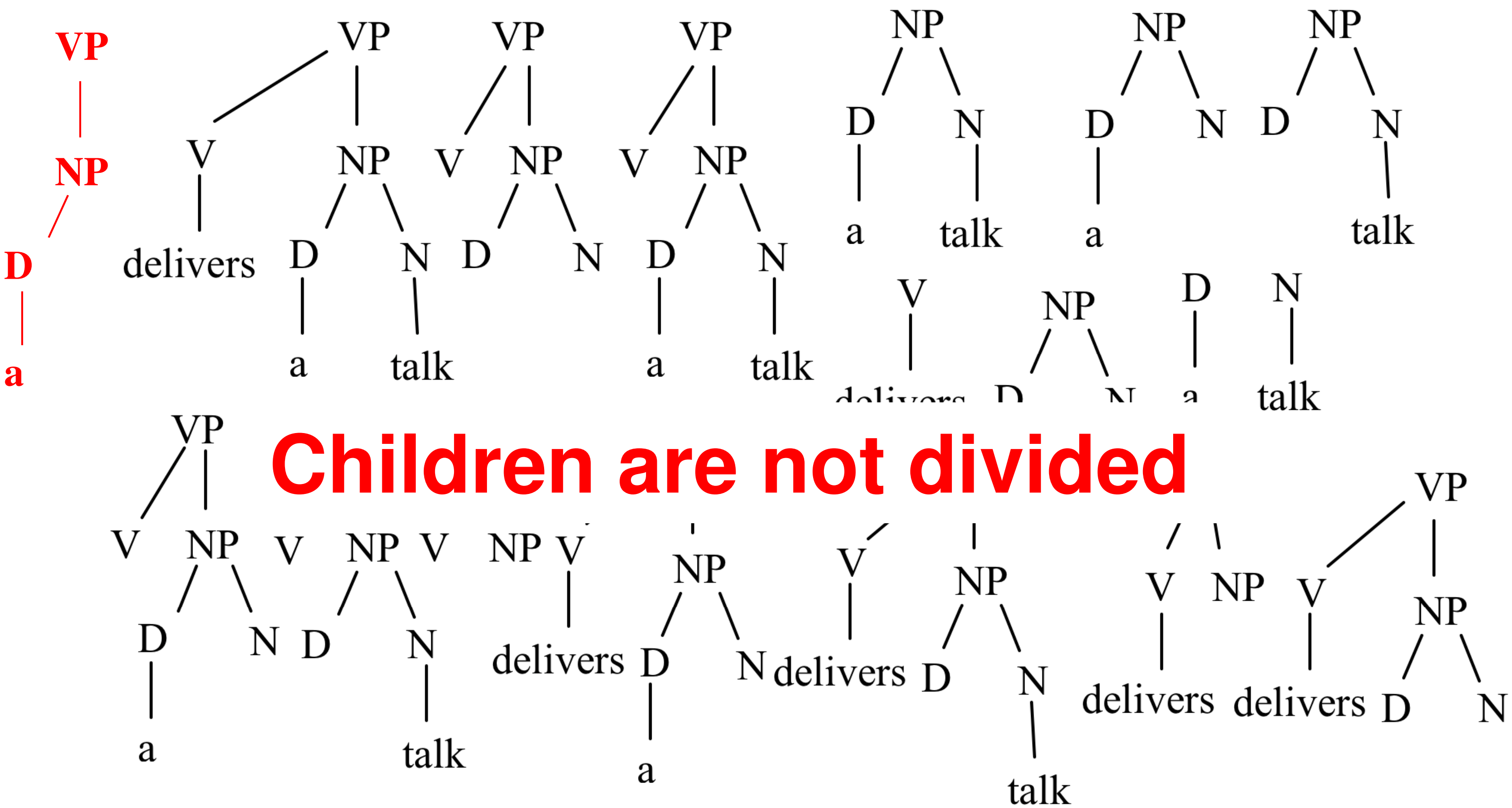
The Syntactic Tree Kernel (STK)
[Collins and Duffy, 2002]



The overall fragment set



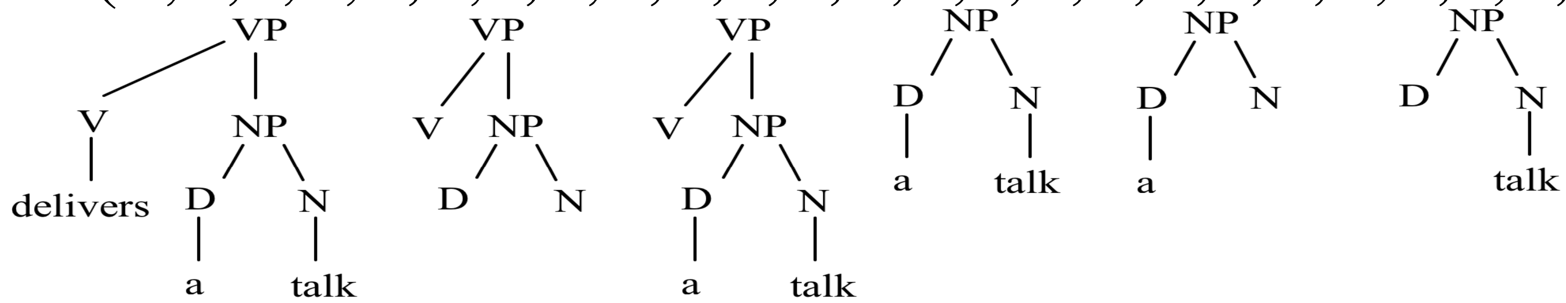
The overall fragment set



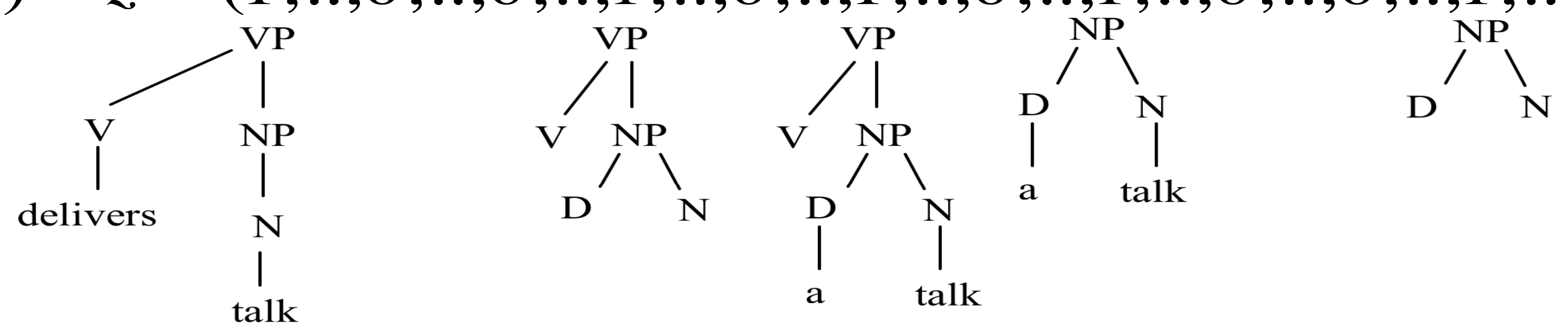
Children are not divided

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

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

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

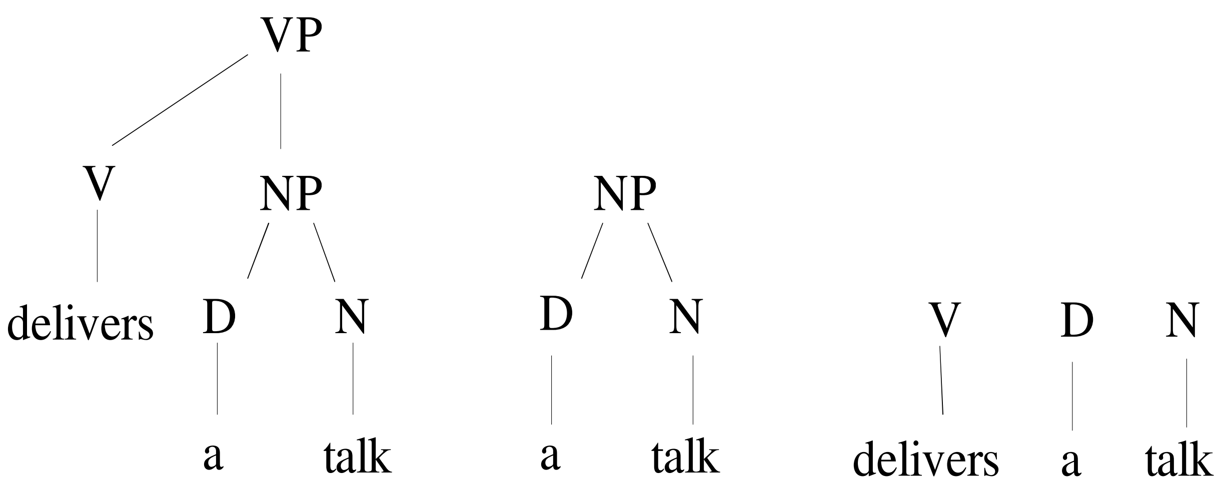
- [Collins and Duffy, ACL 2002] evaluate Δ in $O(n^2)$:

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

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

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

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



Evaluation

- Given the equation for STK

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

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

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

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?

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

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