MACHINE LEARNING

Kernel Methods

Alessandro Moschitti

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

Outline (1)

PART I: Theory

- Motivations
- Kernel Based Machines
- Kernel Methods
  - Kernel Trick
  - Mercer's conditions
  - Kernel operators
- Basic Kernels
  - Linear Kernel
  - Polynomial Kernel
  - Lexical kernel
  - String Kernel

Outline (2)

- Tree kernels
  - subtree, subset tree and partial tree kernels
- **PART II: Kernel engineering for Natural Language Applications**
- Object Transformation and Kernel Combinations:

  The Semantic Role Labeling case
  - Node marking
  - Shallow Semantic Tree Kernels
  - Experiments

Outline (2)

- Merging of Kernels and Kernel Combinations: Question/Answer Classification
  - The Syntactic/Semantic Tree Kernel
  - Advanced Kernel combinations
  - Experiments
- Example of Custom Kernel in **SVM-LIGHT-TK**
- Conclusions

Kernels for Natural Language Processing (1)

- Typical tasks: from text categorization to syntactic and semantic parsing
- Feature design for NLP applications:
  - complex and difficult phase, e.g., structural feature representation:
  - deep knowledge and intuitions are required
  - design problems when the phenomenon is described by many features

Kernels for Natural Language Processing (2)

- Kernel methods alleviate such problems
  - Structures represented in terms of substructures
  - High dimensional feature spaces
  - Implicit and abstract feature spaces
- Generate high number of features
  - Support Vector Machines “select” the relevant features
  - Automatic Feature engineering side-effect

Kernels Engineering

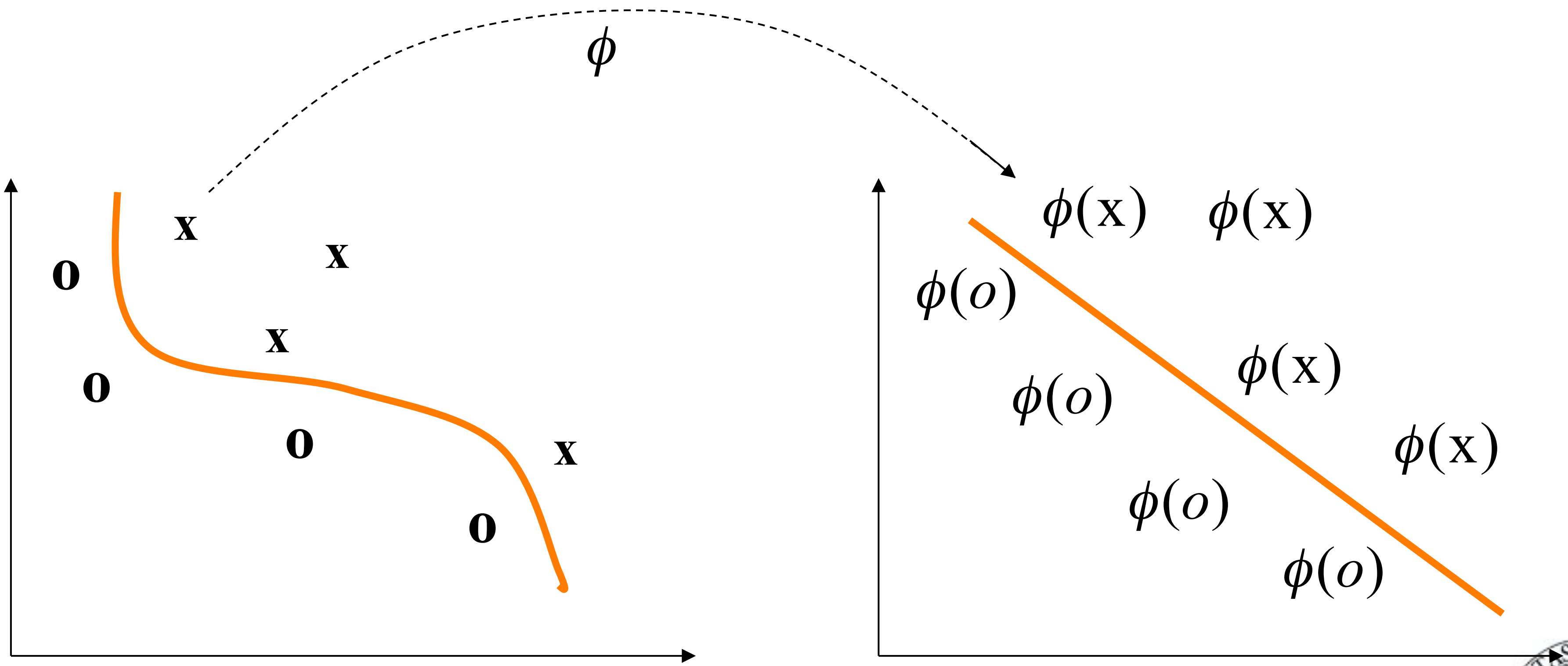
- Kernel design still requires noticeable creativity and expertise.
- An engineering approach is needed:
  - starting from basic and well known kernel functions
  - *basic combinations*
  - *canonical mappings*, e.g. object transformations
  - *merging of kernels*

Theory

- Kernel Trick
- Kernel Based Machines
- Basic Kernel Properties
- Kernel Types

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 $f(m_1, m_2, r) < f_a$

Mapping Example

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

- 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

Classification function in the dual space

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

- Note that data only appears in the scalar product
- The Matrix $G=\left(\langle \mathbf{x}_i\cdot\mathbf{x}_j\rangle\right)_{i,j=1}^{l}$ is called Gram matrix

Dual Perceptron algorithm and Kernel functions

- We can rewrite the classification function as

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

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

- As well as the updating function

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

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

Kernels in Support Vector Machines

- In Soft Margin SVMs we maximize:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \frac{1}{2C} \vec{\alpha} \cdot \vec{\alpha} - \frac{1}{C} \vec{\alpha} \cdot \vec{\alpha}$$

- 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.*

- Kernel are the product of mapping functions such as

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

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

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

$$\begin{aligned}
&\Phi(\vec{x}) \cdot \Phi(\vec{z}) = \\
&= (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) = \\
&= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 = \\
&= (x_1z_1 + x_2z_2 + 1)^2 = (\vec{x} \cdot \vec{z} + 1)^2 = K_{Poly}(\vec{x}, \vec{z})
\end{aligned}$$

Using character sequences

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

bank ank bnk bk b

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

rank ank rnk rk r

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

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

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

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

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

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

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

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

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

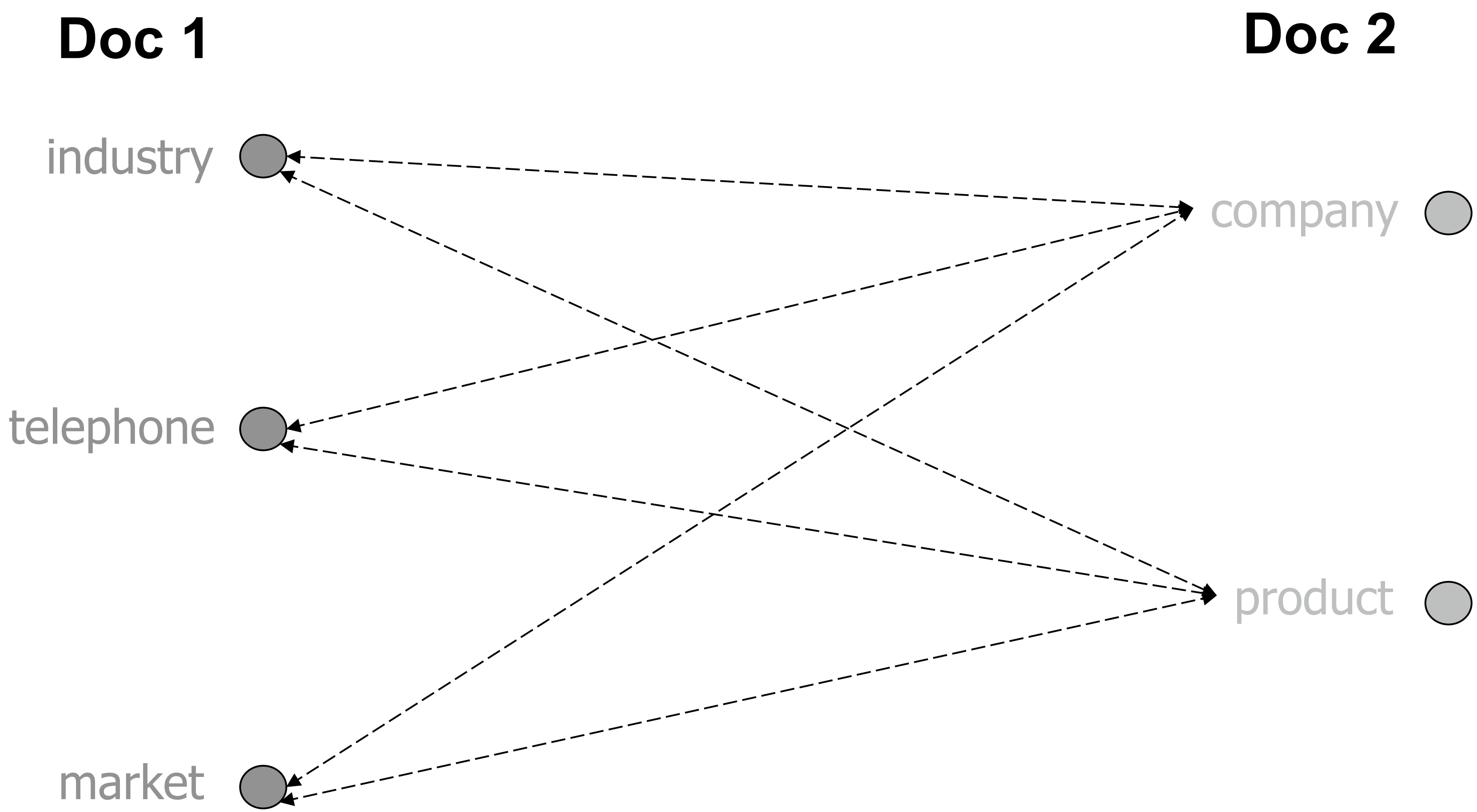
- $\phi_{\mathsf{a}}(\mathsf{Bank}) = \phi_{\mathsf{a}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_{\mathsf{n}}(\mathsf{Bank}) = \phi_{\mathsf{n}}(\mathsf{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_{\mathsf{k}}(\mathsf{Bank}) = \phi_{\mathsf{k}}(\mathsf{Rank})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

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

Intuition behind efficient evaluation

- Dynamic programming
- Given two strings of size *n* and *m,*
  - $NS(n,m,\rho) = NS(n,m,\rho-1)+NS(n-1,\mathrm{m},\rho)+NS(n,m-1,\rho) - NS(n-1,m-1,\rho)$

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 poor

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* $\vec{x} \neq 0$

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

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

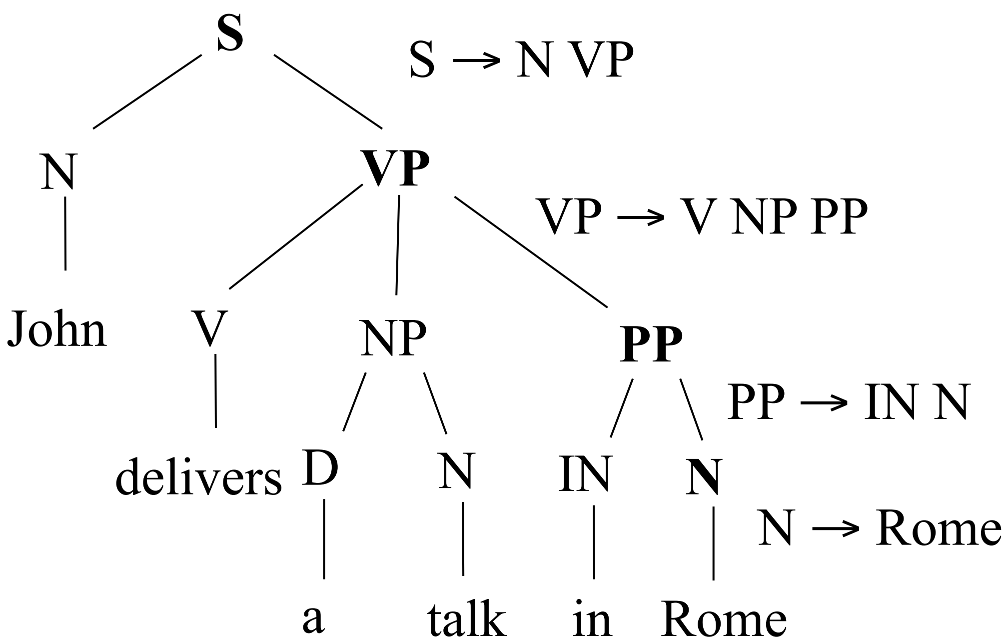
$$\vec{x}' \boldsymbol{A} \vec{x} = \vec{x}' \boldsymbol{M}' \boldsymbol{M} \vec{x} = (\boldsymbol{M} \vec{x})' (\boldsymbol{M} \vec{x}) = \boldsymbol{M} \vec{x} \cdot \boldsymbol{M} \vec{x} = ||\boldsymbol{M} \vec{x}||^2 \geq 0.$$

Tree kernels

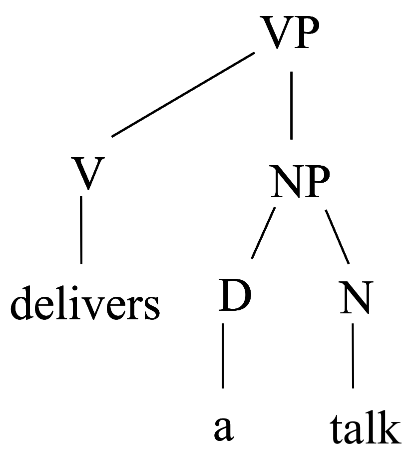
- Subtree, Subset Tree, Partial Tree kernels
- Efficient computation

Example of a parse tree

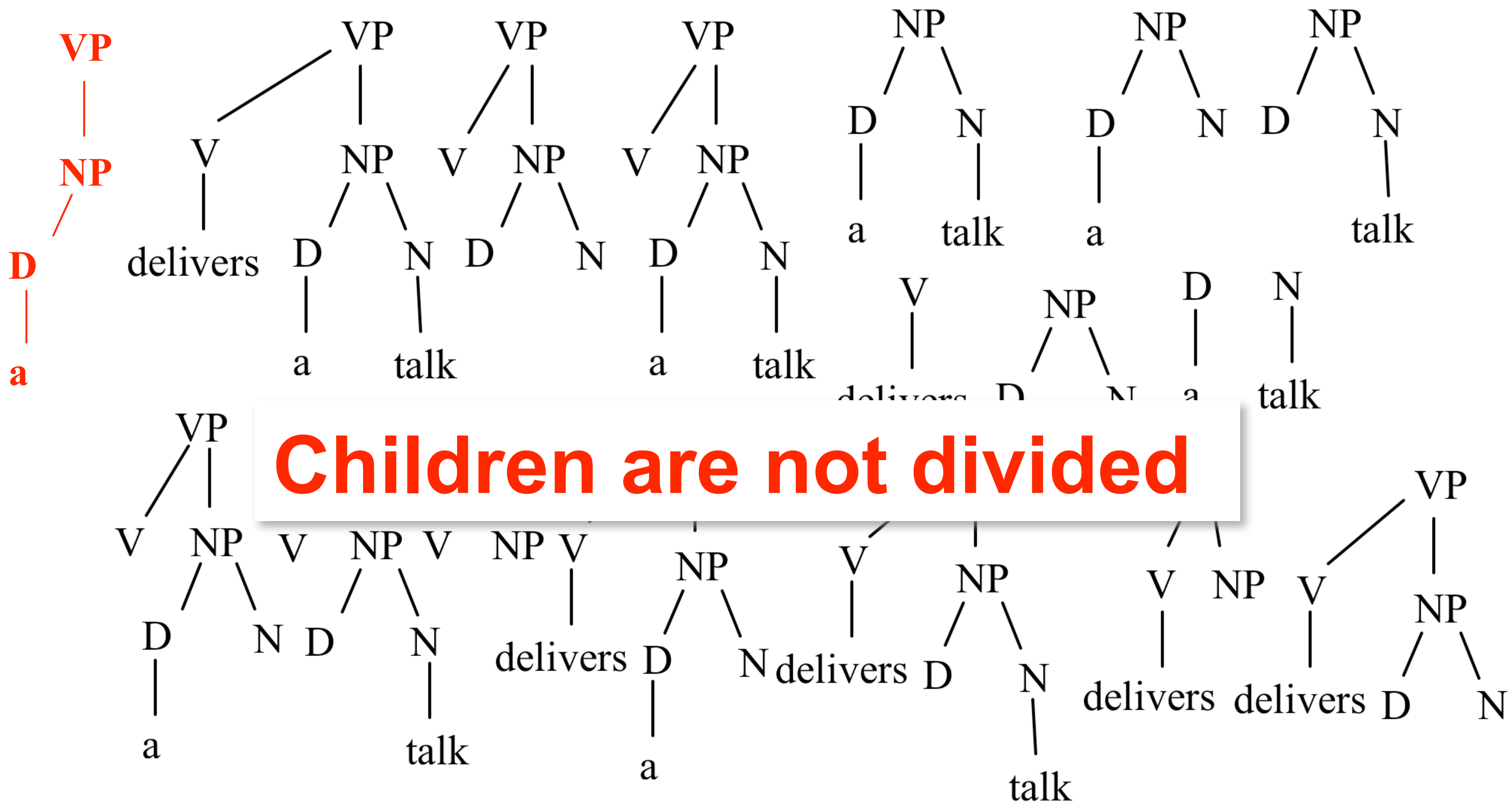
- “John delivers a talk in Rome”



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



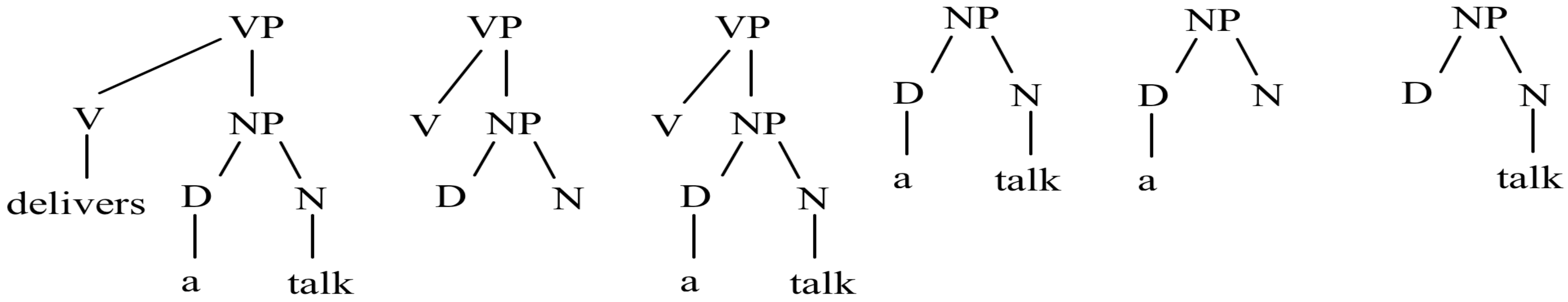
The overall fragment set



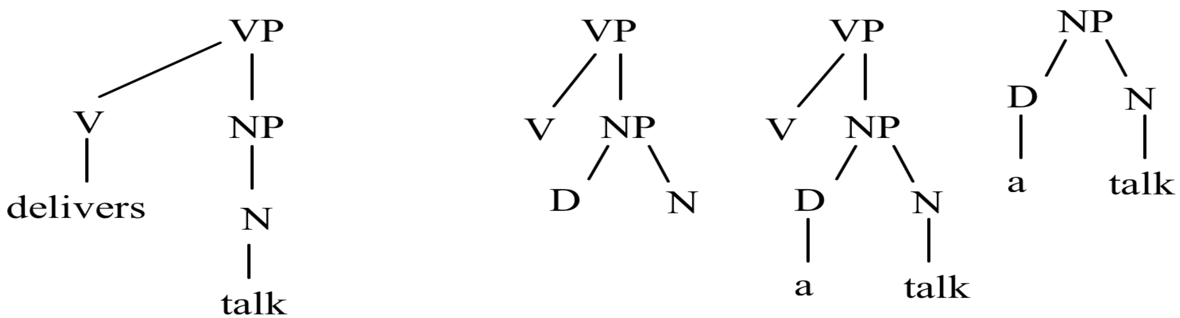
Children are not divided

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 $O(n^2)$:

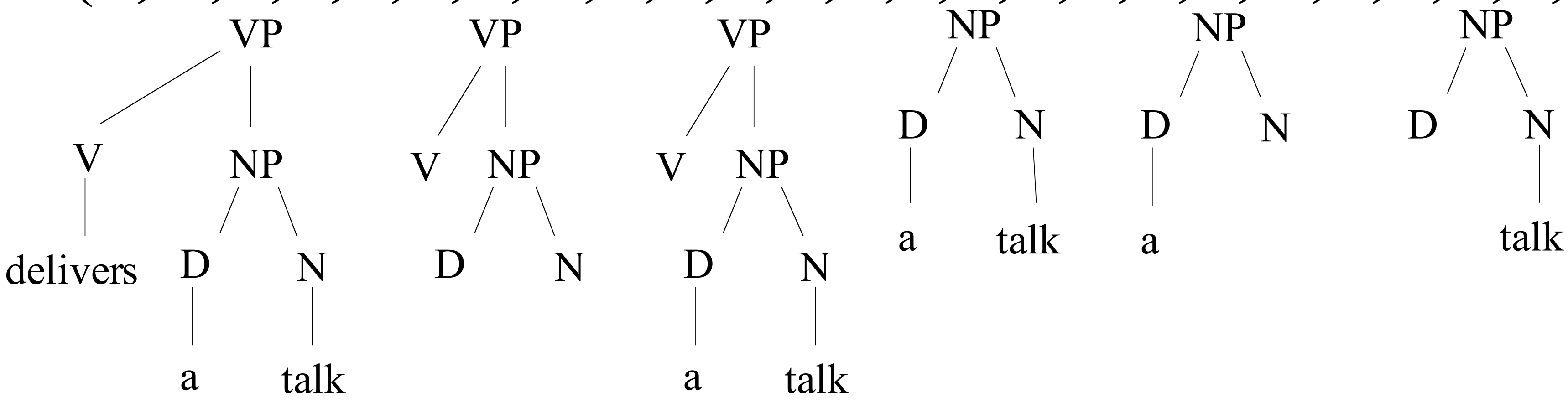
$$\Delta(n_x, n_z) = 0, \text{ if the productions are different else}$$

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

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

Explicit kernel space

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



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

Implicit Representation

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

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

Implicit Representation

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

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

- [Collins and Duffy, ACL 2002] evaluate Δ in O(n²):

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

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

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

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

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

Fast SST Evaluation [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

Observations

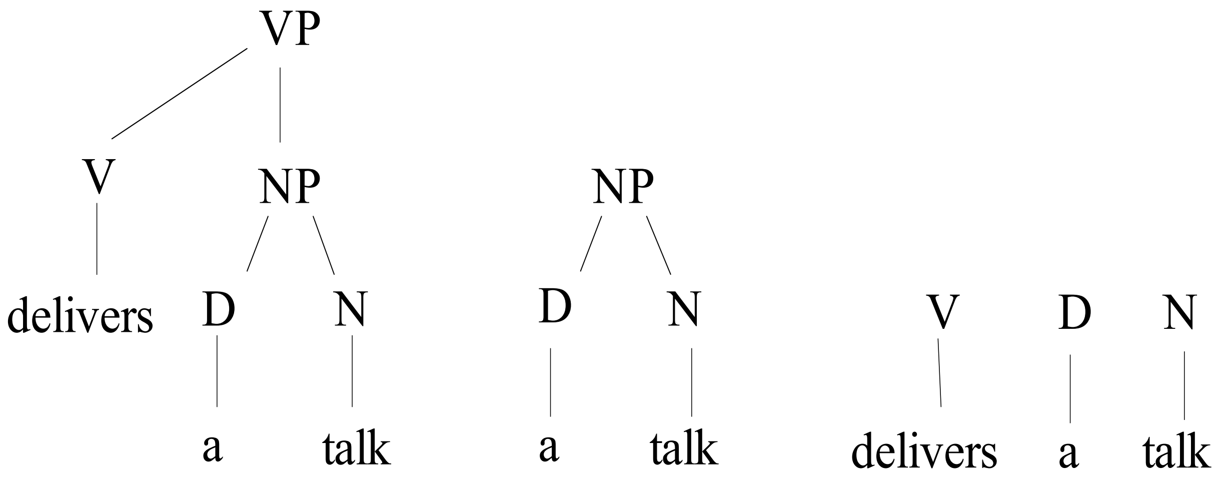
- We order the production rules used in T_x and T_z, at loading time
- At learning time we may evaluate NP in $|T_x| + |T_z|$ *running time*
- If T_x and T_z are generated by only one production rule $\Rightarrow O(|T_x| \times |T_z|)\ldots$

Observations

- 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!!!!***

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



Evaluation

- Given the equation for the SST kernel

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

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

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

Evaluation

- Given the equation for the SST kernel

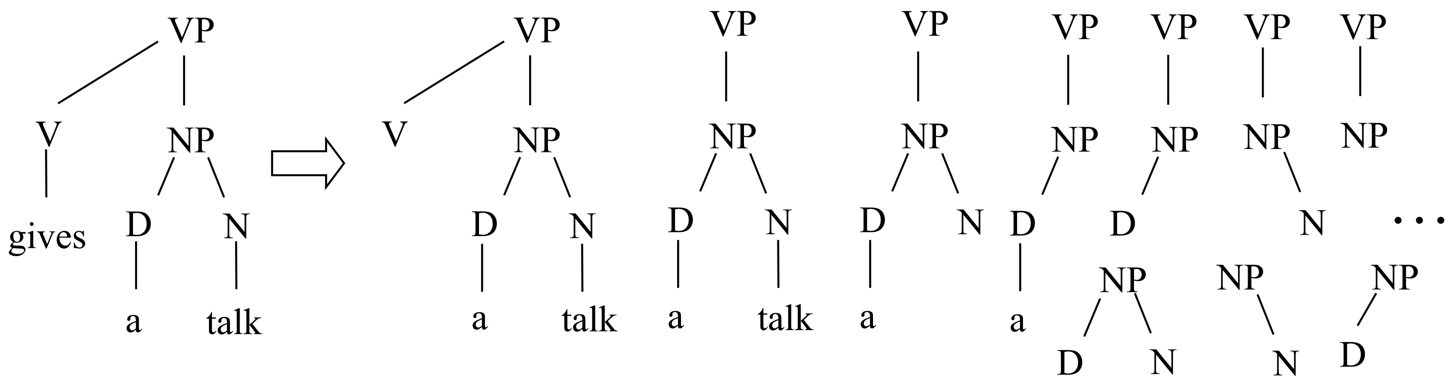
$$\Delta(n_x, n_z) = 0, \ \text{if the productions are different else}$$

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

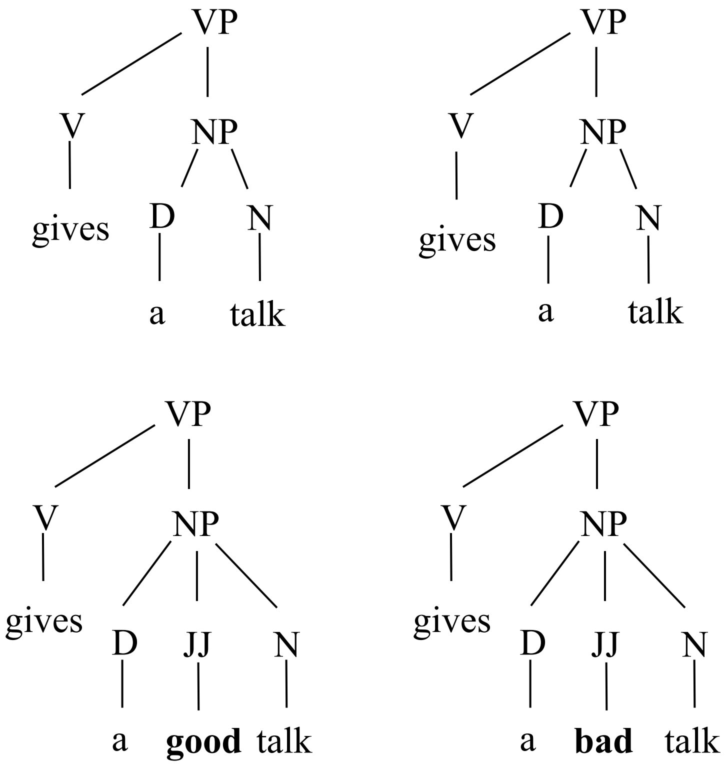
$$\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} \Delta(ch(n_x, j), ch(n_z, j))$$

Labeled Ordered Tree Kernel

- SST 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]



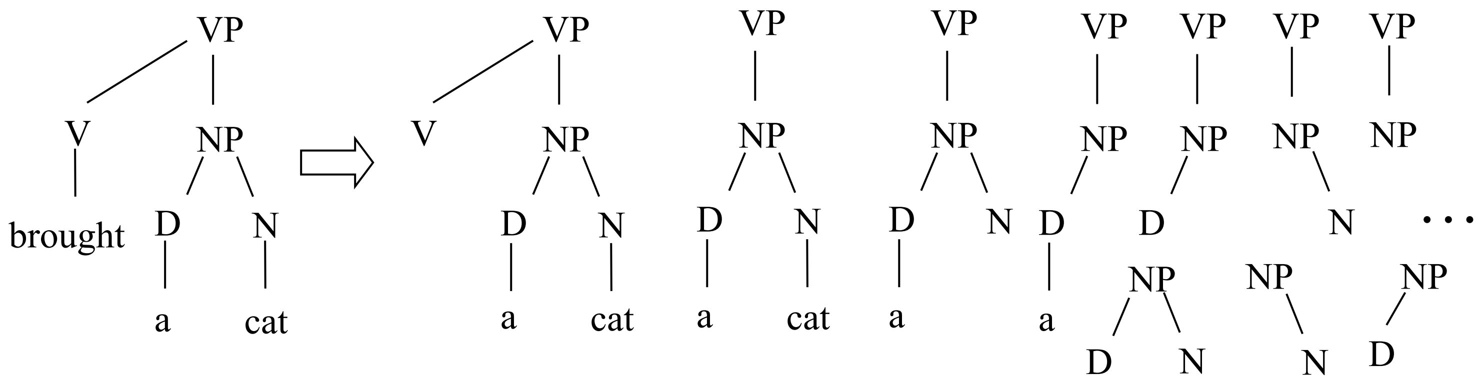
Weighting Problems



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

Partial Trees, [ECML 2006]

- SST + 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

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

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), $\Delta_p(s_1 a, s_2 b) =$

$\mathbf{D_p}$

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

Efficient Evaluation (2)

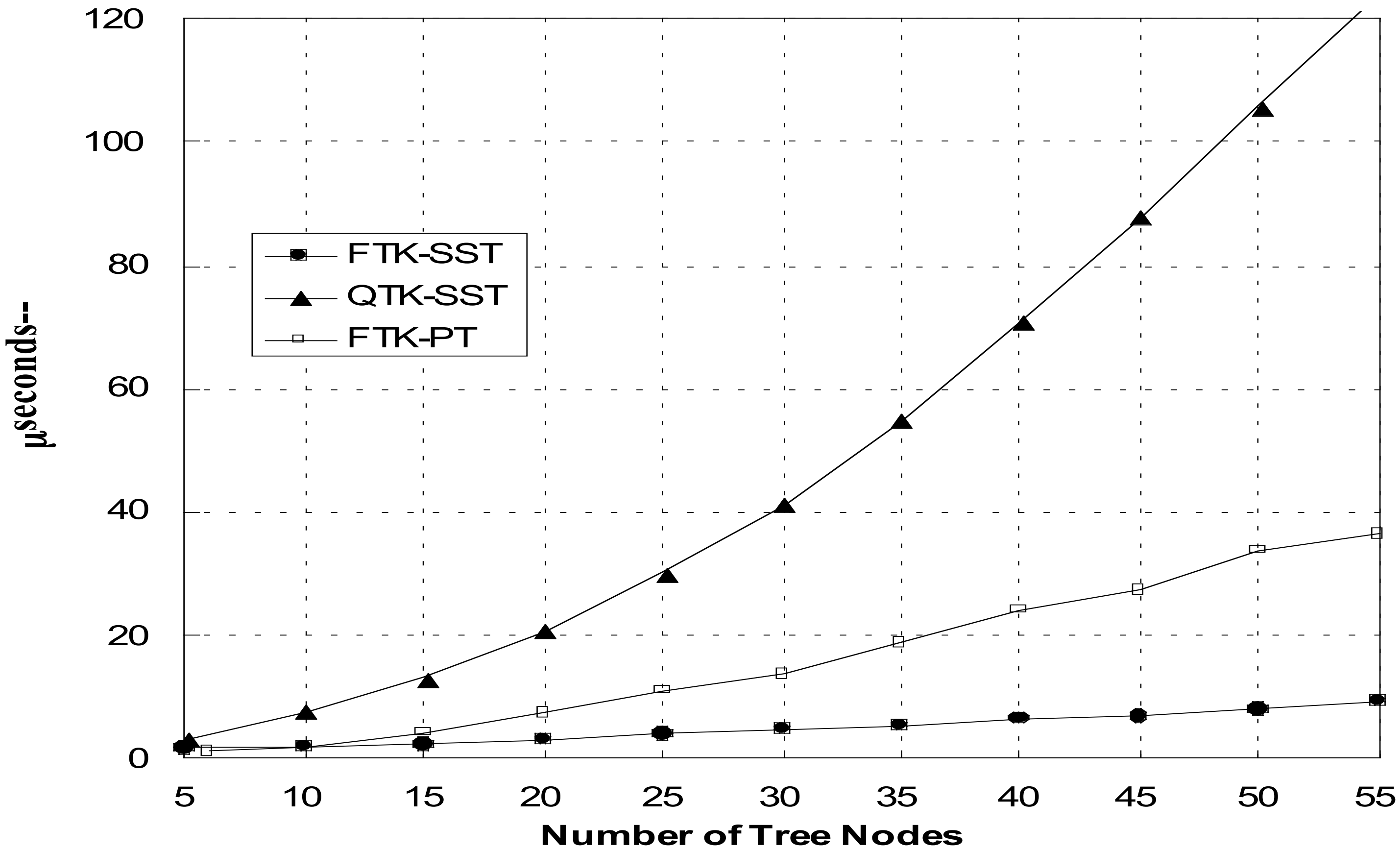
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) & \text{if } a = b; \\ 0 & otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

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

- 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 ($p = \rho$)

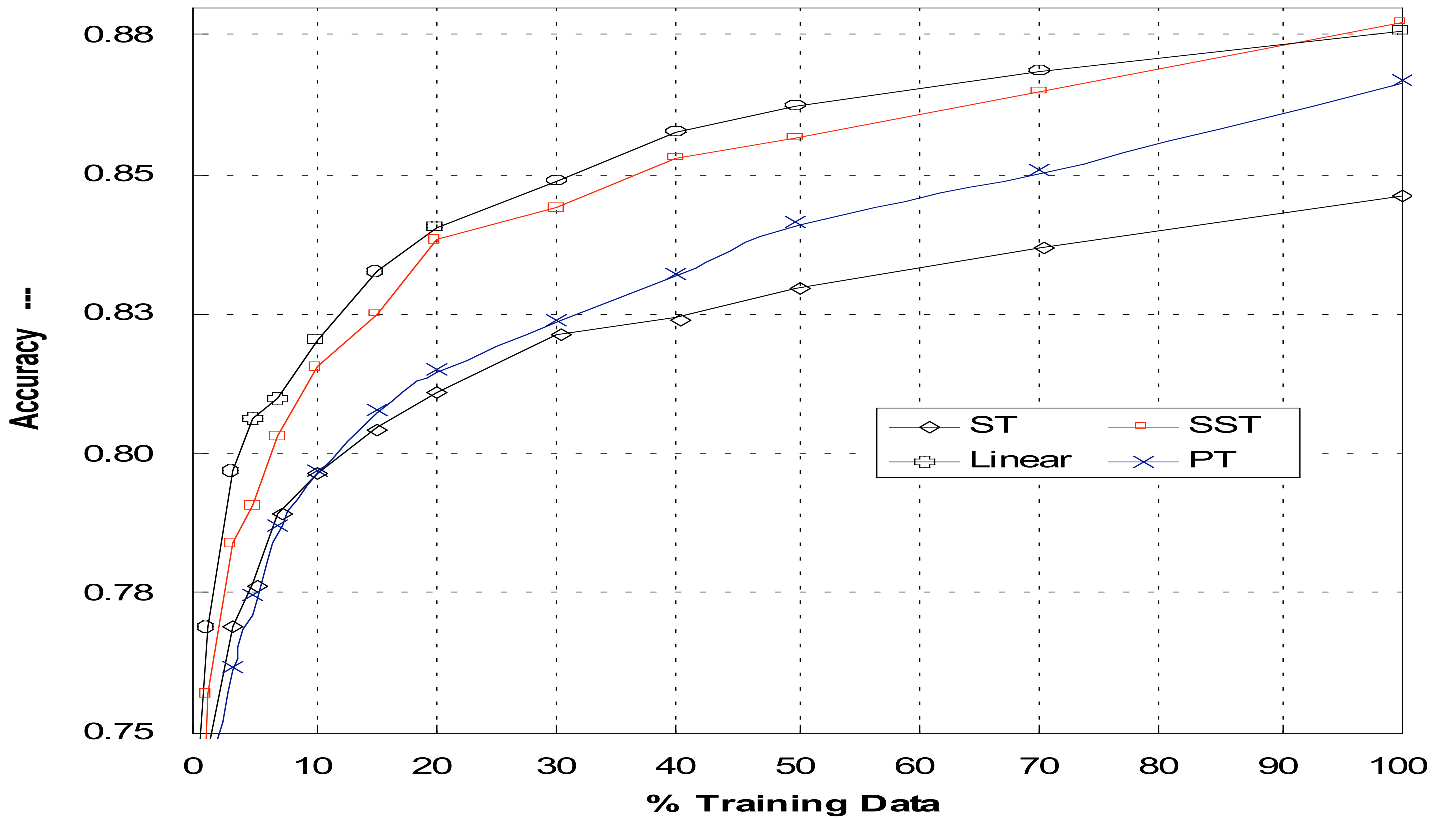
Running Time of Tree Kernel Functions



Gold Standard Tree Experiments

- PropBank and PennTree bank
  - about 53,700 sentences
  - Sections from 2 to 21 train., 23 test., 1 and 22 dev.
  - Arguments from Arg0 to Arg5, ArgA and ArgM for a total of 122,774 and 7,359

Argument Classification Accuracy



PART II: Kernel Engineering for Language Applications

- Basic Combinations
- *Canonical Mappings*, e.g. object transformations
- *Merging of Kernels*

Kernel Combinations an example

K_p^3 polynomial kernel of flat features

K_{Tree} Tree kernel

- Kernel Combinations:

$$K_{Tree+P} = \gamma \times K_{Tree} + K_p^3, \qquad K_{Tree \times P} = K_{Tree} \times K_p^3$$

$$K_{Tree+P} = \gamma \times \frac{K_{Tree}}{|K_{Tree}|} + \frac{K_p^3}{|K_p^3|}, \qquad K_{Tree \times P} = \frac{K_{Tree} \times K_p^3}{|K_{Tree}| \times |K_p^3|}$$

Object Transformation [CLJ 2008]

- $$K(O_1,O_2) = \phi(O_1)\cdot\phi(O_2) = \phi_E(\phi_M(O_1))\cdot\phi_E(\phi_M(O_2))$$
  $$= \phi_E(S_1)\cdot\phi_E(S_2) = K_E(S_1,S_2)$$
- ***Canonical Mapping***, $\phi_M()$
  - object transformation,
  - e. g. a syntactic parse tree, into a verb subcategorization frame tree.
- ***Feature Extraction***, $\phi_E()$
  - maps the canonical structure in all its fragments
  - different fragment spaces, e. g. ST, SST and PT.

Object Transformation and Kernel Combinations: Semantic Role Labeling

- Kernel Engineering via node marking
- Kernels for the whole predicate argument structures
- Kernels for re-ranking

Predicate Argument Classification

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

  Paul gives a lecture in Rome

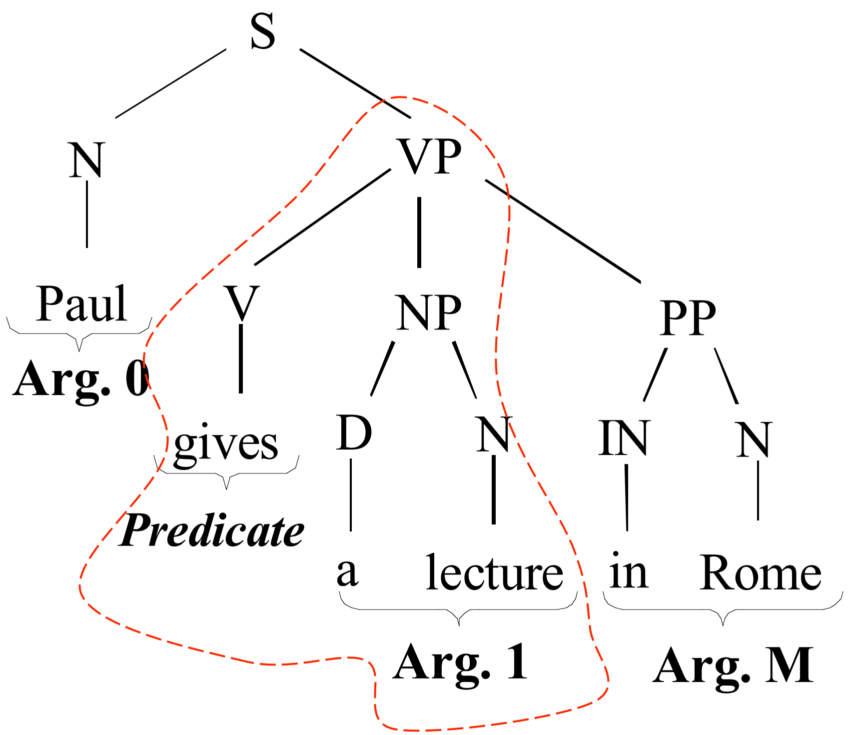
Predicate Argument Classification

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

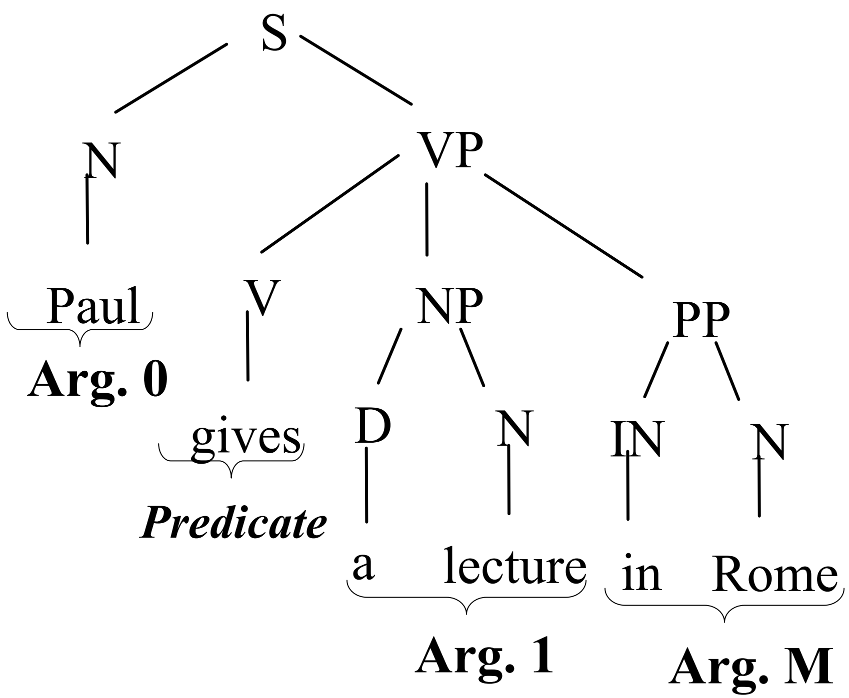
  [$_{Arg0}$ Paul] [$_{predicate}$ gives] [$_{Arg1}$ a lecture] [$_{ArgM}$ in Rome]

Predicate Argument Structures and syntax

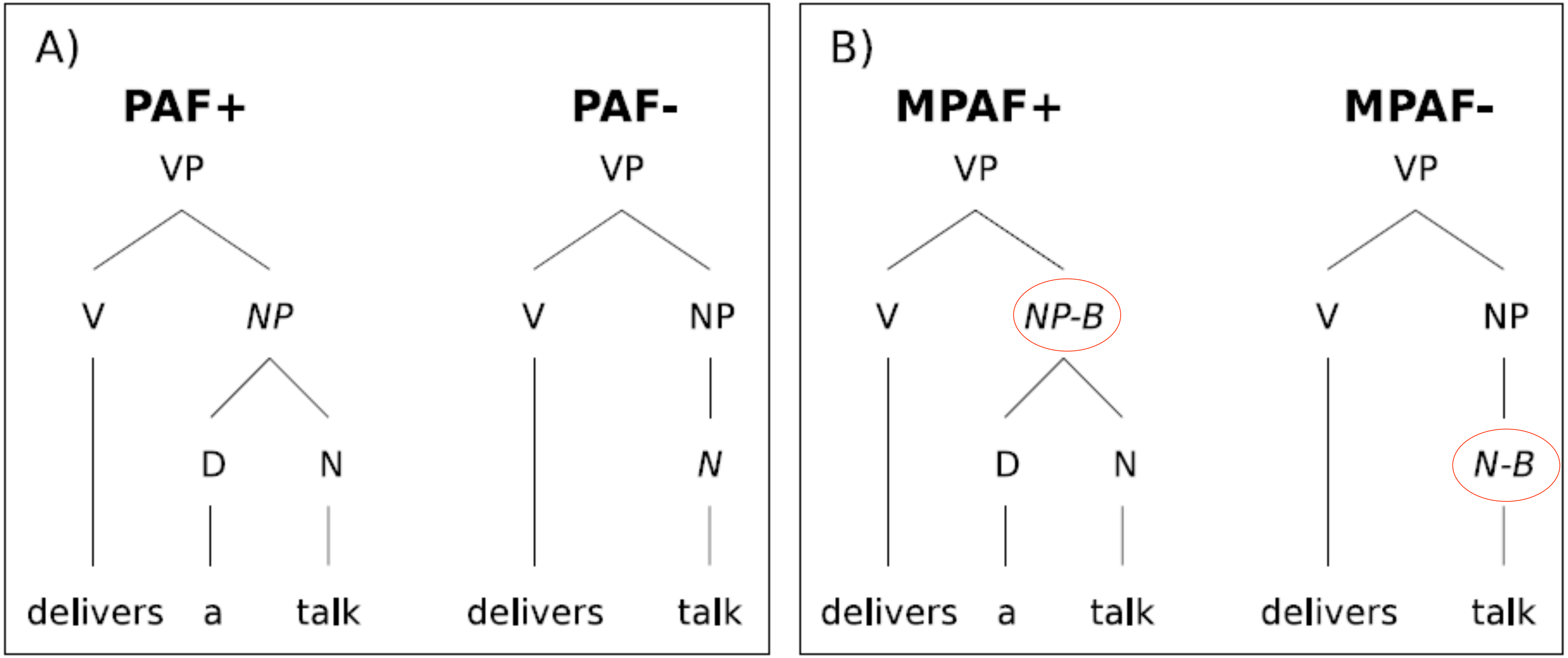
- Semantics are connected to syntax via parse trees



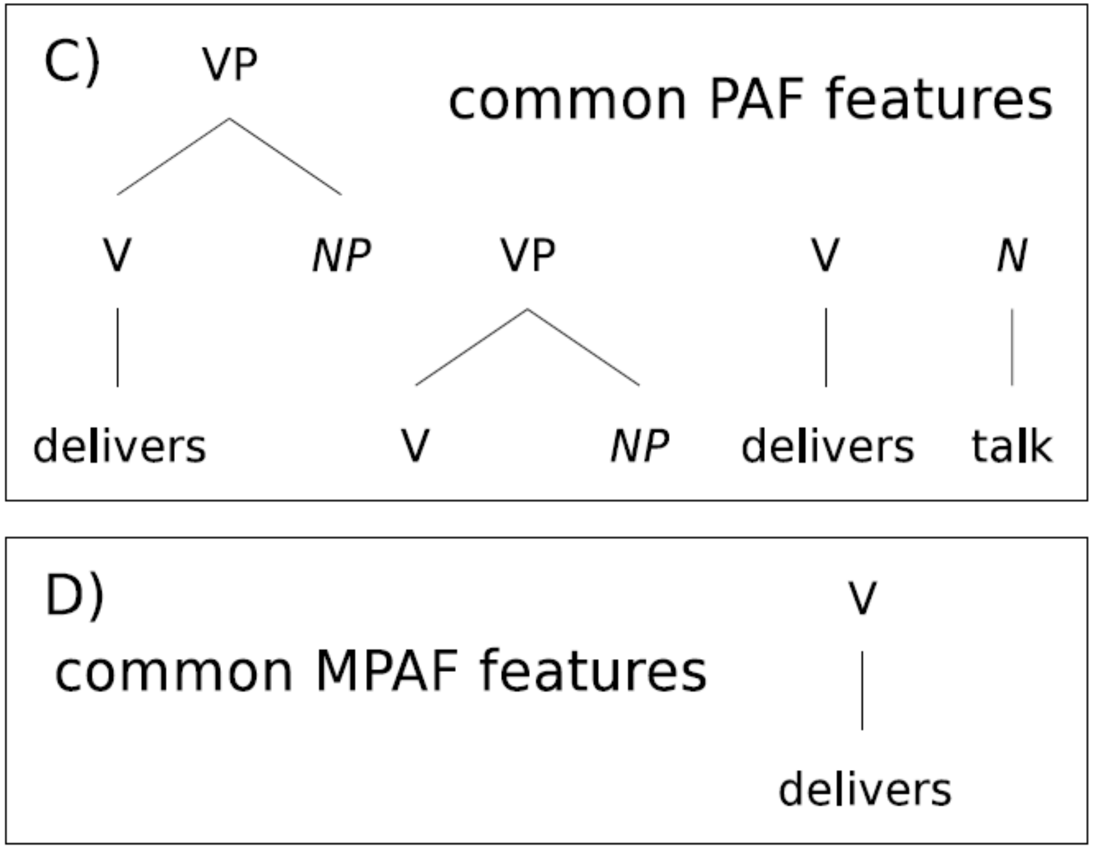
Object Transformation: tree tailoring [ACL 2004]



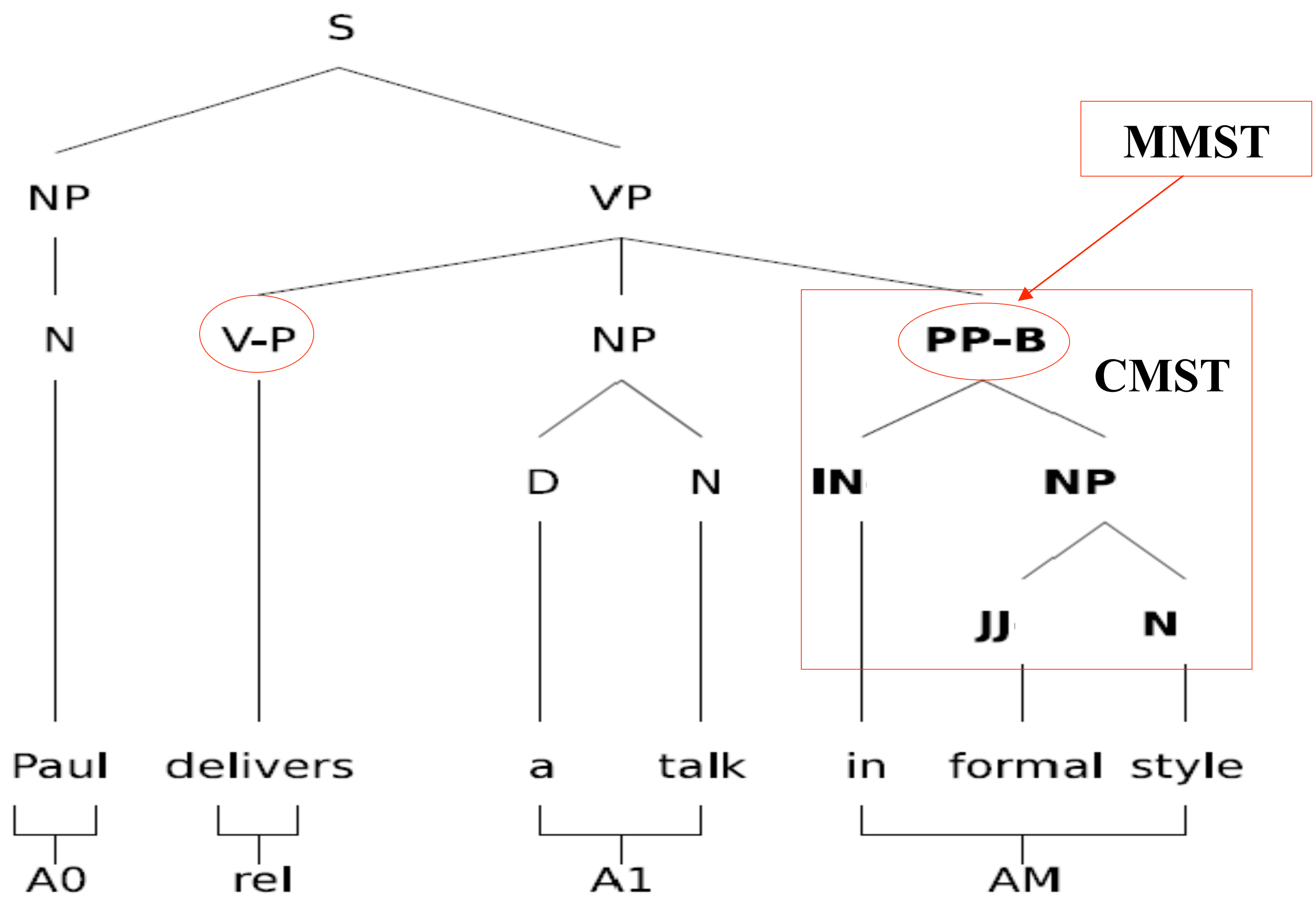
Object Transformation: Node Marking



Node Marking Effect



Different tailoring and marking



Experiments

- PropBank and PennTree bank
  - about 53,700 sentences
  - Charniak trees from CoNLL 2005
- Boundary detection:
  - Section 2 training
  - Section 24 testing
  - PAF and MPAF

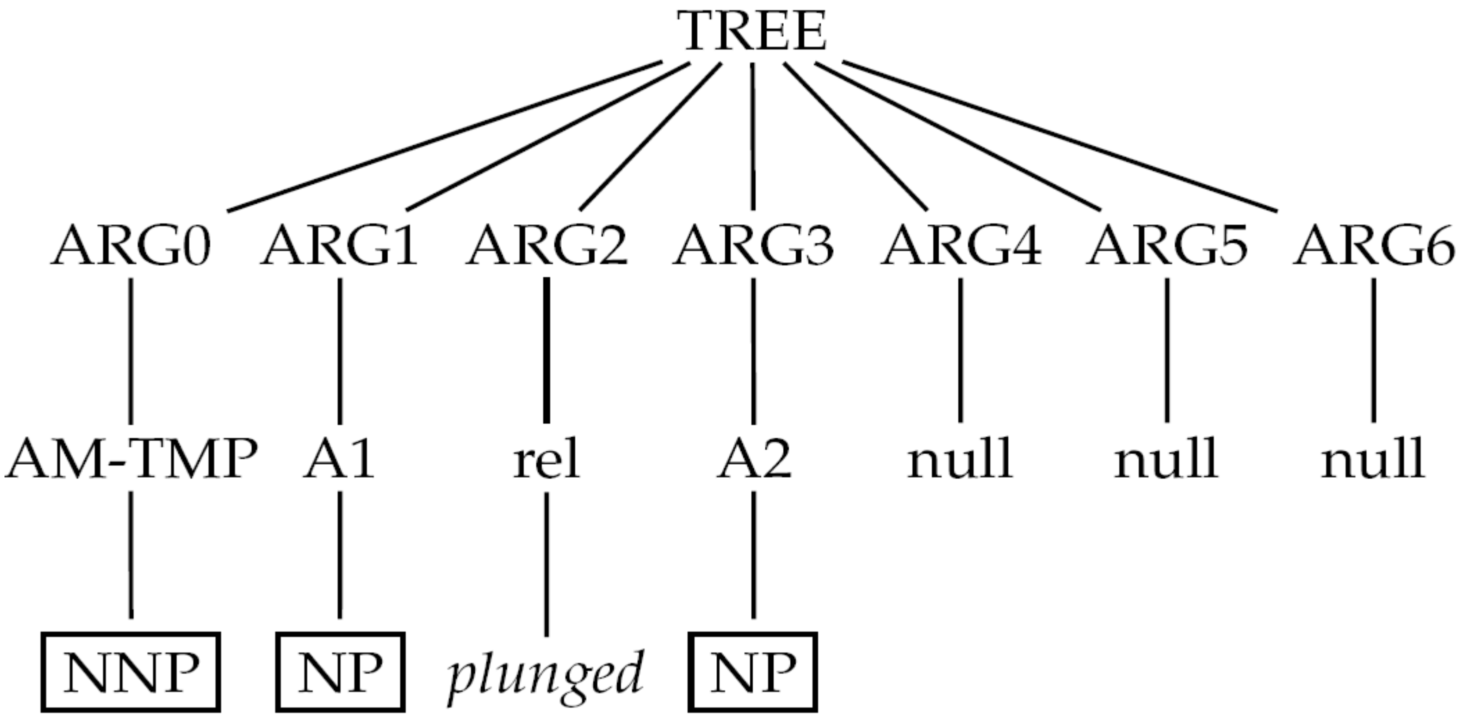
Number of examples/nodes of Section 2

	Section 2			Section 24		
Nodes	pos	neg	tot	pos	neg	tot
Internal	11,847	71,126	82,973	7,525	50,123	57,648
Pre-terminal	894	114,052	114,946	709	80,366	81,075
Both	12,741	185,178	197,919	8,234	130,489	138,723

Predicate Argument Feature (PAF) vs. Marked PAF (MPAF) [ACL-ws-2005]

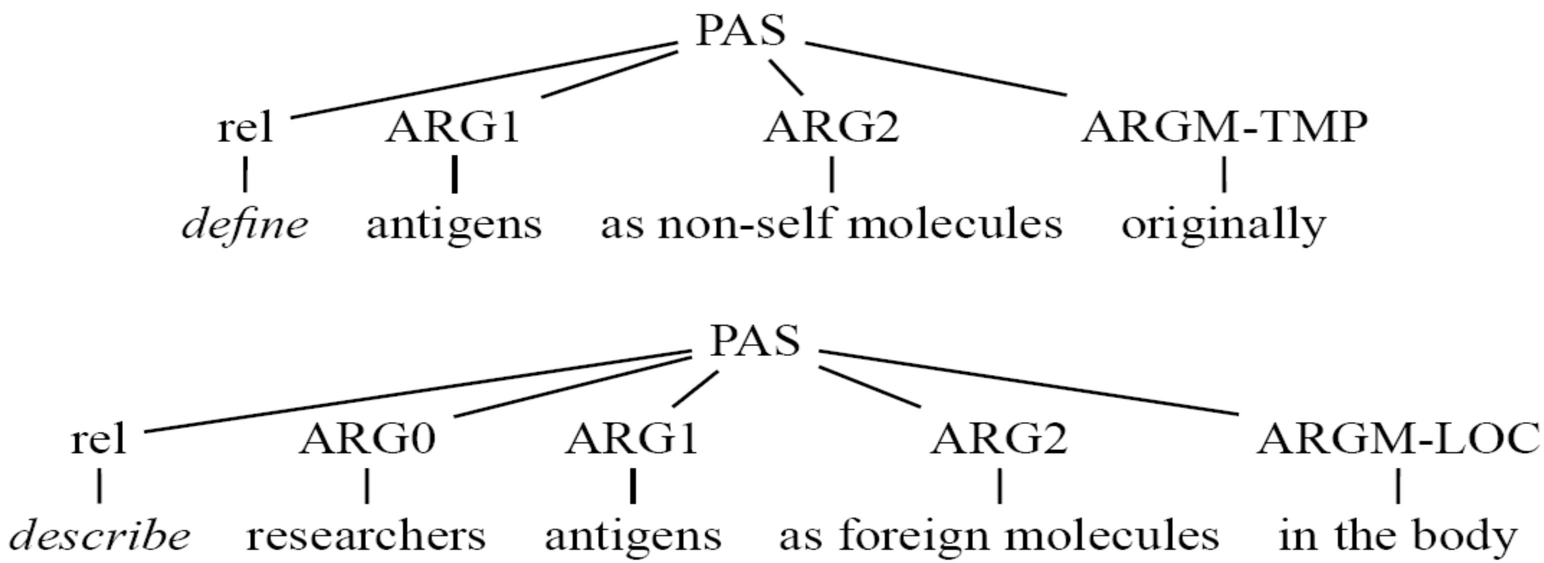
Tagging strategy	CPU_{time}	F1
PAF	5,179.18	75.24
MPAF	3,131.56	82.07

More general mappings: Semantic structures for re-ranking [CoNLL 2006]

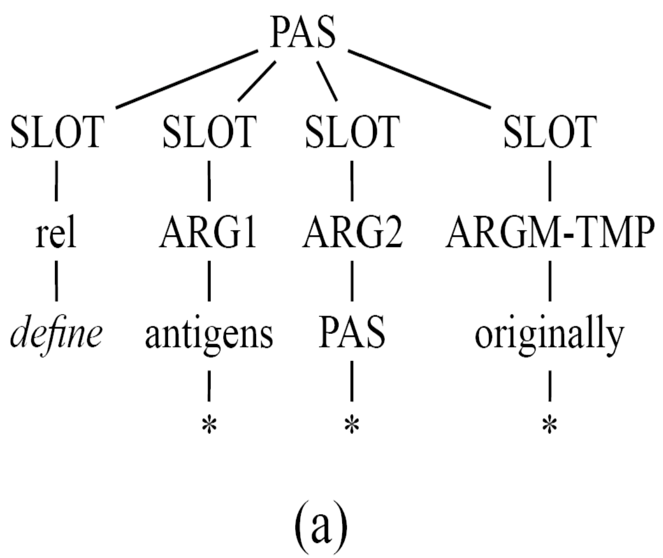


Other Shallow Semantic structures

- [**ARG1** Antigens] were [**AM−TMP** originally] [**rel** defined] [**ARG2** as non-self molecules].
- [**ARG0** Researchers] [**rel** describe] [**ARG1** antigens][**ARG2** as foreign molecules] [**ARGM−LOC** in the body]



Shallow Semantic Trees for SST kernel
[ACL 2007]



Merging of Kernels [ECIR 2007]: Question/Answer Classification

- Syntactic/Semantic Tree Kernel
- Kernel Combinations
- Experiments

Merging of Kernels

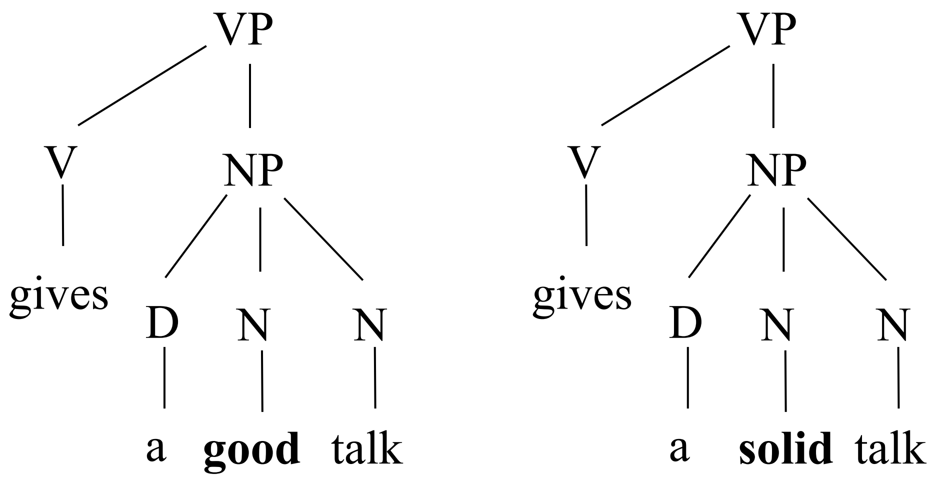
Definition 4 (Tree Fragment Similarity Kernel). *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[4]*:*

$$\kappa_{\mathcal{F}}(f_1, f_2) = comp(f_1, f_2) \prod_{t=1}^{nt(f_1)} \kappa_S(f_1(t), f_2(t))$$

$$\kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$$

where $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j)$.

Merging of Kernels



$$\kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$$

$$where\ \Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j).$$

Delta Evaluation is very simple

0. if n_1 and n_2 are pre-terminals and $label(n_1) = label(n_2)$ then $\Delta(n_1, n_2) = \lambda\kappa_S(ch^1_{n_1}, ch^1_{n_2})$,
1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)}(1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

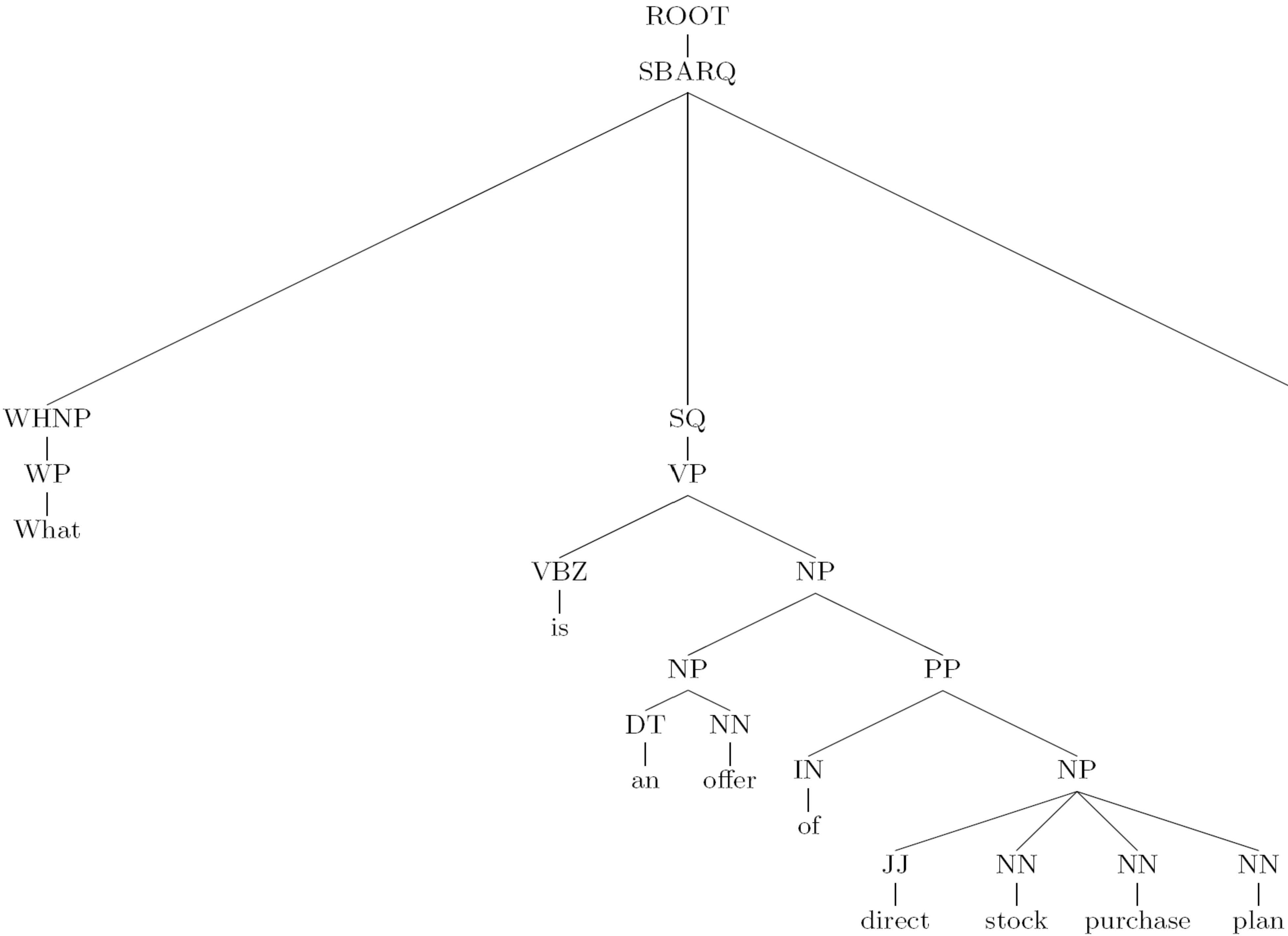
Question Classification

- **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

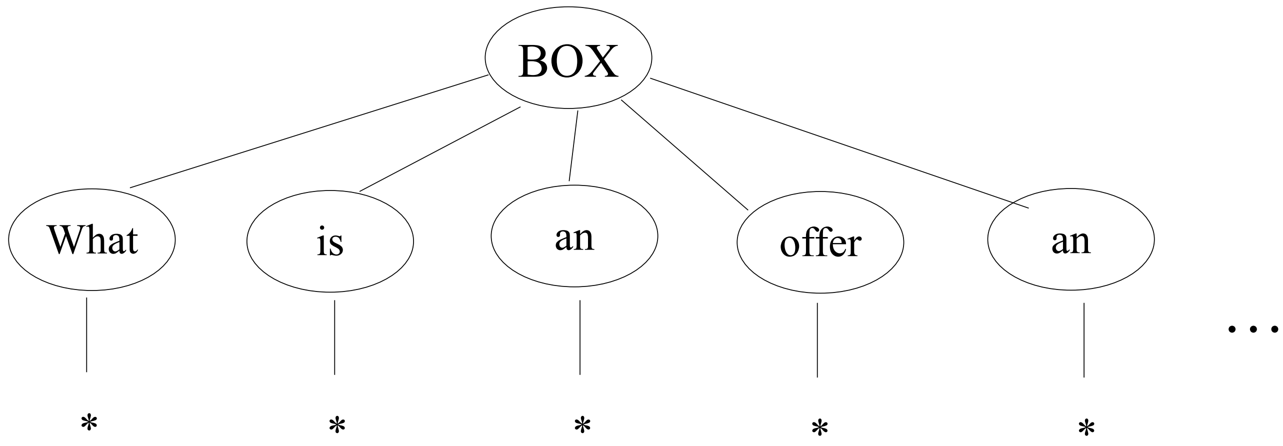
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Cross-validation (10-folds)
- Using the whole question parse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

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


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

Kernels

- BOW, POS are obtained with a simple tree, e.g.



- PT (parse tree)
- PAS (predicate argument structure)

Question classification

Features	Accuracy (UIUC)	Accuracy (c.v.)
PT	90.4	84.8±1.4
BOW	90.6	84.7±1.4
PAS	34.2	43.0±2.2
POS	26.4	32.4±2.5
PT+BOW	**91.8**	**86.1±1.3**
PT+BOW+POS	91.8	84.7±1.7
PAS+BOW	90.0	82.1±1.5
PAS+BOW+POS	88.8	81.0±1.7

Similarity based on WordNet

Inverted Path Length:

$$sim_{IPL}(c_1, c_2) = \frac{1}{(1 + d(c_1, c_2))^{\alpha}}$$

Wu & Palmer:

$$sim_{WUP}(c_1, c_2) =$$
$$\frac{2\, dep(lso(c_1, c_2))}{d(c_1, lso(c_1, c_2)) + d(c_2, lso(c_1, c_2)) + 2\, dep(lso(c_1, c_2))}$$

Resnik:

$$sim_{RES}(c_1, c_2) = -\log\ P(lso(c_1, c_2))$$

Lin:

$$sim_{LIN}(c_1, c_2) = \frac{2\ \log\ P(lso(c_1, c_2))}{\log\ P(c_1) + \log P\,(c_2)}$$

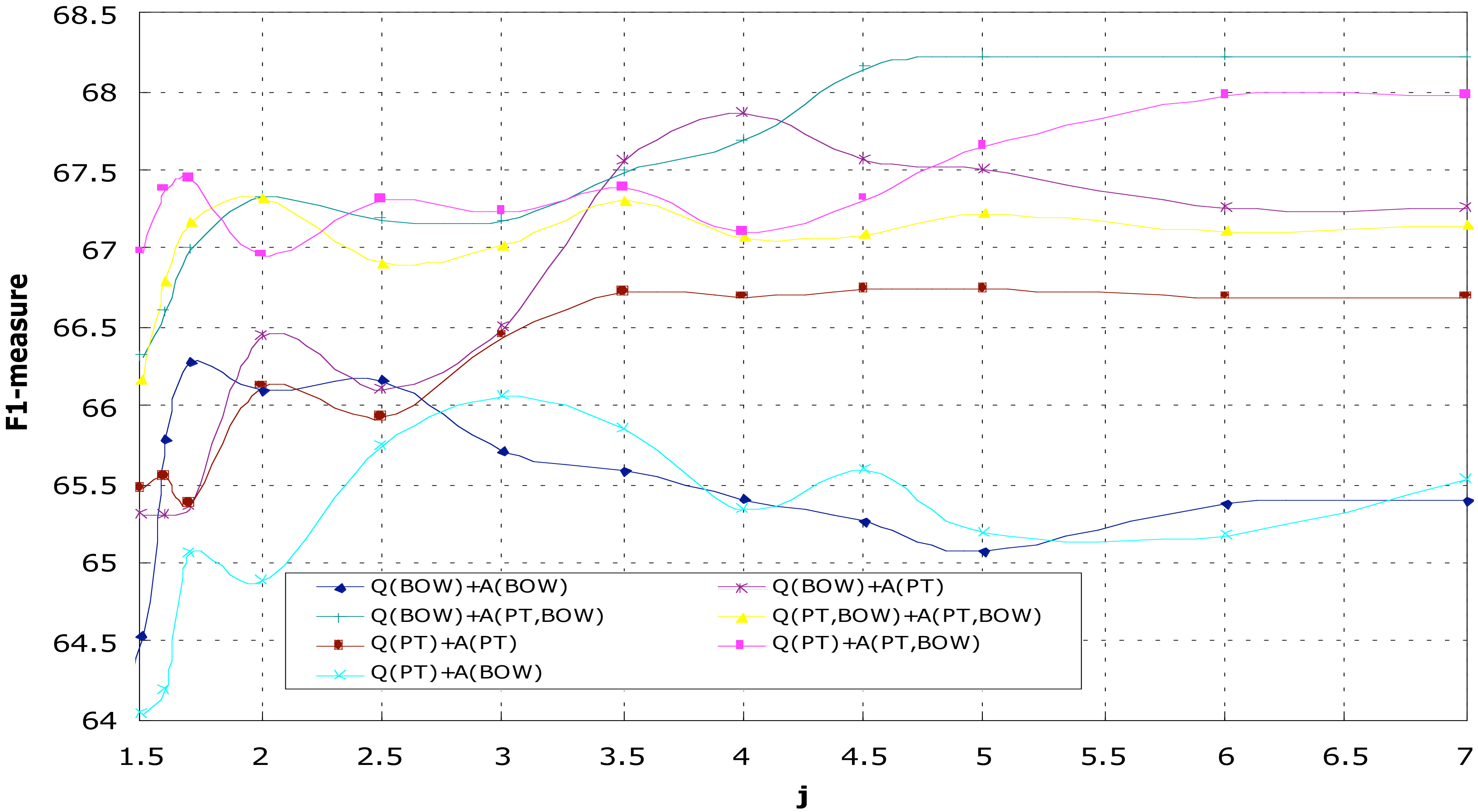
Question Classification with S/STK

	Accuracy				
λ parameter	**0.4**	**0.05**	**0.01**	**0.005**	**0.001**
linear (bow)	0.905				
string matching	0.890	0.910	**0.914**	**0.914**	0.912
full	0.904	**0.924**	0.918	0.922	0.920
full-ic	0.908	**0.922**	0.916	0.918	0.918
path-1	0.906	**0.918**	0.912	**0.918**	0.916
path-2	0.896	0.914	0.914	**0.916**	**0.916**
lin	0.908	**0.924**	0.918	0.922	0.922
wup	0.908	**0.926**	0.918	0.922	0.922

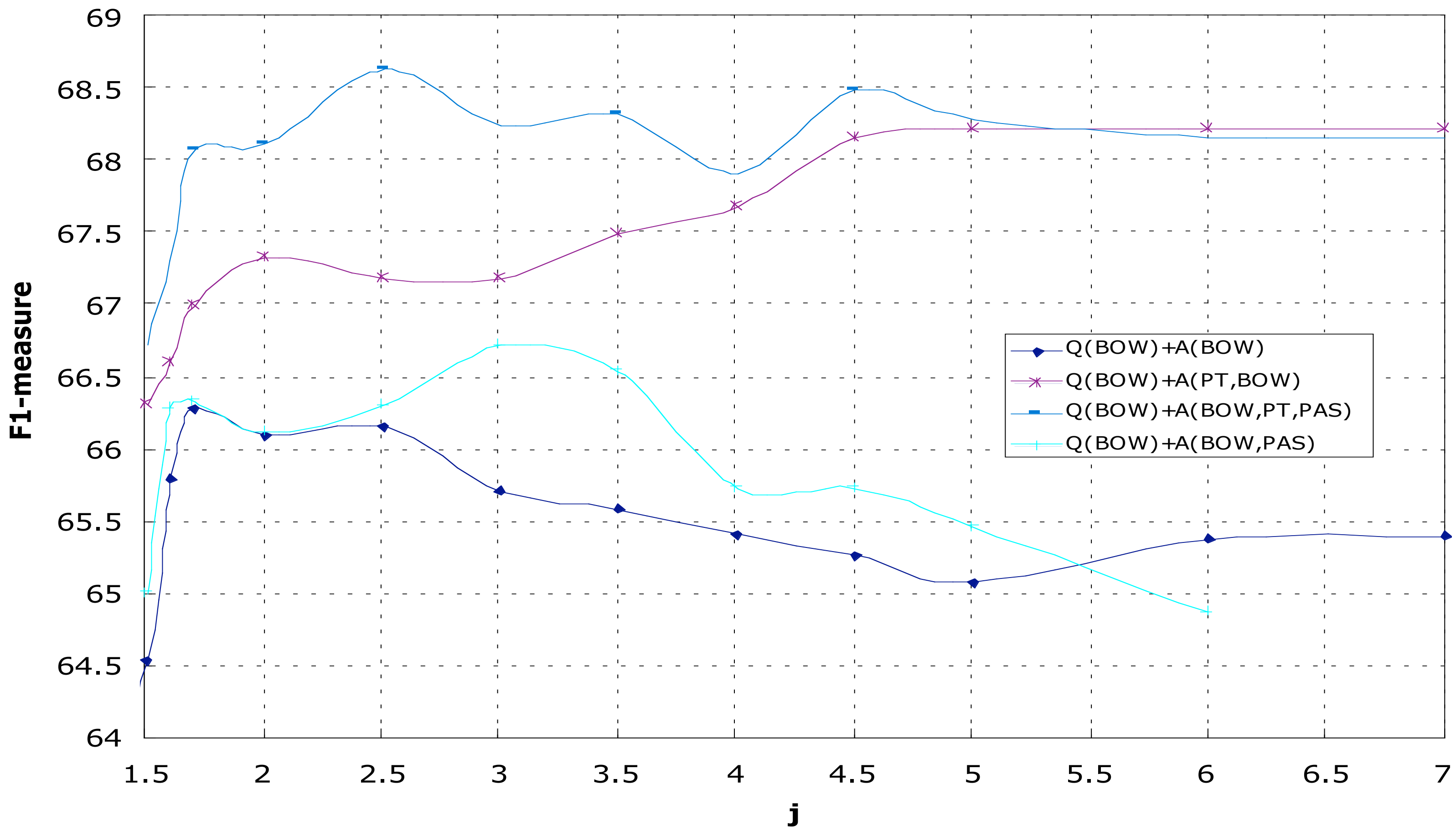
Answer Classification data-set

- 138 TREC 2001 test questions labeled as “description”
- 1309 sentences, extracted from the best ranked paragraphs (using a basic QA system based on Google search engine)
- 416 of which labeled as correct by two annotators

Impact of Bow and PT in Answer classification



The Impact of SSTK in Answer Classification



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

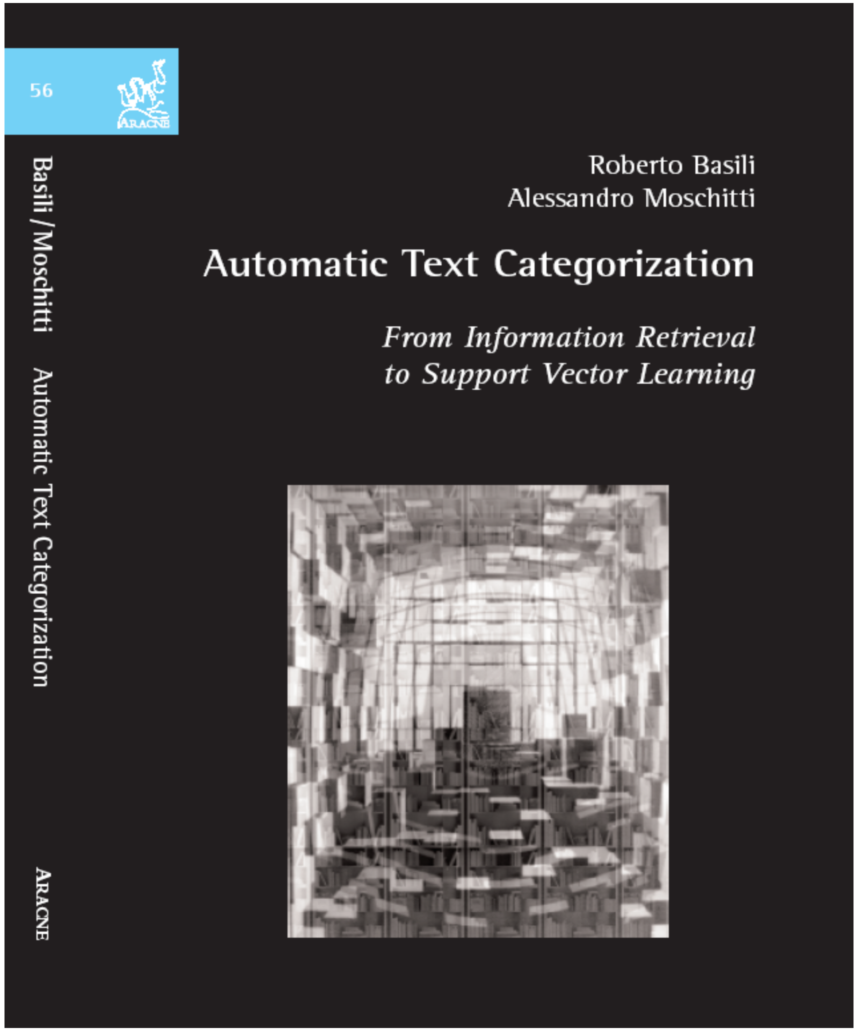
- Kernel methods and SVMs are useful tools to design language applications
- Kernel design still require some level of expertise
- Engineering approaches to tree kernels
  - Basic Combinations
  - Canonical Mappings, e.g.
    - Node Marking
  - Merging of kernels in more complex kernels
- State-of-the-art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

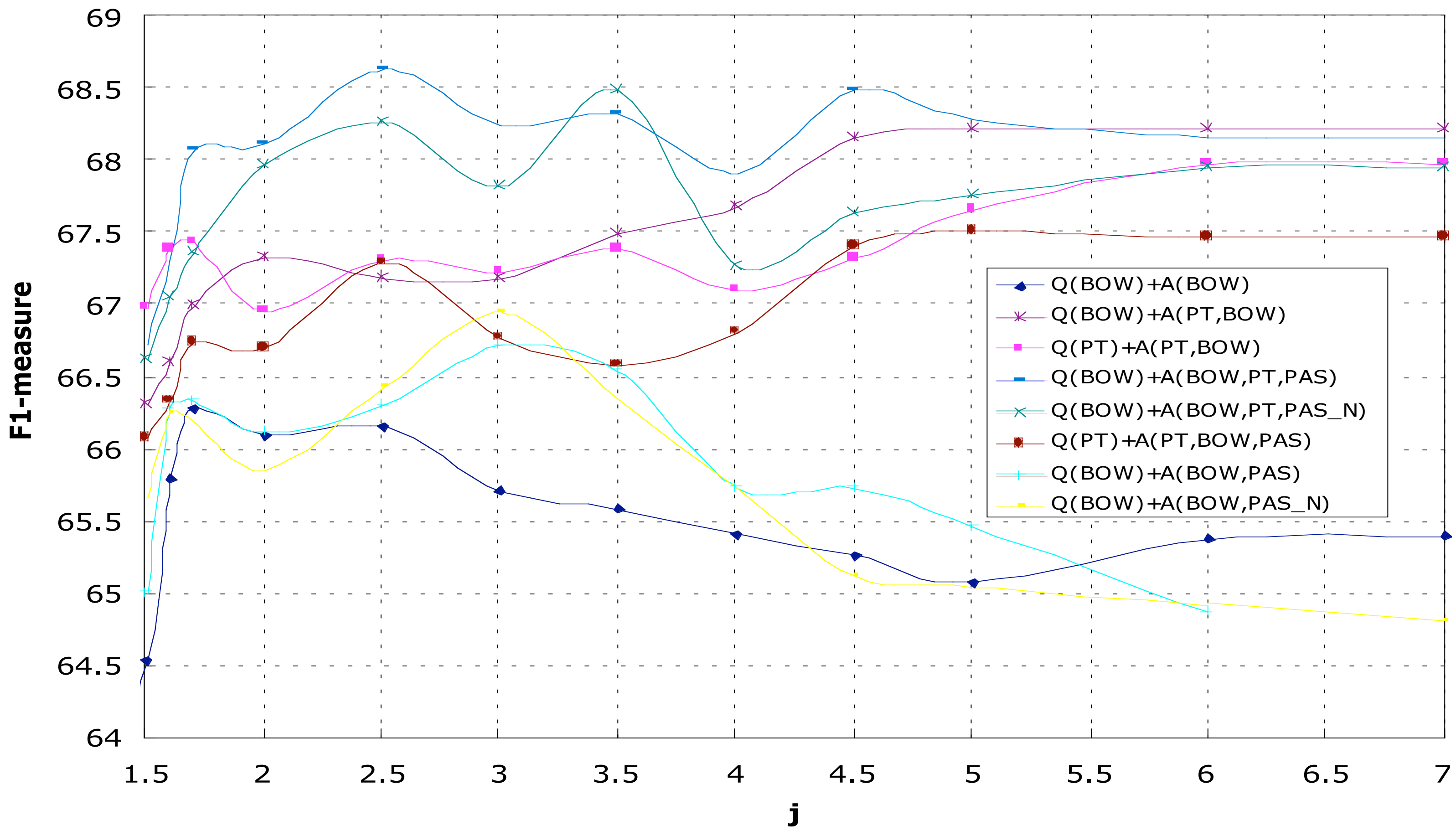
- Alessandro Moschitti, *A study on Convolution Kernel for Shallow Semantic Parsing*. In proceedings of ACL-2004, Spain, 2004.
- Alessandro Moschitti and Cosmin Adrian Bejan, *A Semantic Kernel for Predicate Argument Classification.* In proceedings of the CoNLL-2004, Boston, MA, USA, 2004.
- M. Collins 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

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