Coling 2010

23rd International Conference on Computational Linguistics

Kernel Engineering for Fast and Easy Design of Natural Language Applications

Alessandro Moschitti

Department of Information Engineering and Computer Science
University of Trento

August 22, 2010

To order the CD of Coling 2010 and its Workshop Proceedings, please contact:

Chinese Information Processing Society of China
No.4, Southern Fourth Street
Haidian District, Beijing, 100190
China
Tel: +86-010-62562916
Fax: +86-010-62562916
cips@iscas.ac.cn

Tutorial Instuctor

Alessandro Moschitti: *DISI, University of Trento, Italy. E-mail: moschitti@disi.unitn.it*

Alessandro is a professor at the Information Engineering and Computer Science Department of the University of Trento. In 2003, he obtained his PhD in Computer Science at the University of Rome ”Tor Vergata” and between 2002 and 2004, he worked as an associate researcher in the University of Texas at Dallas for two years. His expertise concerns machine learning approaches to Natural Language Processing, Information Retrieval and Data Mining. In particular, he has recently devised innovative kernels within Support Vector and other kernel-based machines for advanced syntactic/semantic processing. He is author of more than 100 scientific articles published in the major conferences of different research communities, e.g. ACL, ICML, CIKM and ICDM. He has participated in several projects of the European Community (EC), e.g. LIVING-KNOWLEDGE 2008, PRESTOSPACE 2004, NAMIC 2000 and TREVI 1998 and in two US projects: MTBF 2008 (Con-Edison) and ARDA AQUAINT PROGRAM (IQAS 2002). He is currently project coordinator of the EC project, EternalS.

Outline

Previous work on the use of Machine Learning for Computational Linguistics has shown that most of the design effort is devoted to feature engineering. Indeed, the latter requires expertise, intuition and deep knowledge about the target problem to convert linguistic objects into attribute-value representations. Kernel Methods (KM) are powerful techniques, which can simplify data modeling by defining abstract representations and implicit feature spaces. More in particular, KM allow for: (a) directly using a similarity function between instances in learning algorithms, thus avoiding explicit feature design; and (b) implicitly defining huge feature spaces, e.g. structures can be represented in the substructure space.

In this tutorial, practical recipes to successfully use KM for target language applications will be presented: first, after an introduction to Support Vector Machines (explained from an application viewpoint), KM theory will be explained in a way that useful practical methods can be derived. Second, basic kernels, such as linear, polynomial, sequence and tree kernels will be presented, by focusing on the implementation, accuracy and efficiency perspectives. KM application to typical natural language tasks, e.g. text categorization, question and answer classification, semantic role labeling, textual entailment and so on, will be shown. The aim is to provide practical procedures for the selection and exploitation of the right kernel for the target task. Third, the SVM-Light-TK toolkit, which encodes several kernels in SVMs, will be illustrated along with the associated data structures and its practical use in NL tasks. Finally, the tutorial will illustrate how innovative and effective kernels can be engineered starting from basic kernels and using systematic data transformation. Such know-how allows for a very fast and accurate design of applications even if the underlying language phenomena and properties are still not very well understood, e.g. Arabic SRL or relation extraction between pairs of text fragments.

Kernel Engineering for Fast and Easy Design of Natural Language Applications

Alessandro Moschitti
Department of Information Engineering and Computer Science
University of Trento
Email: moschitti@disi.unitn.it

The 23rd International Conference on Computational Linguistics
August 22, 2010
Beijing, China

Schedule

- 14:00 - 15:30 First part
- 15:30 - 16:00 Coffee break
- 16:00 - 17:30 Second part

Coling 2008: Kernel Engineering for Fast and Easy Design of Natural Language Applications–Tutorial notes, pages 1–91,
Beijing, August 2010

Outline (1)

- Motivation
- Kernel-Based Machines
  - Perceptron
  - Support Vector Machines
- Kernel Definition
  - Kernel Trick
  - Mercer's conditions
  - Kernel operators
- Basic Kernels
  - Linear Kernel
  - Polynomial Kernel
  - Lexical Kernel

Outline (2)

- Structural Kernels
  - String and Word Sequence Kernels
  - Tree Kernels
    - Subtree, Syntactic, Partial Tree Kernels
- Applied Examples of Structural Kernels
  - Semantic Role Labeling (SRL)
  - Question Classification (QC)
  - SVM-Light-TK
  - Experiments in classroom with SRL and QC
  - Inspection of the input, output, and model files

Outline (3)

- Kernel Engineering
  - Structure Transformation
  - Syntactic Semantic Tree kernels
  - Kernel Combinations
  - Kernels on Object Pairs
  - Kernels for re-ranking
- Practical Question and Answer Classifier based on SVM-Light-TK
  - Combining Kernels
- Conclusion and Future Work

Motivation (1)

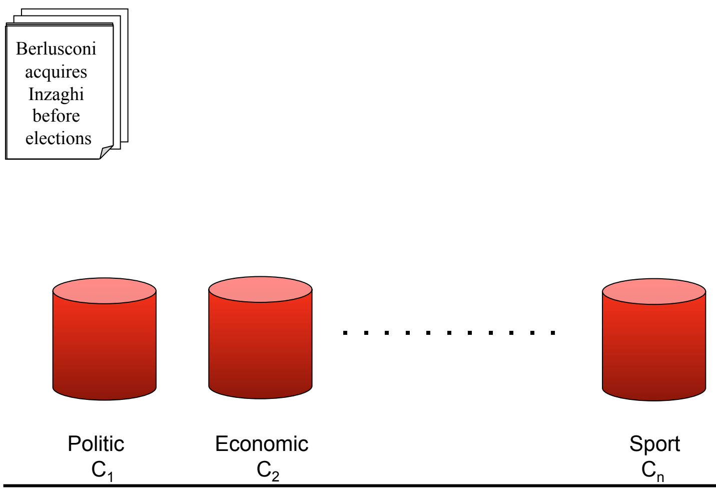
- Feature design most difficult aspect in designing a learning system
  - complex and difficult phase, e.g., structural feature representation:
  - deep knowledge and intuitions are required
  - design problems when the phenomenon is described by many features

Motivation (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

Part I: Kernel Methods Theory

A simple classification problem: Text Categorization



Text Classification Problem

- Given:
  - a set of target categories: $C = \{C^1,..,C^n\}$
  - the set *T* of documents,

  define

  $$f : T \rightarrow 2^C$$

- VSM (Salton89')
  - Features are dimensions of a Vector Space.
  - Documents and Categories are vectors of feature weights.
  - *d* is assigned to C^i if $\vec{d} \cdot \vec{C}^i > th$

More in detail

- 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

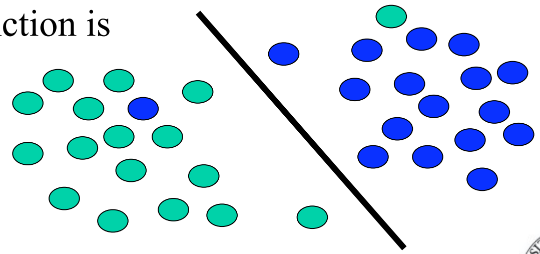
- The dot product $\vec{x} \cdot \vec{z}$ counts the number of features in common
- This provides a sort of *similarity*

Linear Classifier

- The equation of a hyperplane is

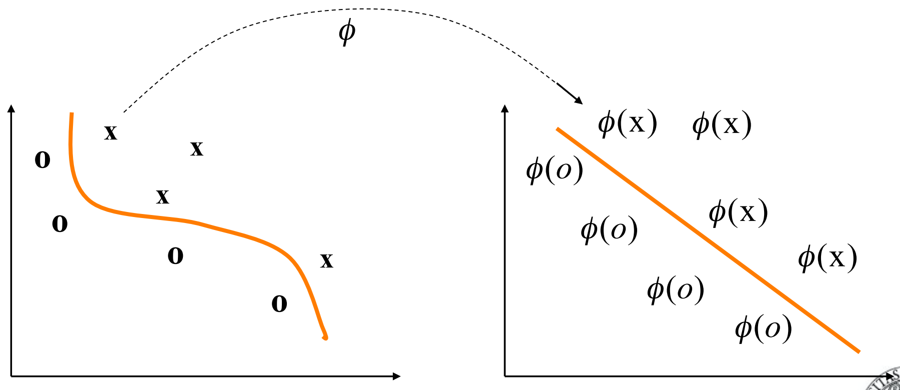
$$f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x}, \vec{w} \in \Re^n, b \in \Re$$

- $\vec{x}$ 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 $\vec{x} \rightarrow \phi(\vec{x})$



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)

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

- 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

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

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

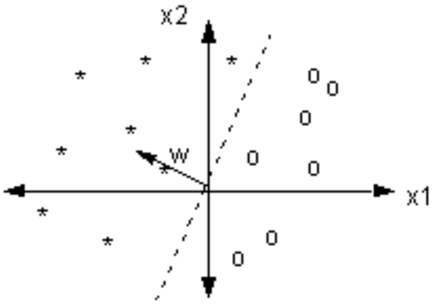
    k = k + 1

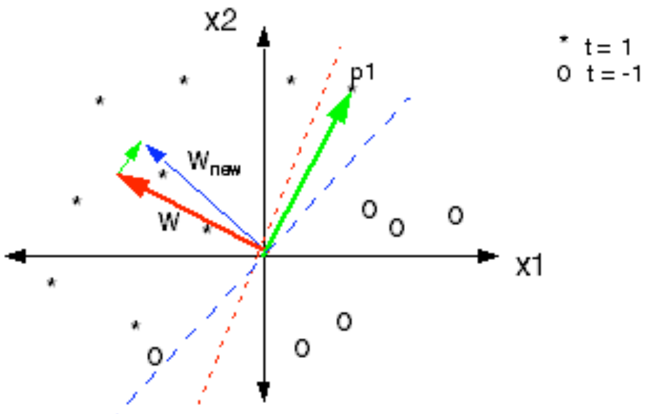
   endif

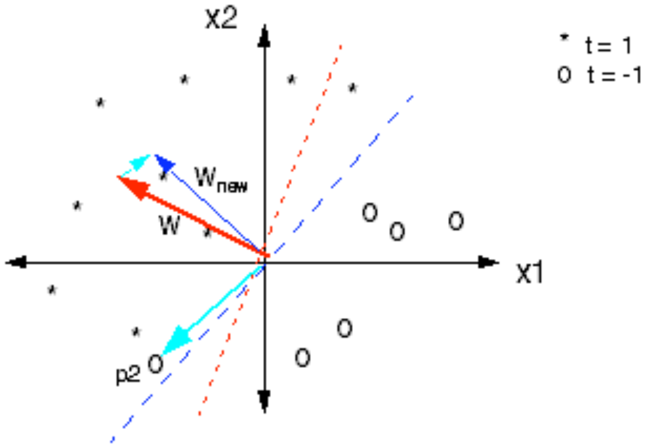
  endfor

while an error is found

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


x2
w
x1


x2
p1
* t = 1
o t = -1
W_new
W
x1



Novikoff's Theorem

Let S be a non-trivial training-set and let

$$R = \max_{1 \le i \le l} || x_i || .$$

Let us suppose there is a vector $\mathbf{w}^*, || \mathbf{w}^* || = 1$ and

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \ge \gamma, \quad i = 1, ..., l,$$

with $\gamma > 0$. Then the maximum number of errors of the perceptron is:

$$t^* = \left(\frac{2R}{\gamma} \right)^2 ,$$

Dual Representation for Classification

- In 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

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

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

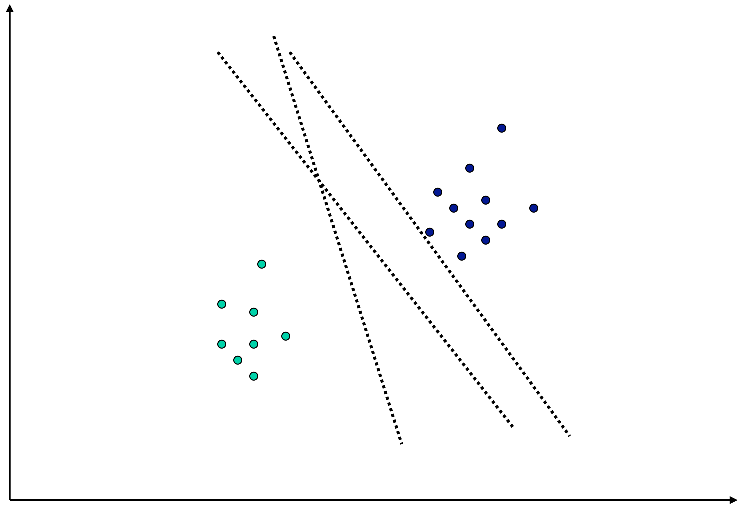
- As well as the updating function

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

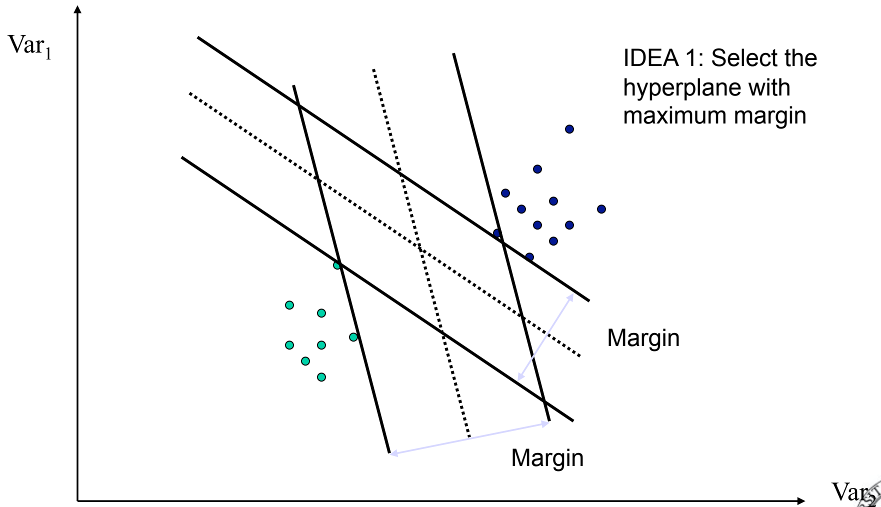
Support Vector Machines

- *Hard-margin SVMs*
- *Soft-margin SVMs*

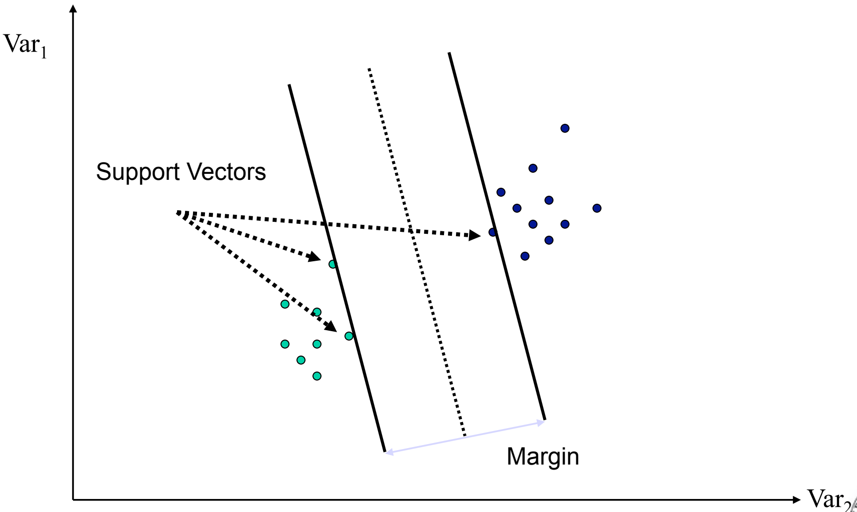
Which hyperplane do we choose?



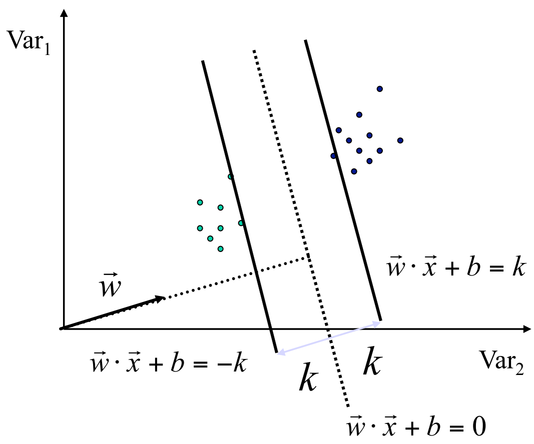
Classifier with a Maximum Margin



Support Vectors

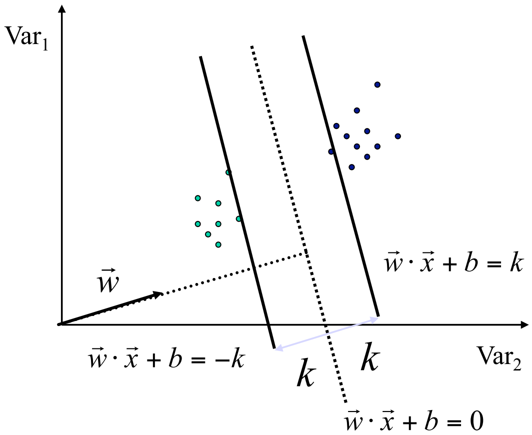


Support Vector Machines



The margin is equal to $\frac{2|k|}{\|w\|}$

Support Vector Machines



The margin is equal to $\frac{2|k|}{\|w\|}$

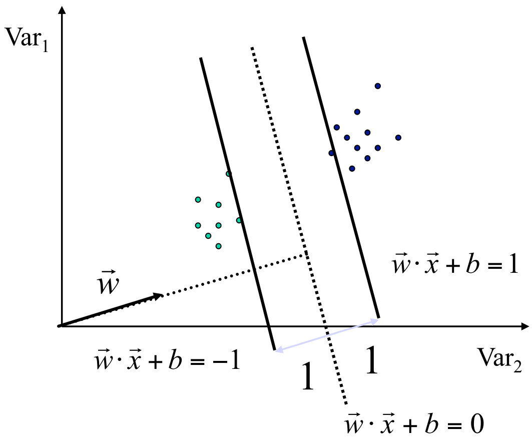
We need to solve

$$\max \frac{2|k|}{\|\vec{w}\|}$$

$$\vec{w} \cdot \vec{x} + b \geq +k, \text{ if } \vec{x} \text{ is positive}$$

$$\vec{w} \cdot \vec{x} + b \leq -k, \text{ if } \vec{x} \text{ is negative}$$

Support Vector Machines



There is a scale for which *k=1*.

The problem transforms in:

$$\max \frac{2}{\|\vec{w}\|}$$

$$\vec{w} \cdot \vec{x} + b \geq +1, \text{ if } \vec{x} \text{ is positive}$$

$$\vec{w} \cdot \vec{x} + b \leq -1, \text{ if } \vec{x} \text{ is negative}$$

Final Formulation

$$\max \frac{2}{\|\vec{w}\|}$$
$$\vec{w}\cdot\vec{x}_i + b \geq +1,\ \ y_i = 1$$
$$\vec{w}\cdot\vec{x}_i + b \leq -1,\ \ y_i = -1$$

$$\Rightarrow \quad \max \frac{2}{\|\vec{w}\|} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 \quad \Rightarrow$$

$$\Rightarrow \quad \min \frac{\|\vec{w}\|}{2} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 \quad \Rightarrow \quad \min \frac{\|\vec{w}\|^2}{2} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

Optimization Problem

- Optimal Hyperplane:
  - Minimize $\tau(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$
  - Subject to $y_i\,((\vec{w}\cdot\vec{x}_i) + b) \geq 1, i = 1,\ldots,l$
- The dual problem is simpler

Lagrangian Definition

Def. 2.24 *Let $f(\vec{w})$, $h_i(\vec{w})$ and $g_i(\vec{w})$ be the objective function, the equality constraints and the inequality constraints (i.e. $\geq$) of an optimization problem, and let $L(\vec{w}, \vec{\alpha}, \vec{\beta})$ be its Lagrangian, defined as follows:*

$$L(\vec{w}, \vec{\alpha}, \vec{\beta}) = f(\vec{w}) + \sum_{i=1}^{m} \alpha_i g_i(\vec{w}) + \sum_{i=1}^{l} \beta_i h_i(\vec{w})$$

Dual Optimization Problem

*The **Lagrangian dual problem** of the above primal problem is*

$$maximize \quad \theta(\vec{\alpha}, \vec{\beta})$$

$$subject \;\; to \quad \vec{\alpha} \geq \vec{0}$$

where $\theta(\vec{\alpha}, \vec{\beta}) = inf_{w \in W} \;\; L(\vec{w}, \vec{\alpha}, \vec{\beta})$

Dual Transformation

- Given the Lagrangian associated with our problem

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\vec{w} \cdot \vec{w} - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$$

- To solve the dual problem we need to evaluate:

$$\theta(\vec{\alpha}, \vec{\beta}) = inf_{w \in W} \ \ L(\vec{w}, \vec{\alpha}, \vec{\beta})$$

- Let us impose the derivatives to 0, with respect to $\vec{w}$

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i$$

Dual Transformation (cont'd)

- and wrt b

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial b} = \sum_{i=1}^{m} y_i \alpha_i = 0$$

- Then we substituted them in the objective function

$$\begin{aligned} L(\vec{w}, b, \vec{\alpha}) &= \frac{1}{2}\vec{w} \cdot \vec{w} - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1] = \\ &= \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \sum_{i=1}^{m} \alpha_i \\ &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \end{aligned}$$

The Final Dual Optimization Problem

$$
\begin{aligned}
&maximize && \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \\
&subject\ \ to && \alpha_i \geq 0, \quad i = 1, .., m \\
& && \sum_{i=1}^{m} y_i \alpha_i = 0
\end{aligned}
$$

Khun-Tucker Theorem

- Necessary and sufficient conditions to optimality

$$
\begin{aligned}
\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial \vec{w}} &= \vec{0} \\
\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial b} &= \vec{0} \\
\alpha_i^* g_i(\vec{w}^*) &= 0, \quad i = 1, .., m \\
g_i(\vec{w}^*) &\leq 0, \quad i = 1, .., m \\
\alpha_i^* &\geq 0, \quad i = 1, .., m
\end{aligned}
$$

Properties coming from constraints

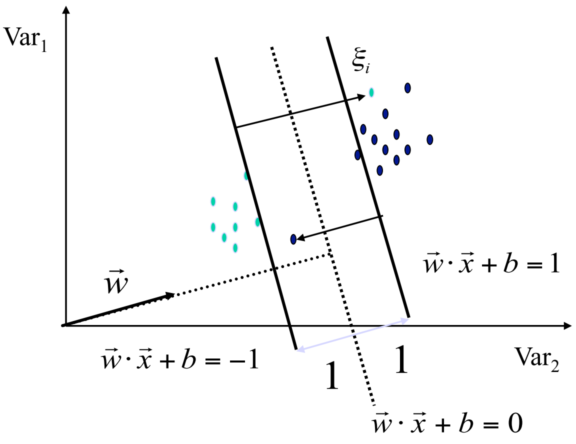
- Lagrange constraints: $\sum_{i=1}^{l} a_i y_i = 0, \quad \vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i$
- Karush-Kuhn-Tucker constraints

$$\alpha_i \cdot [y_i(\vec{x}_i \cdot \vec{w} + b) - 1] = 0, \quad i = 1,\dots,l$$

- Support Vectors have α_i not null
- To evaluate *b,* we can apply the following equation

$$b^* = -\frac{\vec{w}^* \cdot \vec{x}^+ + \vec{w}^* \cdot \vec{x}^-}{2}$$

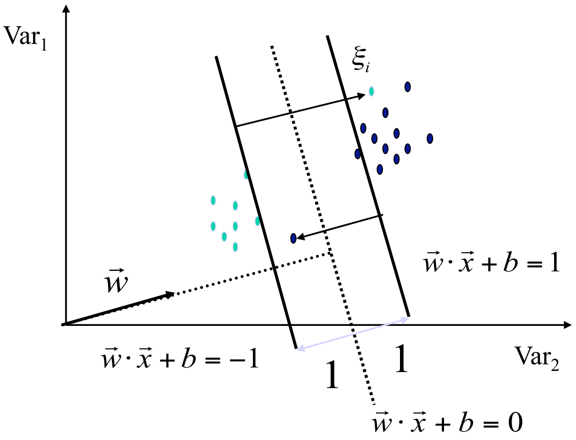
Soft Margin SVMs



ξ_i slack variables are added

Some errors are allowed but they should penalize the objective function

Soft Margin SVMs



The new constraints are

$$y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1-\xi_i$$
$$\forall\vec{x}_i \ \text{ where } \ \xi_i\geq 0$$

The objective function penalizes the incorrect classified examples

$$\min\frac{1}{2}\|\vec{w}\|^2+C\sum_i\xi_i$$

C is the trade-off between margin and the error

Dual formulation

$$\begin{cases} min \quad \frac{1}{2}||\vec{w}|| + C\sum_{i=1}^{m}\xi_i^2 \\ y_i(\vec{w}\cdot\vec{x_i}+b)\geq 1-\xi_i, \quad \forall i=1,..,m \\ \xi_i\geq 0, \quad i=1,..,m \end{cases}$$

$$L(\vec{w},b,\vec{\xi},\vec{\alpha})=\frac{1}{2}\vec{w}\cdot\vec{w}+\frac{C}{2}\sum_{i=1}^{m}\xi_i^2-\sum_{i=1}^{m}\alpha_i[y_i(\vec{w}\cdot\vec{x_i}+b)-1+\xi_i],$$

- By deriving wrt $\vec{w},\vec{\xi}$ and b

Partial Derivatives

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{\xi}} = C\vec{\xi} - \vec{\alpha} = \vec{0}$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial b} = \sum_{i=1}^{m} y_i \alpha_i = 0$$

Substitution in the objective function

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

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

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

- δ_{ij} of Kronecker

Final dual optimization problem

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

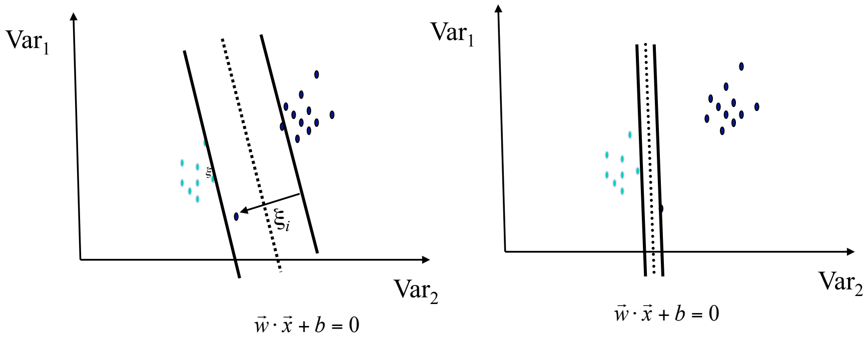
$$\alpha_i \geq 0, \quad \forall i = 1,..,m$$
$$\sum_{i=1}^{m} y_i \alpha_i = 0$$

Soft Margin Support Vector Machines

$$\min \frac{1}{2} \|\vec{w}\|^2 + C\sum_i \xi_i \qquad y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \quad \forall \vec{x}_i$$
$$\xi_i \geq 0$$

- The algorithm tries to keep ξ_i low and maximize the margin
- NB: The number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized
- If $C \to \infty$, the solution tends to the one of the *hard-margin* algorithm
- *Attention !!!:* if $C = 0$ we get $\|\vec{w}\| = 0$, since $y_i b \geq 1 - \xi_i \quad \forall \vec{x}_i$
- If C increases the number of error decreases. When C tends to infinite the number of errors must be 0, i.e. the *hard-margin* formulation

Robusteness of *Soft* vs. *Hard Margin* SVMs



Soft Margin SVM

Hard Margin SVM

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

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

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 $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$, *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 $\Rightarrow \exists$ 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
  - $\vec{\mathrm{v}}_t = (v_{ti})_{i=1}^{n}$ 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:

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

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

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

$$\Phi(d_x) = \vec{x} = (0,..,\underset{\text{buy}}{1},..,0,..,0,..,\underset{\text{acquisition}}{1},..,0,..,0,..,\underset{\text{stocks}}{1},..,0,..,0,..,\underset{\text{sell}}{1},..,0,..,\underset{\text{market}}{1})$$

$$\Phi(d_z) = \vec{z} = (0,..,\underset{\text{buy}}{1},..,0,..,\underset{\text{company}}{1},..,0,..,0,..,0,..,\underset{\text{stocks}}{1},..,0,..,0,..,\underset{\text{sell}}{1},..,0,..,0)$$

- 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

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

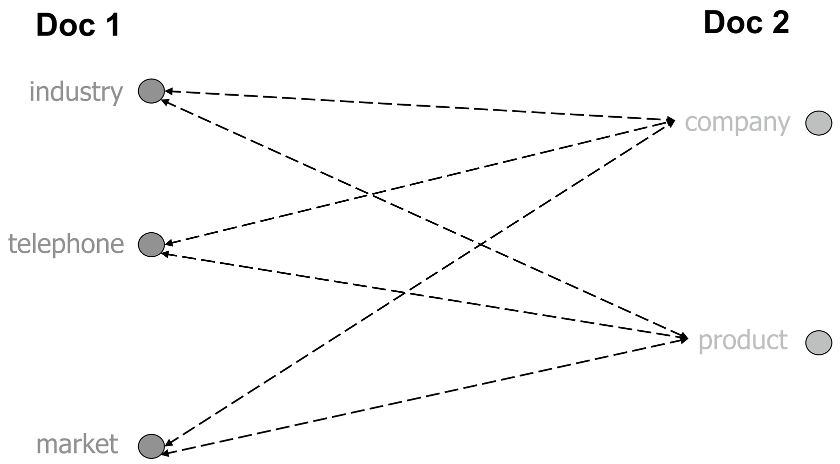
- More expressive, as (x_1x_2) encodes

  `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

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

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

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

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

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.

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

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 $\lambda^2 = \lambda^5$) 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 λ^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

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

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	$\lambda^7+\lambda^5+\lambda^4$

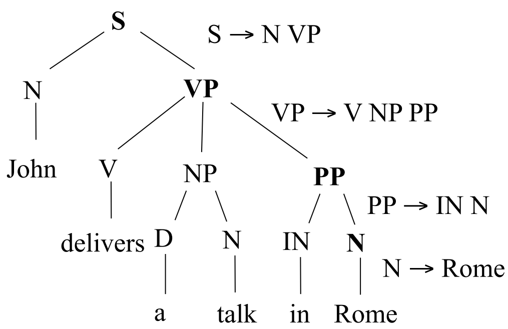
- The number (weight) of substrings of size 2 between "gat" and "cat" is $\lambda^4 = \lambda^2$ (["a","a"] entry of DP) x λ^2(cost of one character), where a = "t" and b = "t".
- Between "gatta" and "cata" is $\lambda^7 + \lambda^5 + \lambda^4$, i.e the matches of "a□□a", "t□a", "ta" with "a□a" and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

Tree kernels

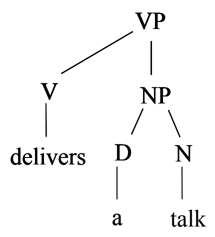
- Subtree, Subset Tree, Partial Tree kernels
- Efficient computation

Example of a parse tree

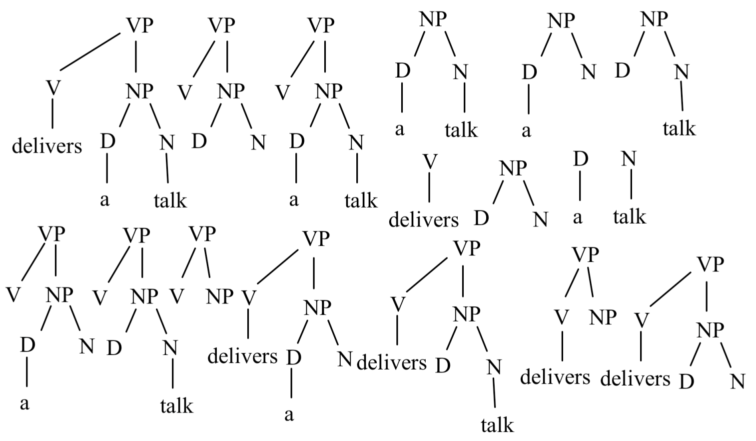
- “John delivers a talk in Rome”



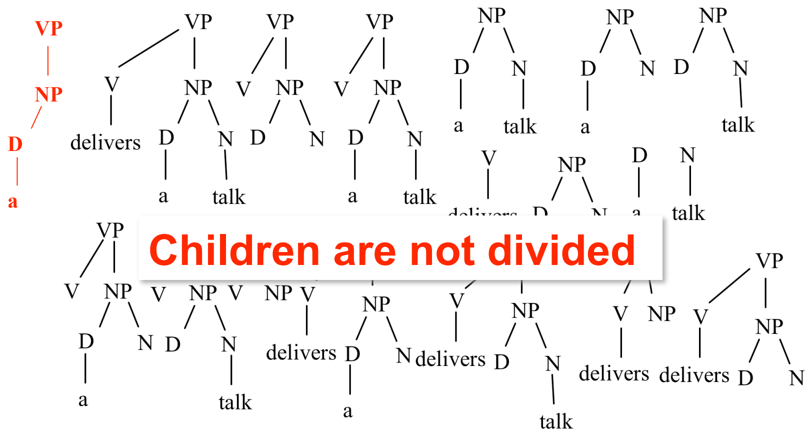
The Syntactic Tree Kernel (STK)
[Collins and Duffy, 2002]



The overall fragment set

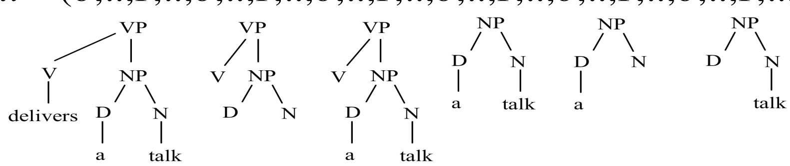


The overall fragment set

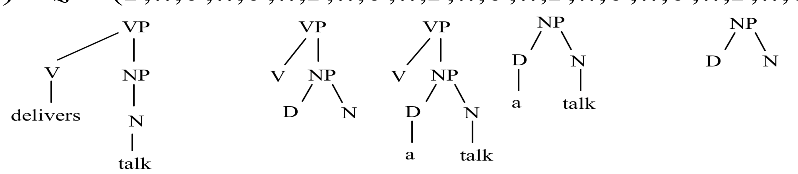


Explicit kernel space

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



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



- $\vec{x} \cdot \vec{z}$ 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 $O(n^2)$:

$$\Delta(n_x, n_z) = 0, \text{ if the productions are different else}$$
$$\Delta(n_x, n_z) = 1, \text{ if pre-terminals else}$$
$$\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))$$

Other Adjustments

- Decay factor

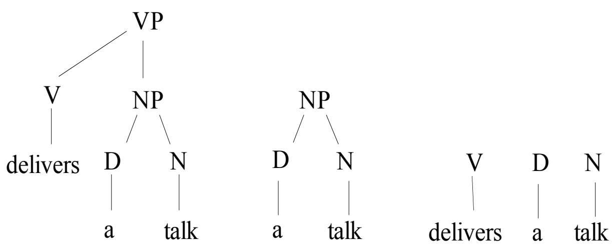
$$\Delta(n_x, n_z) = \lambda, \quad \text{if pre-terminals else}$$

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

- Normalization

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

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

$$\Delta(n_x, n_z) = 0, \text{ if the productions are different else}$$
$$\Delta(n_x, n_z) = 1, \text{ if pre - terminals else}$$
$$\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} \Delta(ch(n_x, j), ch(n_z, j))$$

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
```

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 ⇒ $O(|T_x|\times|T_z|)\ldots$

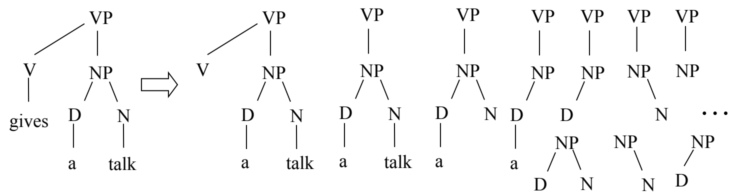
Observations

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

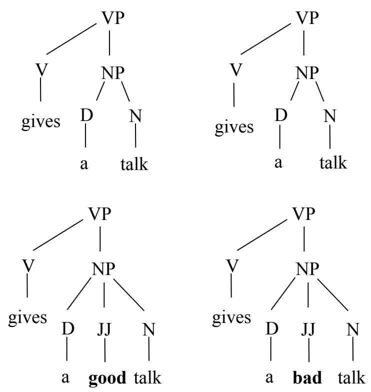
  $|T_x|+|T_z|$ *running time*
- If T_x and T_z are generated by only one production rule ⇒ $O(|T_x|\times|T_z|)\ldots$***Very Unlikely!!!!***

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]



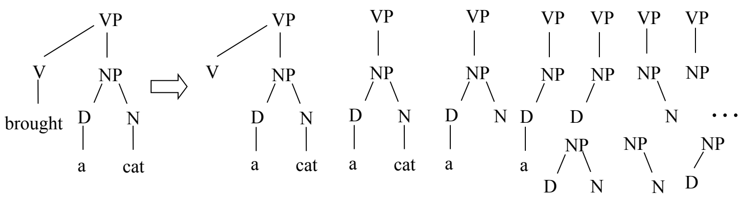
Weighting Problems



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

Partial Trees, [Moschitti, 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

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

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

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

Efficient Evaluation (2)

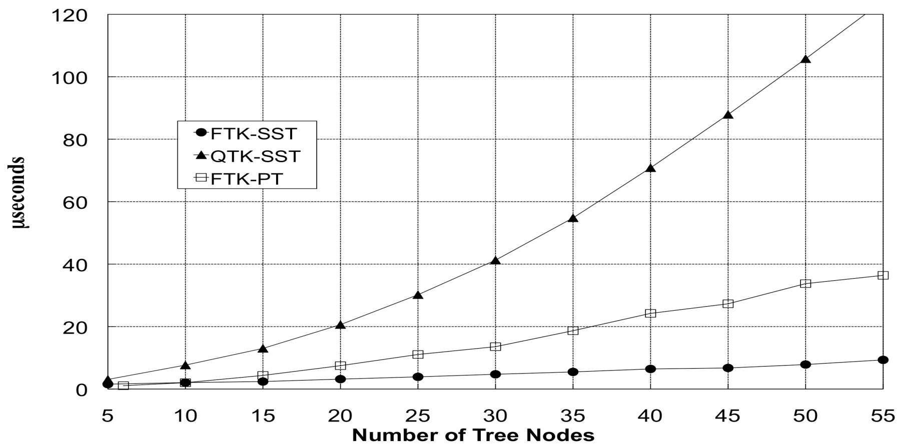
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) & \text{if } a = b; \\ 0 & otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

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

- 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, SST 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

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**

Part II: Kernel Methods for Practical Applications

Kernel Engineering approaches

- 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 [Moschitti et al, 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.

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 talk in Rome

Predicate Argument Classification

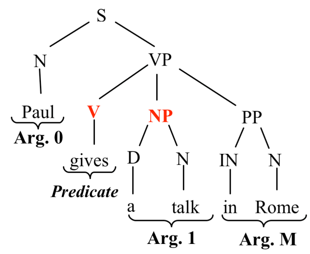
- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

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

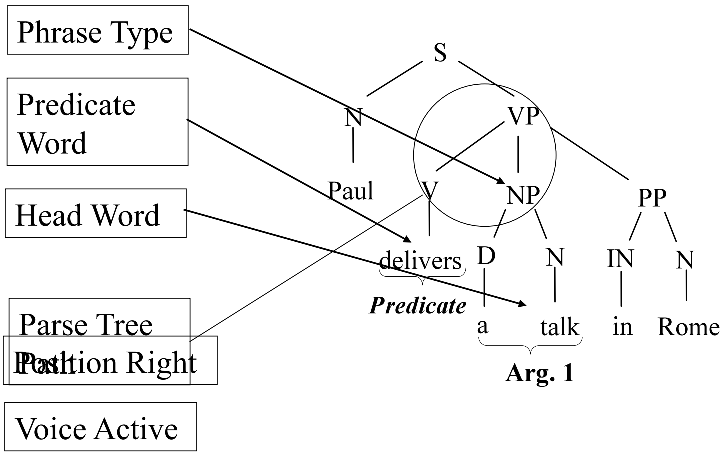
Predicate-Argument Feature Representation

Given a sentence, a predicate *p*:

1. Derive the sentence parse tree
2. For each node pair $<N_p,N_x>$
   a. Extract a feature representation set *F*
   b. If N_x exactly covers the Arg-*i*, *F* is one of its positive examples
   c. *F* is a negative example otherwise



Vector Representation for the linear kernel

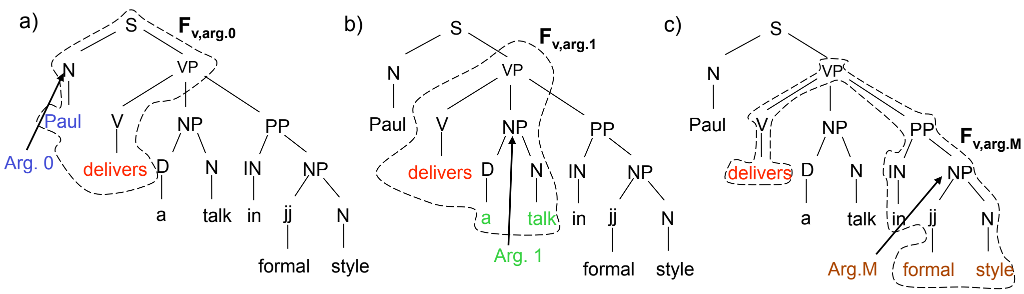


Kernel Engineering: Tree Tailoring

PAT Kernel [Moschitti, ACL 2004]

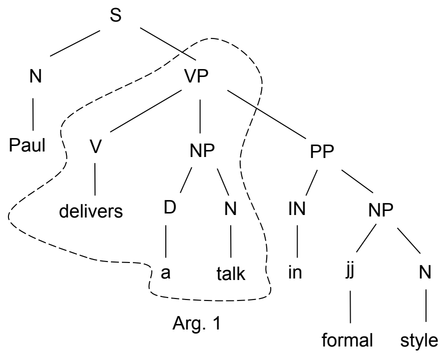
- Given the sentence:

[$_{Arg0}$ Paul] [$_{predicate}$ delivers] [$_{Arg1}$ a talk] [$_{ArgM}$ in formal Style]

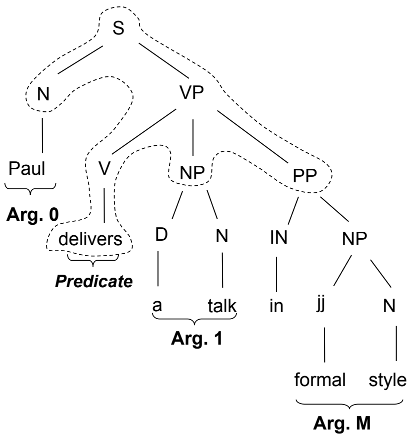


- These are Semantic Structures

In other words we consider…



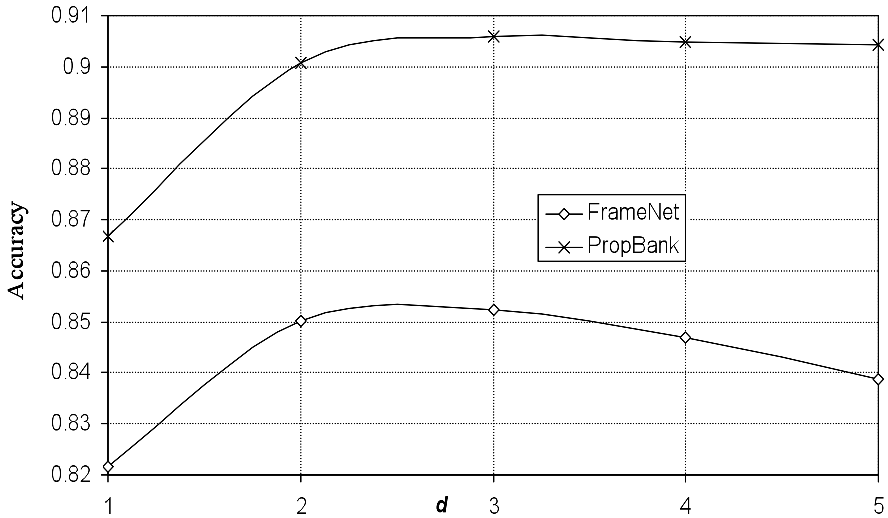
Sub-Categorization Kernel (SCF) [Moschitti, ACL 2004]



Experiments on Gold Standard Trees

- 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
- FrameNet and Collins' automatic trees
  - 24,558 sentences from the 40 frames of Senseval 3
  - 18 roles (same names are mapped together)
  - Only verbs
  - 70% for training and 30% for testing

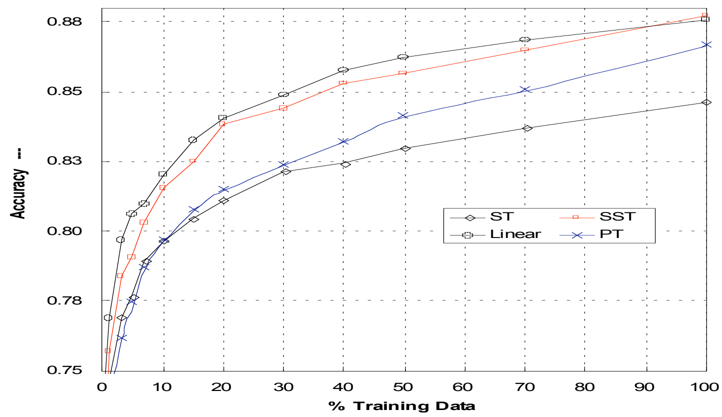
Argument Classification with Poly Kernel



PropBank Results

Args	P3	PAT	PAT+P	PAT×P	SCF+P	SCF×P
Arg0	90.8	88.3	92.6	90.5	94.6	94.7
Arg1	91.1	87.4	91.9	91.2	92.9	94.1
Arg2	80.0	68.5	77.5	74.7	77.4	82.0
Arg3	57.9	56.5	55.6	49.7	56.2	56.4
Arg4	70.5	68.7	71.2	62.7	69.6	71.1
ArgM	95.4	94.1	96.2	96.2	96.1	96.3
Global Accuracy	**90.5**	**88.7**	**91.3**	**90.4**	**92.4**	**93.2**

Argument Classification on PAT using different Tree Fragment Extractor



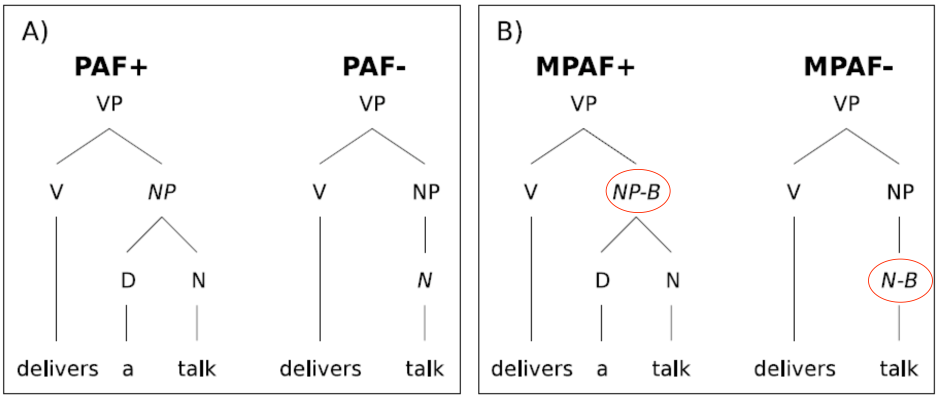
FrameNet Results

Roles	P3	PAF	PAF+P	PAF×P	SCF+P	SCF×P
agent	92.0	88.5	91.7	91.3	93.1	93.9
cause	59.7	16.1	41.6	27.7	42.6	57.3
degree	74.9	68.6	71.4	57.8	68.5	60.9
depictive	52.6	29.7	51.0	28.6	46.8	37.6
duration	45.8	52.1	40.9	29.0	31.8	41.8
goal	85.9	78.6	85.3	82.8	84.0	85.3
instrument	67.9	46.8	62.8	55.8	59.6	64.1
manner	81.0	81.9	81.2	78.6	77.8	77.8
Global Acc. (18 roles)	85.2	79.5	84.6	81.6	83.8	84.2

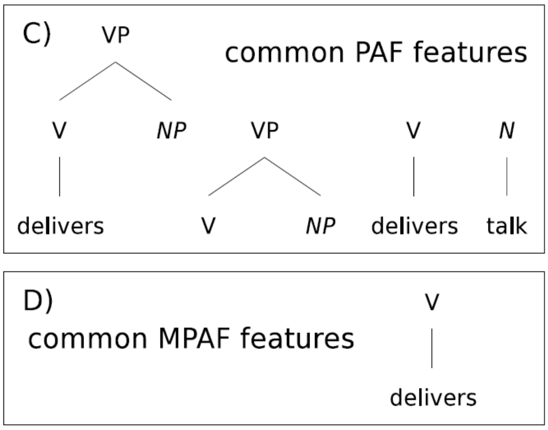
- ProbBank arguments vs. Semantic Roles

Kernel Engineering: Node marking

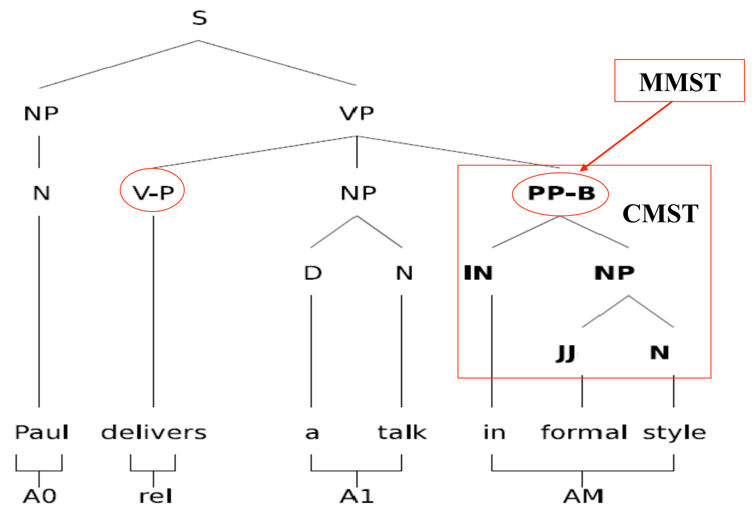
Marking Boundary nodes



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) [Moschitti et al, ACL-ws-2005]

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

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

- Syntactic/Semantic Tree Kernel
- Kernel Combinations
- Experiments

Merging of Kernels [Bloehdorn & Moschitti, ECIR 2007 & CIKM 2007]

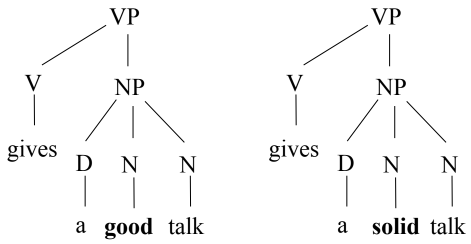
Definition 4 (Tree Fragment Similarity Kernel). *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[4]*:*

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

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

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 $\Delta(n_1, n_2) = \lambda \kappa_{\mathcal{S}}(ch^1_{n_1}, ch^1_{n_2})$,
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. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

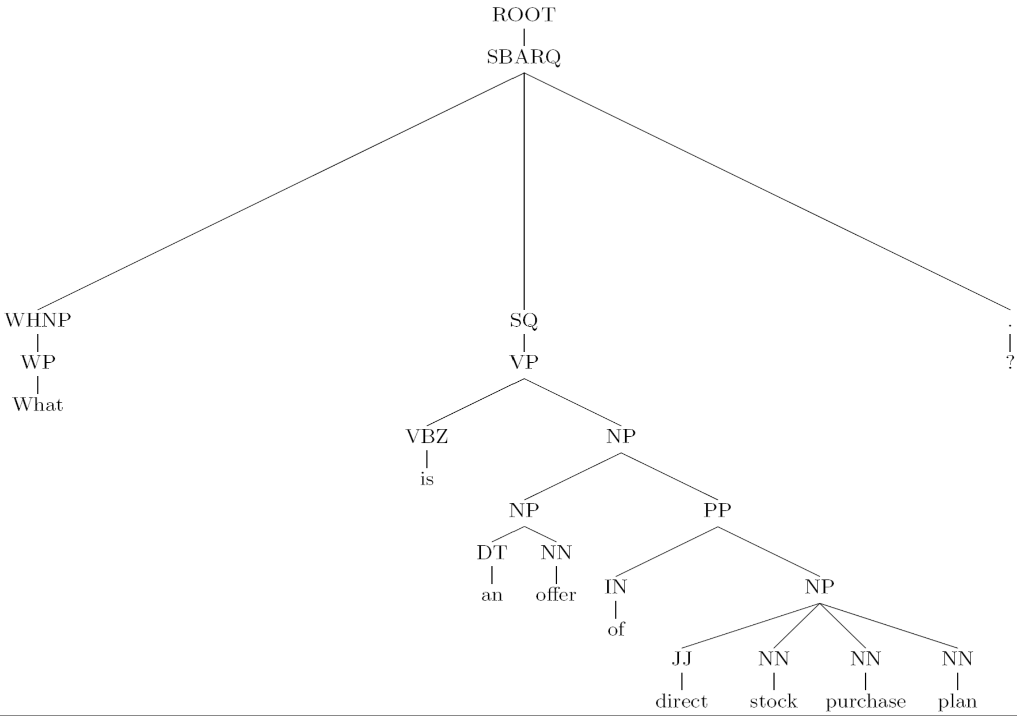
Question Classification

- **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

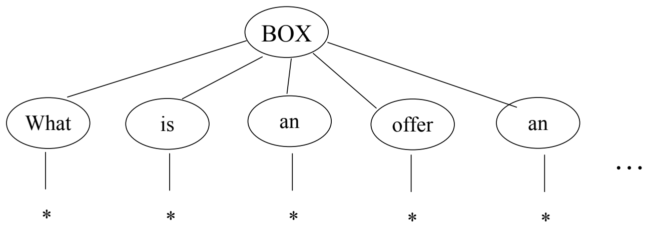
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Cross-validation (10-folds)
- Using the whole question parse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

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



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

Multiple Kernel Combinations

TASK: Question/Answer Classification [Moschitti, CIKM 2008]

- The classifier detects if a pair (question and answer) is correct or not
- A representation for the pair is needed
- The classifier can be used to re-rank the output of a basic QA system

Dataset 2: TREC data

- 138 TREC 2001 test questions labeled as "description"
- 2,256 sentences, extracted from the best ranked paragraphs (using a basic QA system based on Lucene search engine on TREC dataset)
- 216 of which labeled as correct by one annotator

Dataset 2: TREC data

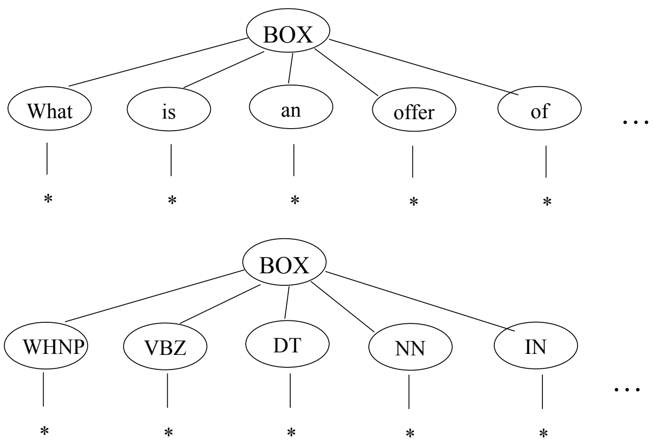
- 138 TREC 2001 test questions labeled as "description"

A question is linked to many answers: all its derived pairs cannot be shared by training and test sets

- 216 of which labeled as correct by one annotator

Bags of words (BOW) and POS-tags (POS)

- To save time, apply STK to these trees:

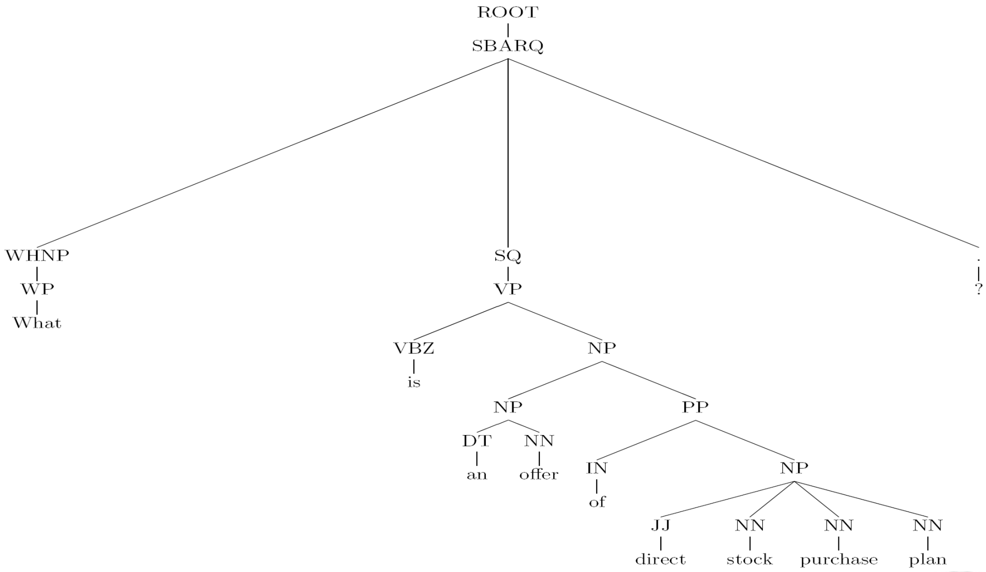


Word and POS Sequences

- What is an offer of…? (word sequence, **WSK**)
  - ➔ `What_is_offer`
  - ➔ `What_is`

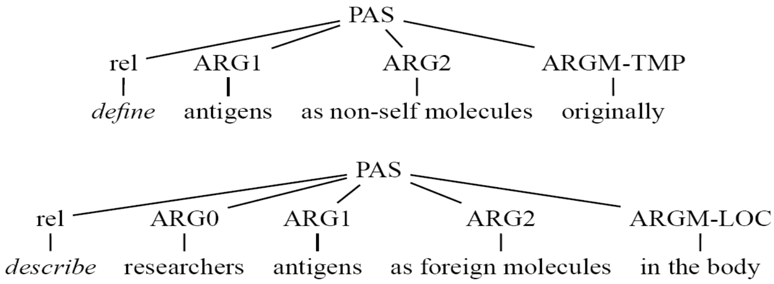
- WHNP VBZ DT NN IN…(POS sequence, **POSSK**)
  - ➔ `WHNP_VBZ_NN`
  - ➔ `WHNP_NN_IN`

Syntactic Parse Trees (PT)



Predicate Argument Structure for Partial Tree Kernel (PAS_{PTK})

- [**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]

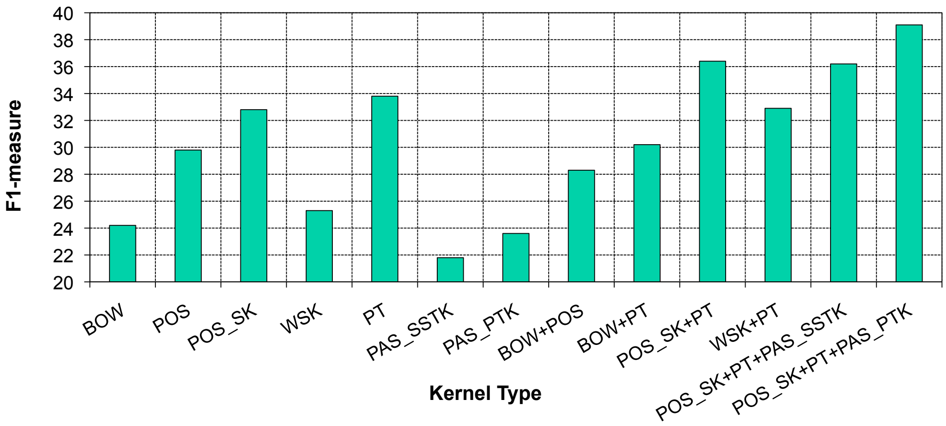


Kernels and Combinations

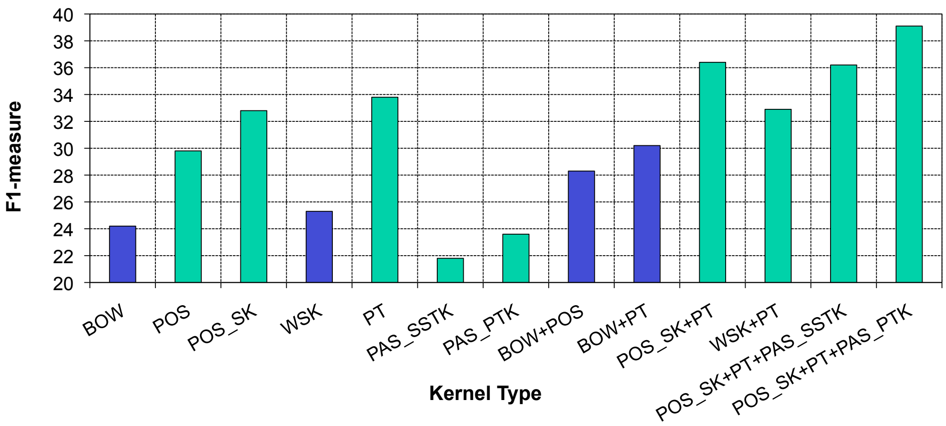
- Exploiting the property: $k(x,z) = k_1(x,z)+k_2(x,z)$
- BOW, POS, WSK, POSSK, PT, PAS_{PTK}

⇒ BOW+POS, BOW+PT, PT+POS, …

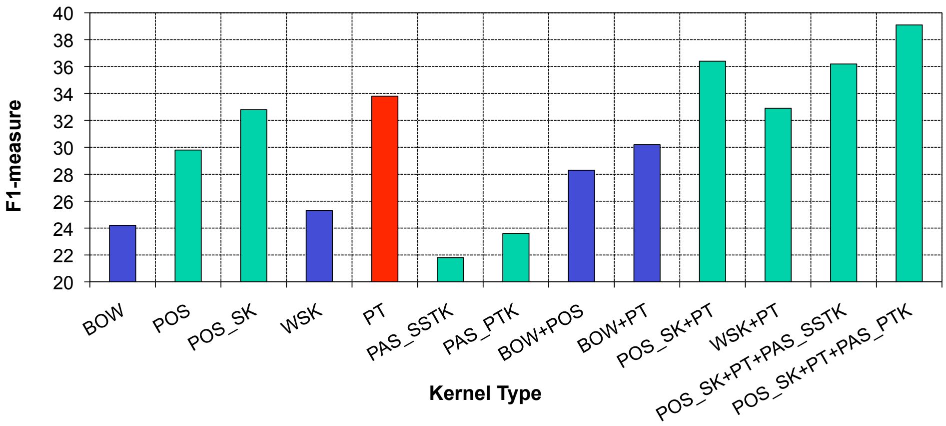
Results on TREC Data
(5 folds cross validation)



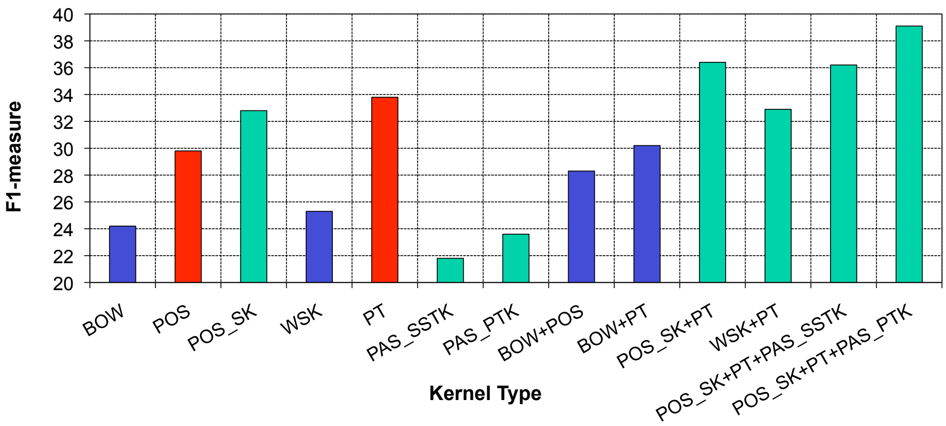
Results on TREC Data
(5 folds cross validation)



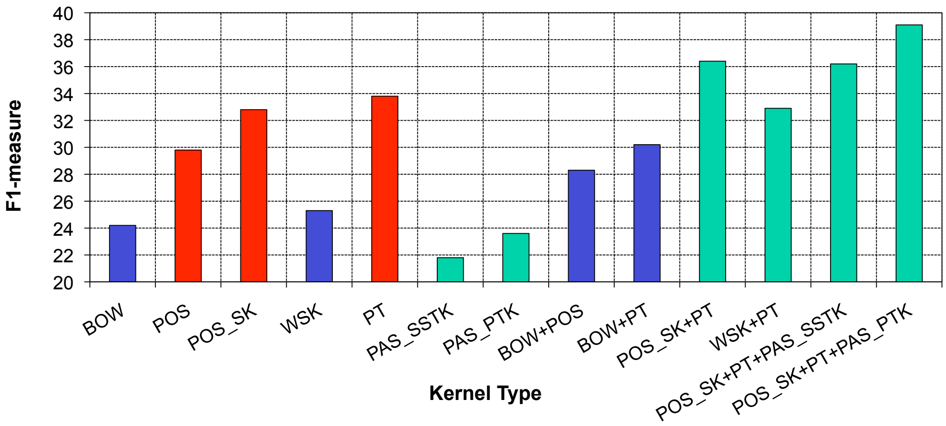
Results on TREC Data
(5 folds cross validation)



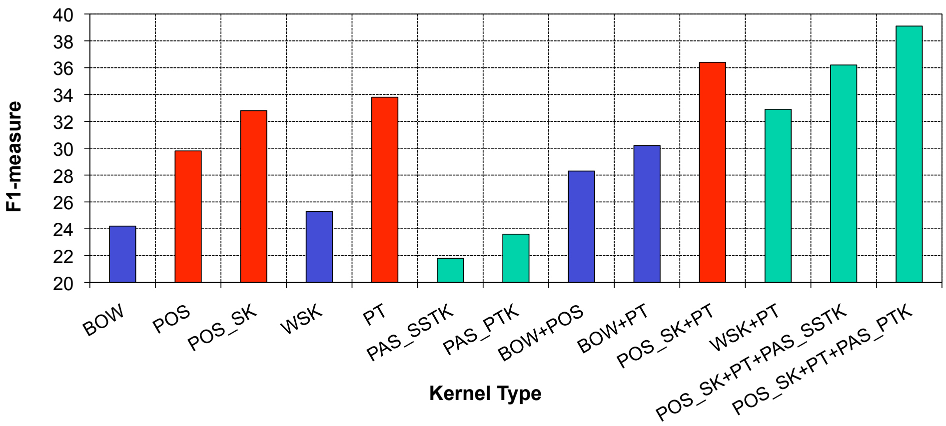
Results on TREC Data
(5 folds cross validation)



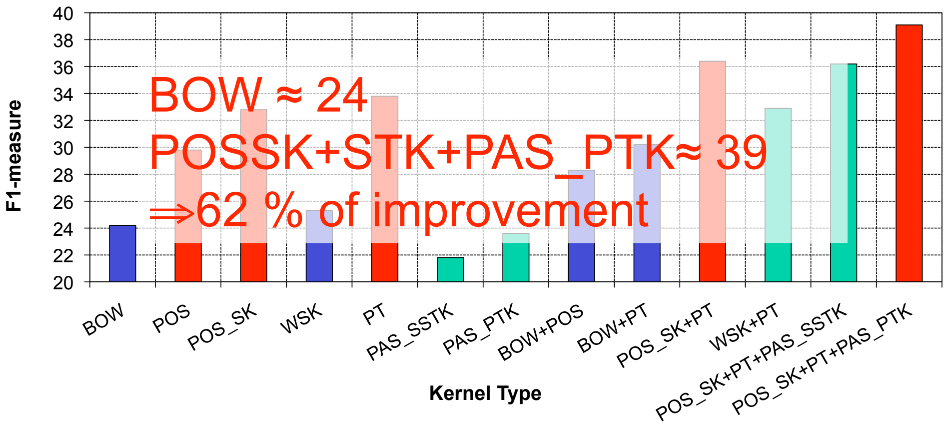
Results on TREC Data
(5 folds cross validation)



Results on TREC Data
(5 folds cross validation)



Results on TREC Data
(5 folds cross validation)

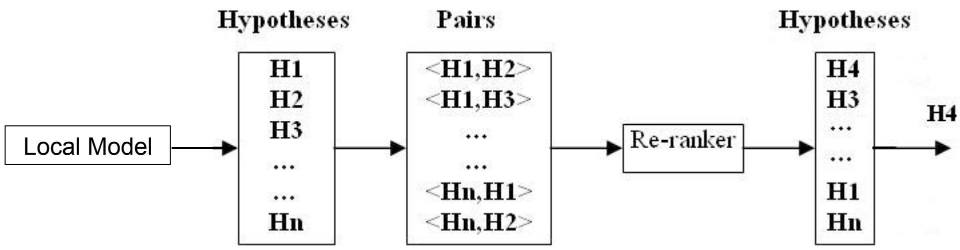


Kernels for Re-ranking

Re-ranking Framework

- Local classifier generates the most likely set of hypotheses.
- These are used to build annotation pairs, $\langle h^i, h^j \rangle$.
  - positive instances if h^i *more correct* than h^j,
- A binary classifier decides if h^i is more accurate than h^j.
- Each candidate annotation h^i is described by a structural representation

Re-ranking framework



Syntactic Parsing Re-ranking

- Pairs of parse trees (Collins and Duffy, 2002)

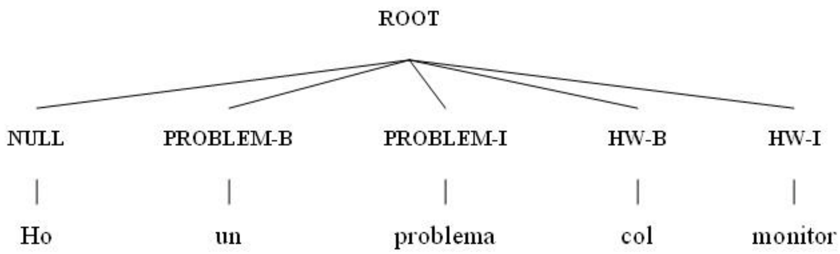
Re-ranking concept labeling
[Dinarelli et al, 2009]

- *I have a problem with my monitor*

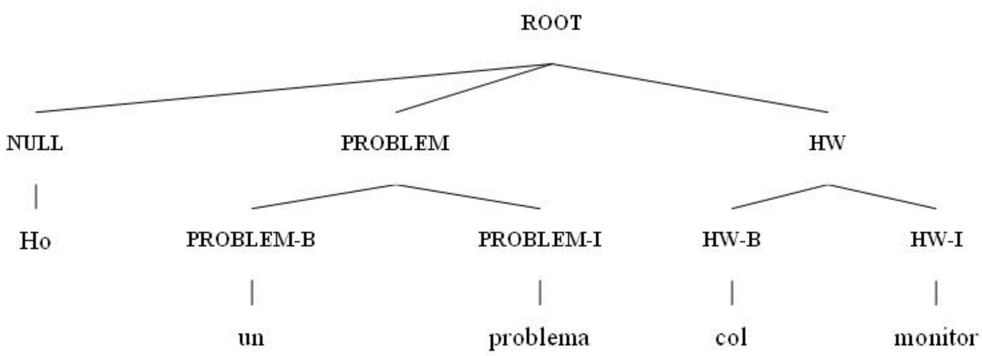
h^i: *I* **NULL** *have* **NULL** *a* **NULL** *problem* **PROBLEM-B** *with* **NULL** *my* **NULL** *monitor* **HW-B**

h^j: *I* **NULL** *have* **NULL** *a* **NULL** *problem* **HW-B** *with* **NULL** *my* **NULL** *monitor*

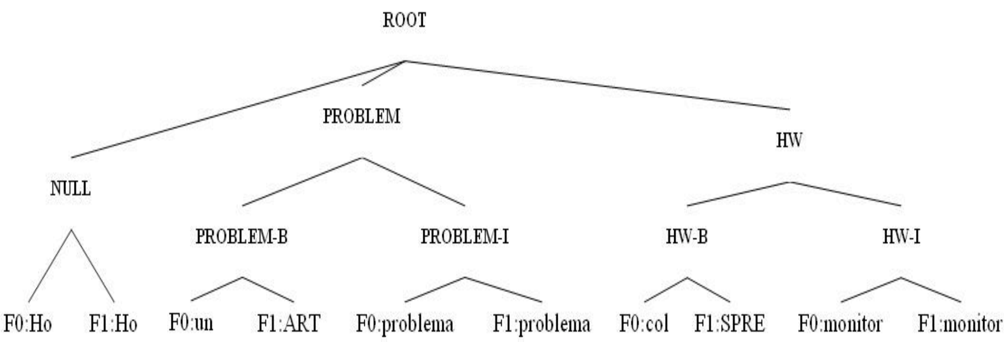
Flat tree representation (cross-language structure)



Multilevel Tree

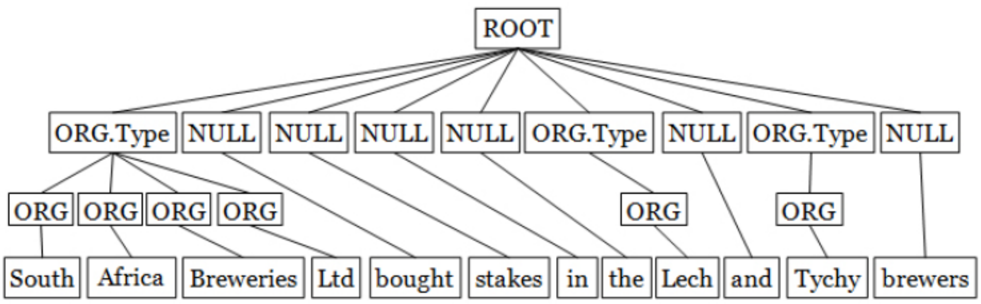


Enriched Multilevel Tree



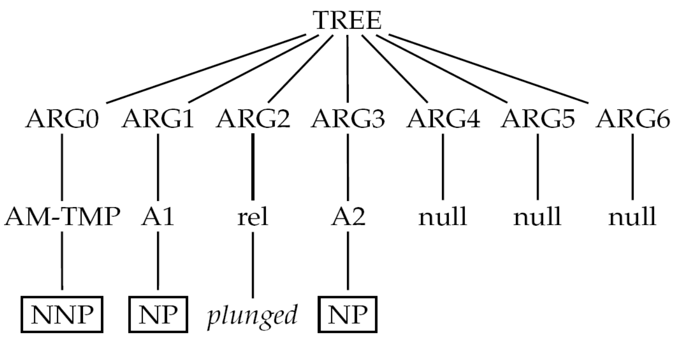
- FST CER from 23.2 to 16.01

Re-ranking for Named-Entity Recognition [Vien et al, 2010]



- CRF F1 from 84.86 to 88.16

Re-ranking Predicate Argument Structures
[Moschitti et al, CoNLL 2006]



- SVMs F1 from 75.89 to 77.25

Conclusions

- Kernel methods and SVMs are useful tools to design language applications
- Kernel design still requires some level of expertise
- Engineering approaches to tree kernels
  - Basic Combinations
  - Canonical Mappings, e.g.
    - Node Marking
  - Merging of kernels in more complex kernels
- Easy modeling produces state-of-the-art accuracy in many tasks, RTE, SRL, QC, NER, RE
- SVM-Light-TK efficient tool to use them

Future (on going work)

- Once we have found the right kernel, are we satisfied?
- What about knowing the most relevant features?
- Can we speed up learning/classification at real-application scenario level?
- The answer is reverse kernel engineering:
  - [Pighin&Moschitti, CoNLL2009, EMNLP2009, CoNLL2010]
  - Mine the most relevant fragments according to SVMs gradient
  - Use the linear space
- Software for reverse kernel engineering available in the next months

Thank you

References

- Alessandro Moschitti and Silvia Quarteroni, *Linguistic Kernels for Answer Re-ranking in Question Answering Systems,* Information and Processing Management, ELSEVIER, 2010.
- Yashar Mehdad, Alessandro Moschitti and Fabio Massimo Zanzotto. *Syntactic/ Semantic Structures for Textual Entailment Recognition*. Human Language Technology - North American chapter of the Association for Computational Linguistics (HLT-NAACL), 2010, Los Angeles, Calfornia.
- Daniele Pighin and Alessandro Moschitti. *On Reverse Feature Engineering of Syntactic Tree Kernels*. In Proceedings of the 2010 Conference on Natural Language Learning, Upsala, Sweden, July 2010. Association for Computational Linguistics.
- Thi Truc Vien Nguyen, Alessandro Moschitti and Giuseppe Riccardi. *Kernel-based Reranking for Entity Extraction.* In proceedings of the 23rd International Conference on Computational Linguistics (COLING), August 2010, Beijing, China.

References

- Alessandro Moschitti. *Syntactic and semantic kernels for short text pair categorization*. In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 576–584, Athens, Greece, March 2009. Association for Computational Linguistics.
- Truc-Vien Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. *Convolution kernels on constituent, dependency and sequential structures for relation extraction*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1378–1387, Singapore, August 2009. Association for Computational Linguistics.
- Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. *Re-ranking models based-on small training data for spoken language understanding*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1076–1085, Singapore, August 2009. Association for Computational Linguistics.
- Alessandra Giordani and Alessandro Moschitti. *Syntactic Structural Kernels for Natural Language Interfaces to Databases*. In ECML/PKDD, pages 391–406, Bled, Slovenia, 2009.

References

- Alessandro Moschitti, Daniele Pighin and Roberto Basili. *Tree Kernels for Semantic Role Labeling,* Special Issue on Semantic Role Labeling, Computational Linguistics Journal. March 2008.
- Fabio Massimo Zanzotto, Marco Pennacchiotti and Alessandro Moschitti*, A Machine Learning Approach to Textual Entailment Recognition,* Special Issue on Textual Entailment Recognition, Natural Language Engineering, Cambridge University Press., 2008
- Mona Diab, Alessandro Moschitti, Daniele Pighin, *Semantic Role Labeling Systems for Arabic Language using Kernel Methods*. In proceedings of the 46th Conference of the Association for Computational Linguistics (ACL'08). Main Paper Section. Columbus, OH, USA, June 2008.
- Alessandro Moschitti, Silvia Quarteroni, Kernels on Linguistic Structures for Answer Extraction. In proceedings of the 46th Conference of the Association for Computational Linguistics (ACL'08). Short Paper Section. Columbus, OH, USA, June 2008.

References

- Yannick Versley, Simone Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang and Alessandro Moschitti, *BART: A Modular Toolkit for Coreference Resolution*, In Proceedings of the Conference on Language Resources and Evaluation, Marrakech, Marocco, 2008.
- Alessandro Moschitti, *Kernel Methods, Syntax and Semantics for Relational Text Categorization*. In proceeding of ACM 17th Conference on Information and Knowledge Management (CIKM). Napa Valley, California, 2008.
- Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. *Shallow semantic parsing for spoken language understanding*. In Proceedings of HLT-NAACL Short Papers, pages 85–88, Boulder, Colorado, June 2009. Association for Computational Linguistics.
- 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).

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

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, Giuseppe Riccardi, Christian Raymond, *Spoken Language Understanding with Kernels for Syntactic/Semantic Structures*, Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU2007), Kyoto, Japan, December 2007
- Stephan Bloehdorn and Alessandro Moschitti, *Combined Syntactic and Semantic Kernels for Text Classification*, to appear in the 29th European Conference on Information Retrieval (ECIR), April 2007, Rome, Italy.
- Stephan Bloehdorn, Alessandro Moschitti: Structure and semantics for expressive text kernels. In proceeding of ACM 16th Conference on Information and Knowledge Management (CIKM-short paper) 2007: 861-864, Portugal.

References

- 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.
- Alessandro Moschitti, *Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees*. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
- Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti, *Fast On-line Kernel Learning for Trees*, International Conference on Data Mining (ICDM) 2006 (short paper).
- Stephan Bloehdorn, Roberto Basili, Marco Cammisa, Alessandro Moschitti, *Semantic Kernels for Text Classification based on Topological Measures of Feature Similarity*. In Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06), Hong Kong, 18-22 December 2006. (short paper).

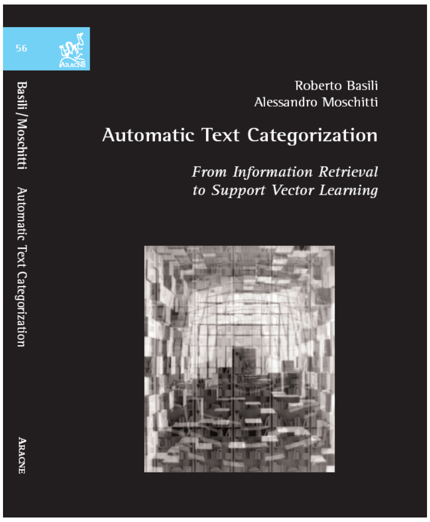
References

- Roberto Basili, Marco Cammisa and Alessandro Moschitti, *A Semantic Kernel to classify texts with very few training examples*, in Informatica, an international journal of Computing and Informatics, 2006.
- Fabio Massimo Zanzotto and Alessandro Moschitti, *Automatic learning of textual entailments with cross-pair similarities*. In Proceedings of COLING-ACL, Sydney, Australia, 2006.
- Ana-Maria Giuglea and Alessandro Moschitti, *Semantic Role Labeling via FrameNet, VerbNet and PropBank.* 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.

References

- 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
- Alessandro Moschitti, *A study on Convolution Kernel for Shallow Semantic Parsing*. In proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004), Barcelona, Spain, 2004.
- Alessandro Moschitti and Cosmin Adrian Bejan, *A Semantic Kernel for Predicate Argument Classification.* In proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004), Boston, MA, USA, 2004.

An introductory book on SVMs, Kernel methods and Text Categorization



Non-exhaustive reference list from other authors

- V. Vapnik. The Nature of Statistical Learning Theory. Springer, 1995.
- P. Bartlett and J. Shawe-Taylor, 1998. Advances in Kernel Methods - Support Vector Learning, chapter Generalization Performance of Support Vector Machines and other Pattern Classifiers. MIT Press.
- David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Dept. of Computer Science, University of California at Santa Cruz.
- Lodhi, Huma, Craig Saunders, John Shawe Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. JMLR,2000
- Schölkopf, Bernhard and Alexander J. Smola. 2001. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA.

Non-exhaustive reference list from other authors

- N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines (and other kernel-based learning methods)* Cambridge University Press, 2002
- M. Collins and N. Duffy, New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In ACL02, 2002.
- Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In Proceedings of ICML'02.
- S.V.N. Vishwanathan and A.J. Smola. Fast kernels on strings and trees. In Proceedings of NIPS, 2002.
- Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. Journal of Machine Learning Research, 3:1059–1082. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. JMLR, 3:1083–1106, 2003.

Non-exhaustive reference list from other authors

- Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In Proceedings of ACL'03.
- Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In Proceedings of SIGIR'03, pages 26–32.
- Libin Shen, Anoop Sarkar, and Aravind k. Joshi. Using LTAG Based Features in Parse Reranking. In Proceedings of EMNLP'03, 2003
- C. Cumby and D. Roth. Kernel Methods for Relational Learning. In Proceedings of ICML 2003, pages 107–114, Washington, DC, USA, 2003.
- J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.
- A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In Proceedings of the 42nd Annual Meeting on ACL, Barcelona, Spain, 2004.

Non-exhaustive reference list from other authors

- Kristina Toutanova, Penka Markova, and Christopher Manning. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In Proceedings of EMNLP 2004.
- Jun Suzuki and Hideki Isozaki. 2005. Sequence and Tree Kernels with Statistical Feature Mining. In Proceedings of NIPS'05.
- Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting based parse reranking with subtree features. In Proceedings of ACL'05.
- R. C. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In Proceedings of NIPS, 2005.
- R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In Proceedings of EMNLP, pages 724–731, 2005.
- S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In Proceedings of the 43rd Meeting of the ACL, pages 419–426, Ann Arbor, Michigan, USA, 2005.

Non-exhaustive reference list from other authors

- J. Kazama and K. Torisawa. Speeding up Training with Tree Kernels for Node Relation Labeling. In Proceedings of EMNLP 2005, pages 137–144, Toronto, Canada, 2005.
- M. Zhang, J. Zhang, J. Su, , and G. Zhou. A composite kernel to extract relations between entities with both flat and structured features. In Proceedings of COLING-ACL 2006, pages 825–832, 2006.
- M. Zhang, G. Zhou, and A. Aw. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Information Processing and Management, 44(2):825–832, 2006.
- G. Zhou, M. Zhang, D. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In Proceedings of EMNLP-CoNLL 2007, pages 728–736, 2007.

Non-exhaustive reference list from other authors

- Ivan Titov and James Henderson. Porting statistical parsers with data-defined kernels. In Proceedings of CoNLL-X, 2006
- Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In Proceedings of NAACL.
- M. Wang. A re-examination of dependency path kernels for relation extraction. In Proceedings of the 3rd International Joint Conference on Natural Language Processing-IJCNLP, 2008.

Author Index