Outline: Part I – Kernel Machines

- Outline and Motivation
- Kernel Machines
  - Perceptron
  - Support Vector Machines
  - Kernel Definition (Kernel Trick)
  - Mercer's Conditions
  - Kernel Operators
  - Efficiency issue: when can we use kernels?

Outline: Part I – Basic Kernels

- Basic Kernels and their Feature Spaces
  - Linear Kernels
  - Polynomial Kernels
  - Lexical Semantic Kernels
  - String and Word Sequence Kernels
  - Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK

Outline: Part I – Classification with Kernels

- Classification with Kernels
  - Question Classification (QC) using constituency, dependency and semantic structures
  - Question Classification (QC) in Jeopardy!
  - Relation Extraction
  - Coreference Resolution

Outline: Part II – Kernels for Ranking

- Reranking with kernels
  - Classification of Question/Answer (QA) pairs
  - Preference Reranking Kernel
  - Reranking NLP tasks
    - Named Entities in a Sentence
    - Predicate Argument Structures
    - Segmentation and labeling of Speech Transcriptions
  - Reranking the output of a hierarchical text classifier
  - Reranking Passages with relational representations: the IBM Watson system case

Outline: Part III – Advanced Topics

- Large-scale learning with kernels
  - Cutting Plane Algorithm for SVMs
  - Sampling methods (uSVMs)
  - Compacting space with DAGs
- Reverse Kernel Engineering
  - Model linearization
  - Question Classification
- Conclusions and Future Directions

Motivation (1)

- Structures and Semantics more and more important in IR
- Applying off-the-shelf Natural Language Processing (NLP) tools is not enough:
  - How to exploit linguistic and semantic information?
  - Definition of rules and heuristics for exploiting such information
- An alternative effective solution is the use of Machine Learning (ML), e.g., learning to rank algorithms
  - Training data is required:
    - Query logs, crow-sourcing, skilled annotators
  - **ML feature design: which solution?**

Motivation (2)

- Feature design is the most difficult aspect in designing an ML system
- Complex and difficult task, e.g., in case of structural feature representation:
  - Deep knowledge and intuitions are required
  - Design problems when the phenomenon is described by many features

Motivation (3)

- 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

Motivation (4)

- High accuracy especially for new applications and new domains
  - Manual engineering still poor, e.g., Arabic SRL
- Inherent higher accuracy when many structural patterns are needed, e.g., for Relation Extraction
- Fast prototyping and adaptation for new domains and applications
- The major contribution of kernels is to make system modeling easier

What can really kernels do?

- Optimistic view:
  - Better feature spaces not manually designable
  - The overall feature space produced by kernel is essential for a given task
  - Features impractical to be manually designed
- Bottom-line view
  - Faster feature engineering approach
  - Higher-level feature engineering, e.g., structures instead of vector components
  - Towards automatic feature engineering
  - Structures are more meaningful when inspected

Why and when using kernels?

- Using them is very simple: much simpler than feature vectors
- They are like any other machine learning approach simply better than feature vectors
- Small training data: absolutely no reason for not using them
  - Many features provide back-off models
  - Structural features provide domain adaptation
- Large training data: new methods enable them
  - using large data many features become important
  - kernels become very effective

Part I – Kernel Machines

- Kernel Machines (30 min)
  - Perceptron
  - Support Vector Machines
  - Kernel Definition (Kernel Trick)
  - Mercer's Conditions
  - Kernel Operators
  - Efficiency issue: when can we use kernels?

Binary Classification Problem (on text)

- Given:
  - a category: C
  - and a set *T* of documents,

  define

$$f : T \rightarrow \{C, \bar{C}\}$$

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

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 market sell stocks trade

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

buy company sell stock

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

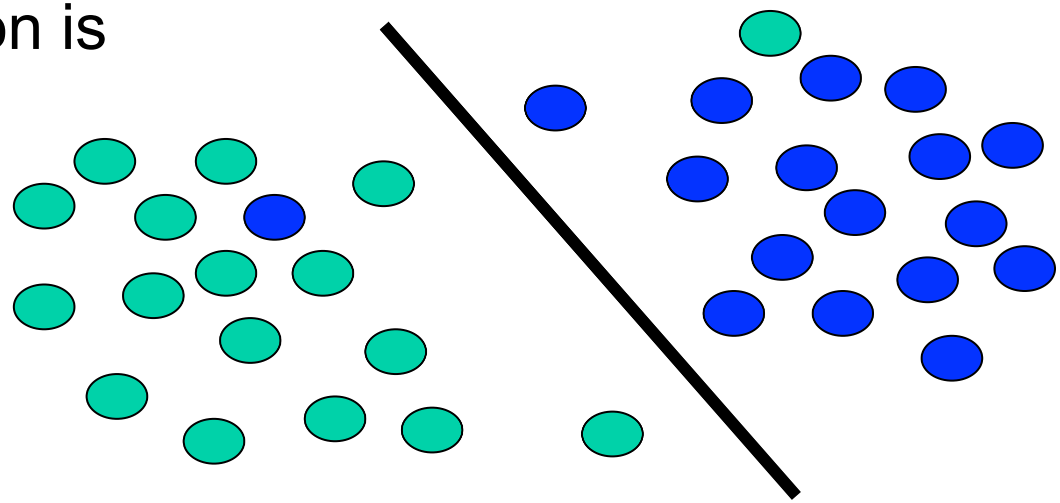
Linear Classifier

- 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

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



Note that the hyperplane classifier is just: $\vec{d} \cdot \vec{C} > th$

An example of 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

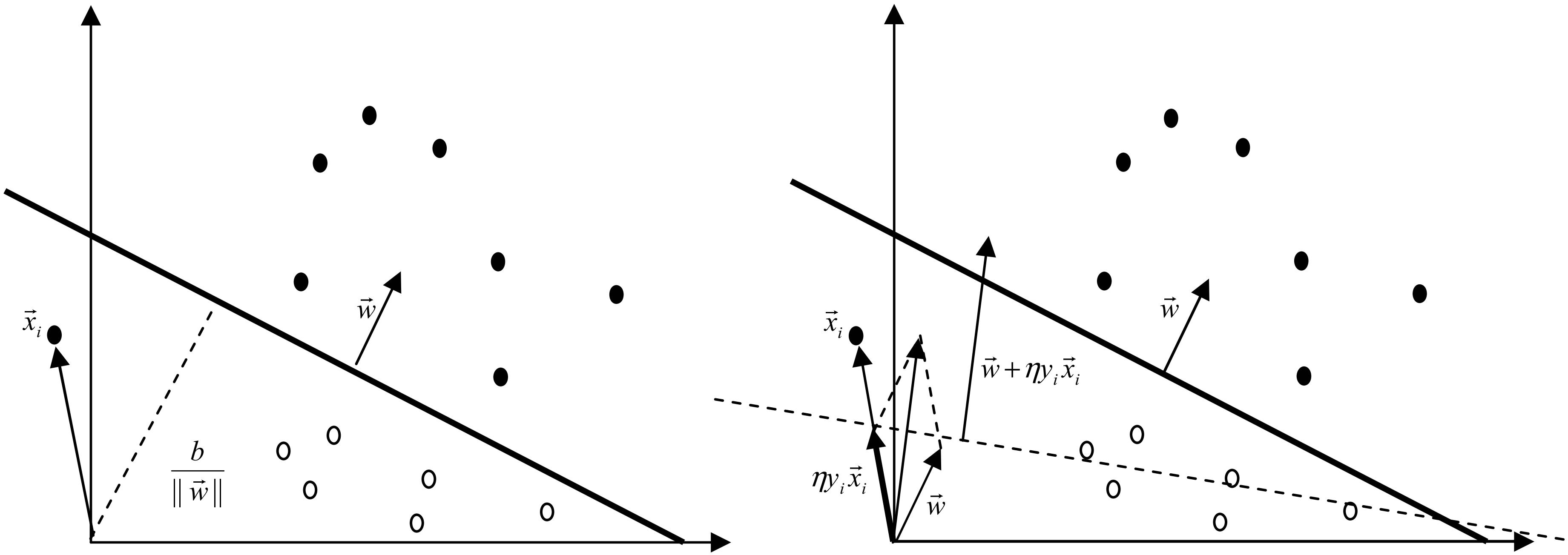
   endif

  endfor

while an error is found

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

Graphic interpretation of the Perceptron



Dual Representation for Classification

- In each step of perceptron algorithm only training data is added with a certain weight:

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

- Hence the classification function results:

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

- Note that data only appears in the scalar product

Dual Representation for Learning

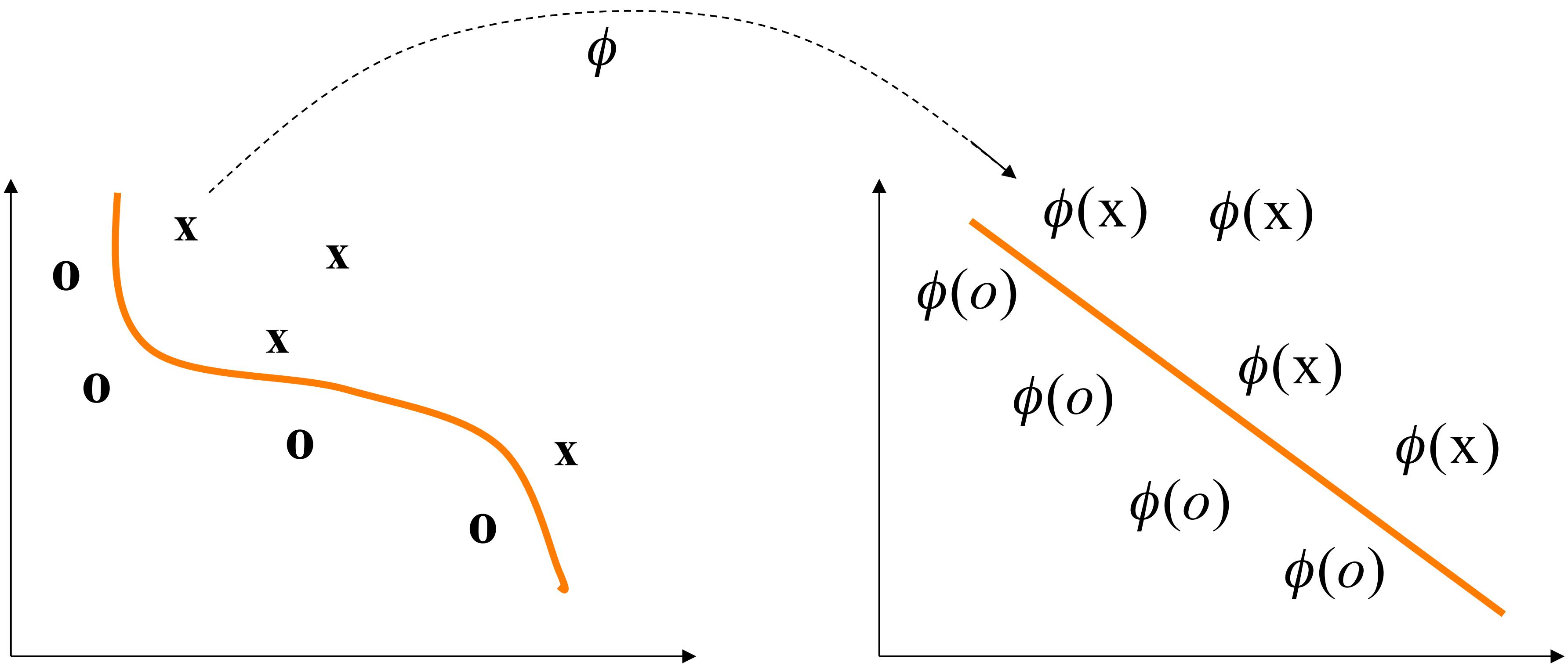
- as well as the updating function

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

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

The main idea of Kernel Functions

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



Dual Perceptron algorithm and kernel functions

- In the space ϕ, we can rewrite the classification function as:

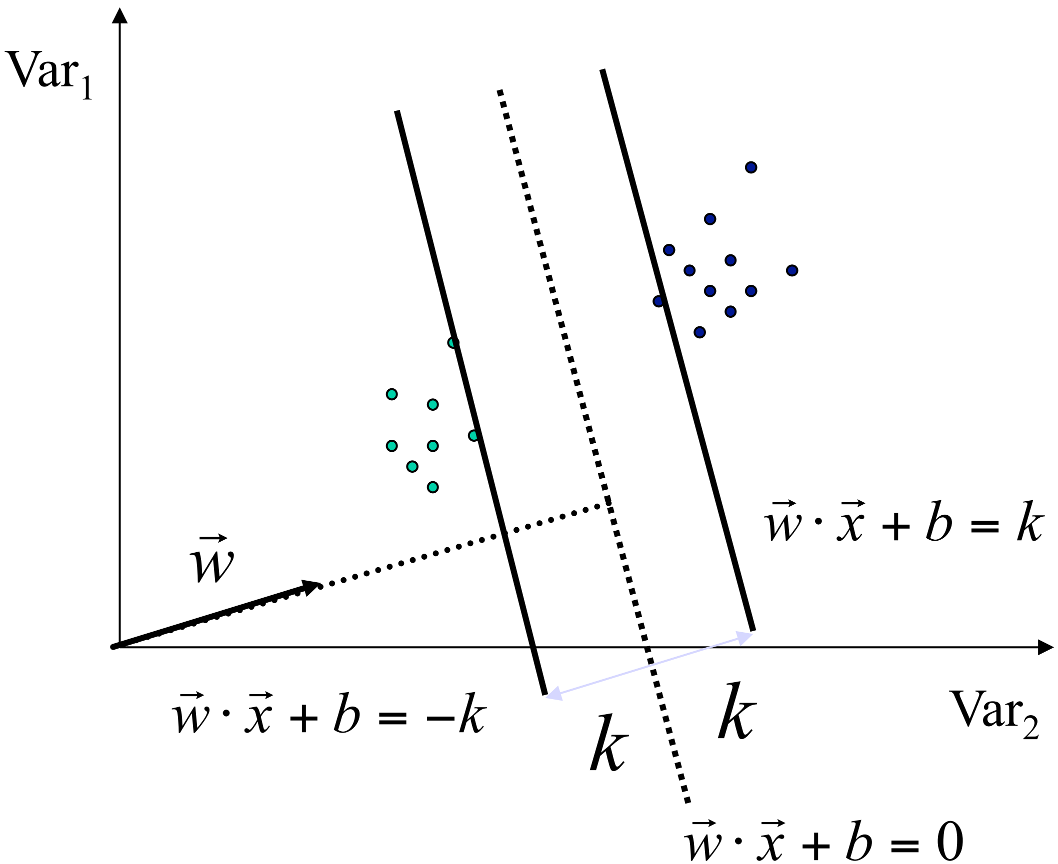
$$h(x) = \operatorname{sgn}(\vec{w}_\phi \cdot \phi(\vec{x}) + b_\phi) =$$

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

- As well as the updating function

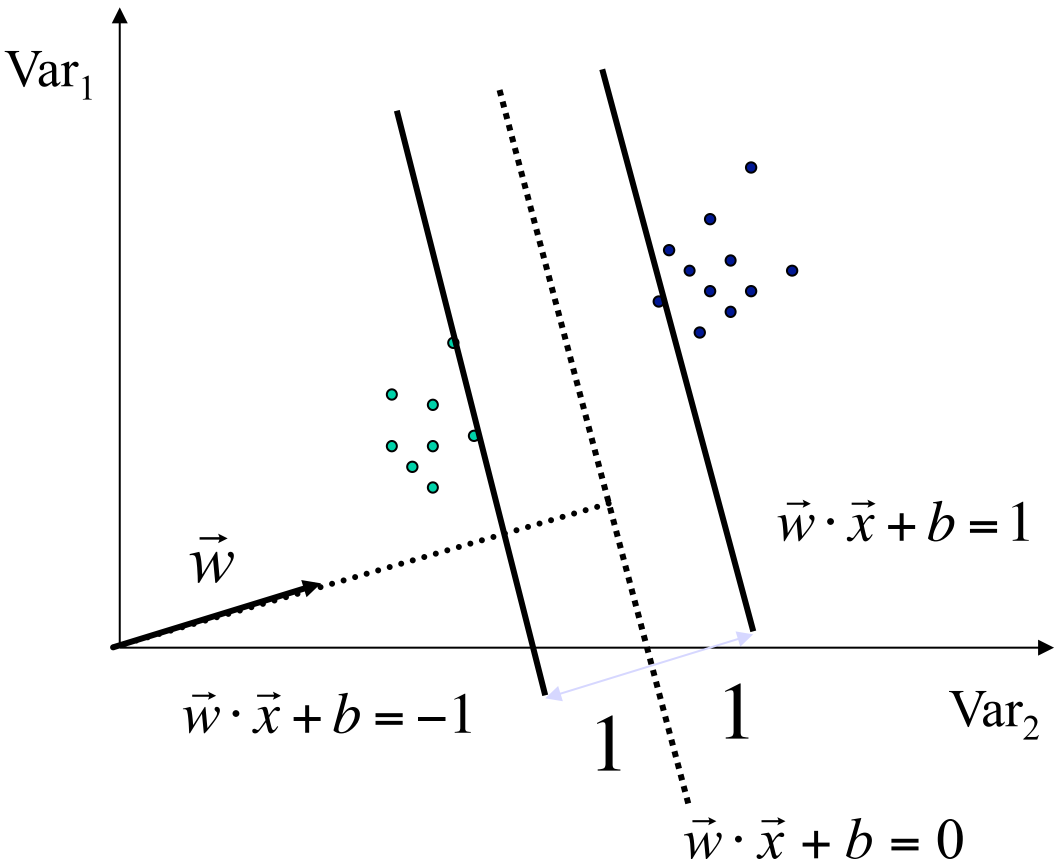
$$\text{if } y_i \left(\sum_{j=1..\ell} \alpha_j y_j k(\vec{x}_j, \vec{x}_i) + b_\phi \right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

Support Vector Machines



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

Support Vector Machines



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

We need to solve

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

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

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

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

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

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

Soft Margin optimization problem

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

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

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

Kernels in Support Vector Machines

- In Soft Margin SVMs we maximize:

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

- By using kernel functions we rewrite the problem as:

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

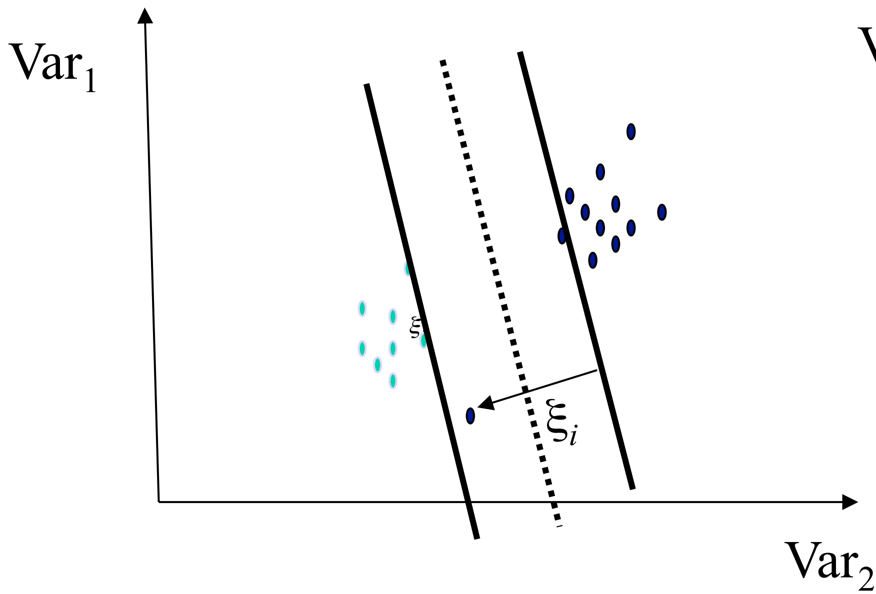
Soft Margin Support Vector Machines

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

$$\xi_i \geq 0$$

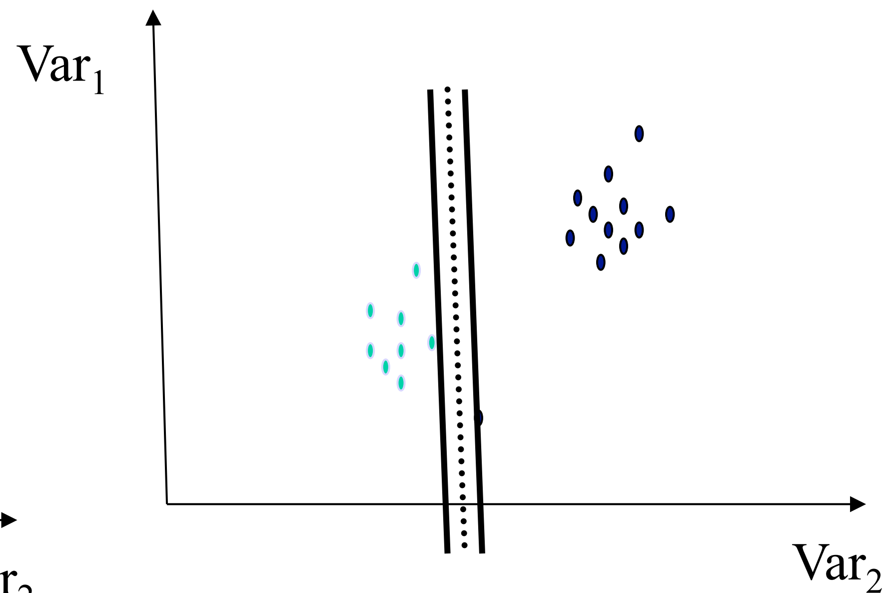
- The algorithm tries to keep ξ_i low and maximize the margin
- NB: the distances from the hyperplane are minimized; the number of error is not directly minimized (NP-complete problem)
- If $C \rightarrow \infty$, the solution tends to the one of the *hard-margin* algorithm
  - If C increases the number of error decreases. When C tends to infinite the number of errors must be 0, i.e. the *hard-margin* formulation

Trade-off between Generalization and Empirical Error



$\vec{w} \cdot \vec{x} + b = 0$

Soft Margin SVM



$\vec{w} \cdot \vec{x} + b = 0$

Hard Margin SVM

Parameters

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

$$= \min \frac{1}{2} \| \vec{w} \|^2 + C \left(J \sum_i \xi_i^+ + \sum_i \xi_i^- \right)$$

- C: trade-off parameter
- J: cost factor

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

- The **sole** information used for training is the kernel Gram matrix

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

- If the kernel is valid, K is symmetric positive-semidefinite

Valid Kernels

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

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

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

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

Valid Kernels cont'd

Proposition 1. *(Mercer's conditions)*
Let X be a finite input space and let $K(\boldsymbol{x}, \boldsymbol{z})$ be a symmetric function on X. Then $K(\boldsymbol{x}, \boldsymbol{z})$ is a kernel function if and only if the matrix

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

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
  - $\vec{\mathrm{v}}_t = \left(v_{ti}\right)_{i=1}^{n}$ are the eigenvectors, i.e. the columns of V
- Let us assume lambda values non-negative

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

Mercer's Theorem (sufficient conditions)

- Therefore

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

- which implies that K is a kernel function

Mercer's Theorem (necessary conditions)

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

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

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

this contradicts the geometry of the space.

Is it a valid kernel?

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

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

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

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

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

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. String and Tree Kernels

Part I – Basic Kernels (for structured data)

- Basic Kernels and their Feature Spaces (35 min)
  - Linear Kernels
  - Polynomial Kernels
  - Lexical Semantic Kernels
  - String and Word Sequence Kernels
  - Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK

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 market sell stocks trade

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

buy company sell stock

- 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

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

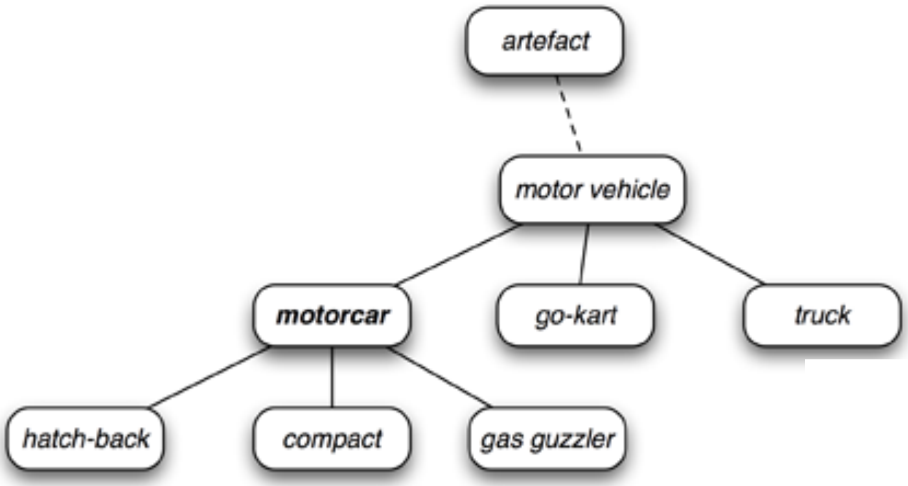
- More expressive, as (x_1x_2) encodes

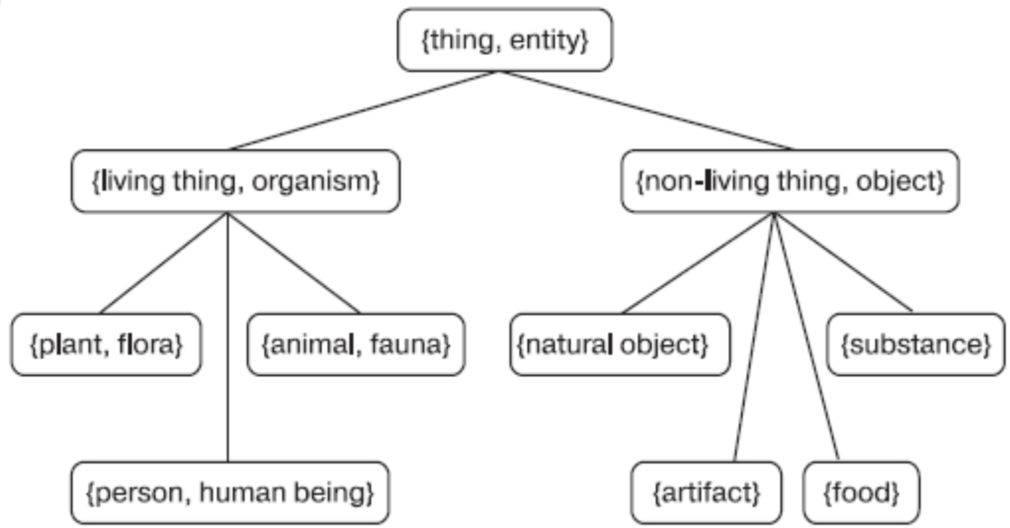
  `Stock+Market` vs. **`Downtown+Market`** features

- We can smartly compute the scalar product as

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

Sub-hierarchies in WordNet





Similarity based on WordNet

Inverted Path Length:

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

Wu & Palmer:

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

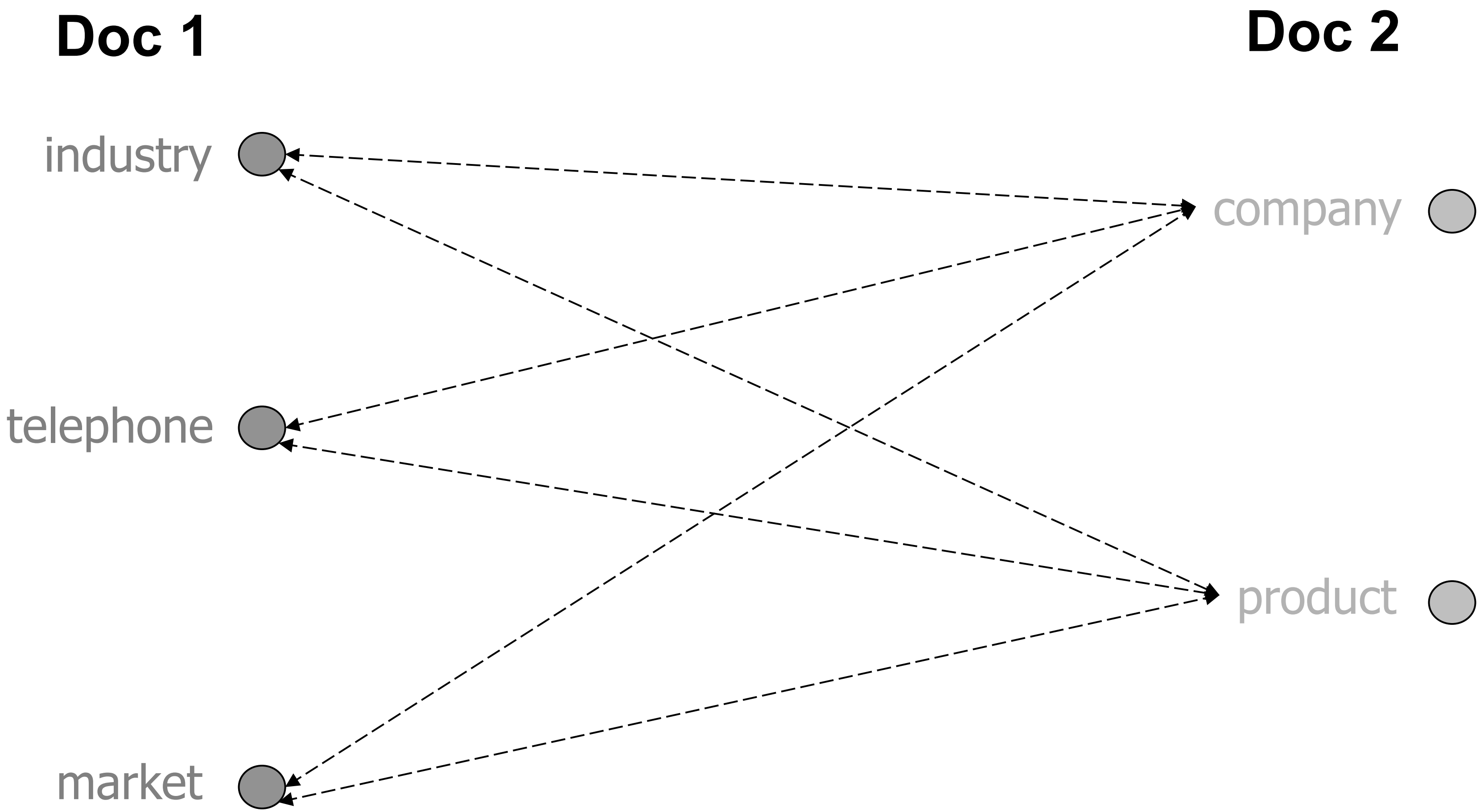
Resnik:

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

Lin:

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

Document Similarity



Lexical Semantic Kernels

- The document similarity is the following 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

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

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

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

Formal Definition

$$s = s_1, .., s_{|s|}, \quad \vec{I} = (i_1, ..., i_{|u|})$$

$$u = s[\vec{I}]$$

$$\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})}$$, 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

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

$$K(\mathrm{Bank}, \mathrm{Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3) \cdot (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)$$

$$= 3\lambda^2 + 2\lambda^4 + 2\lambda^6$$

Efficient Evaluation: Intuition

- Dynamic Programming technique over:
  - The size of the two input strings, *m*, *n* and
  - The size of their common substrings, *p*
- Evaluate the spectrum string kernels
  - Substrings of size *p*
- Sum the contribution of the different *p* spectra

Efficient Evaluation

Given two sequences $s_1 a$ and $s_2 b$, 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.

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

D_p satisfies the recursive relation:

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

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

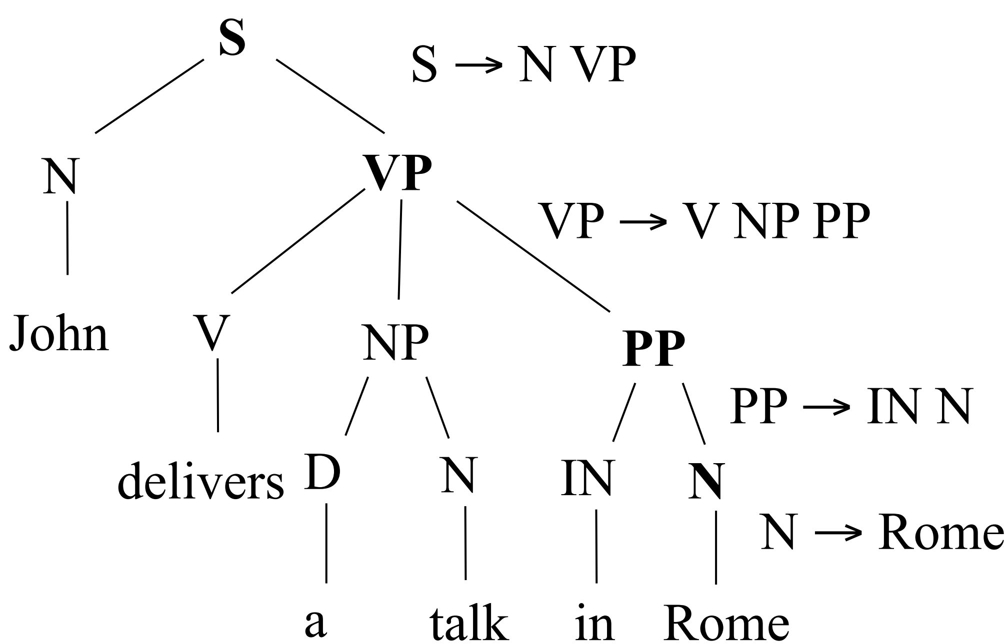
$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r])$$

Tree kernels

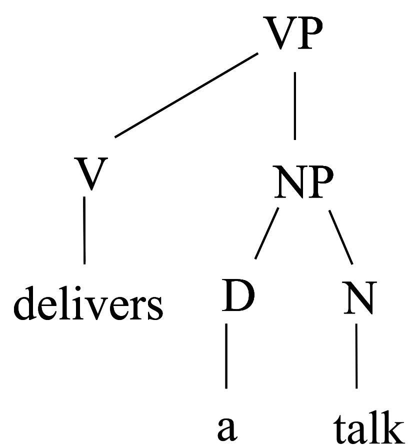
- Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK
- Efficient computation

Example of a parse tree

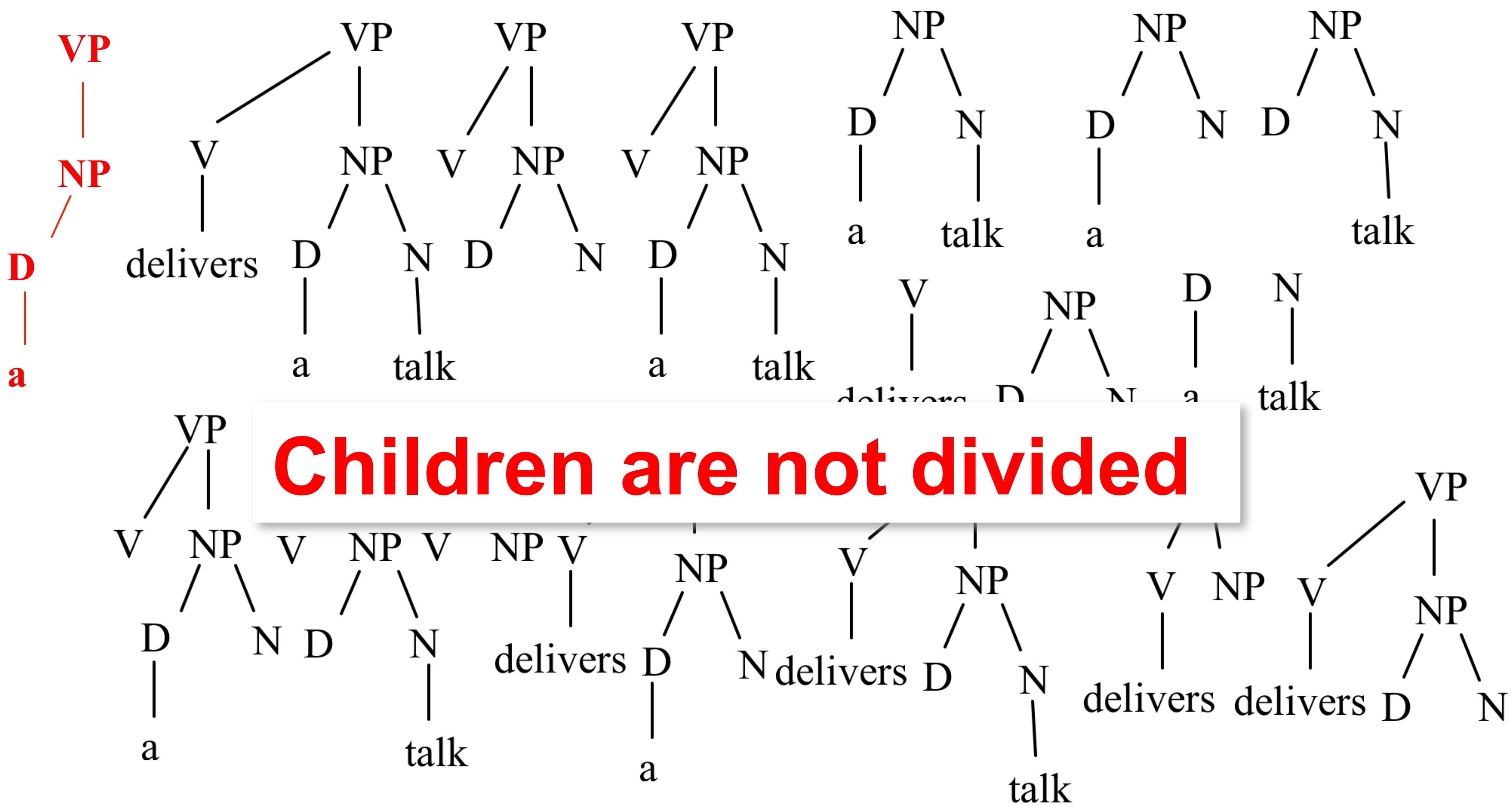
- "John delivers a talk in Rome"



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

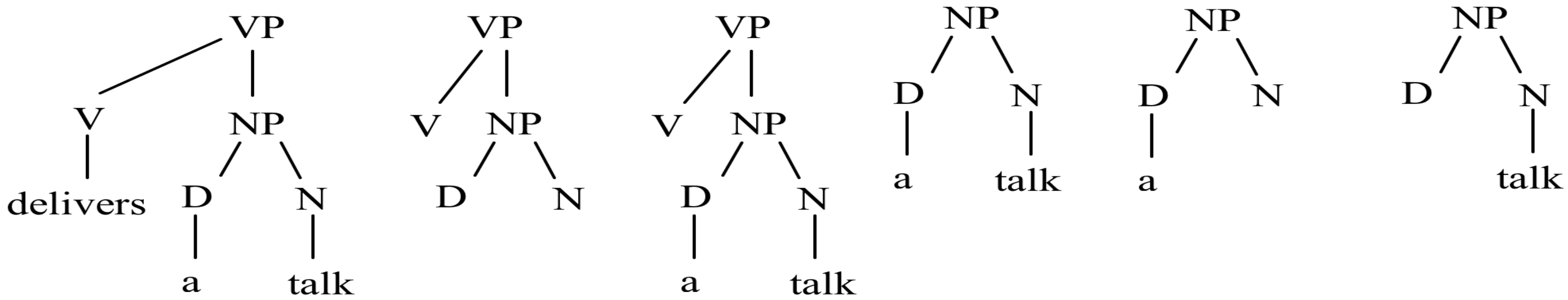


The overall fragment set

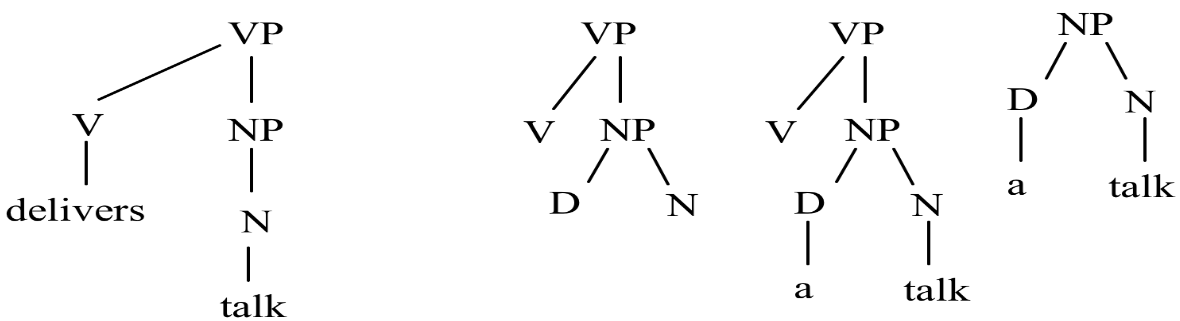


Explicit kernel space

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



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



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

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

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

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

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

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

Observations

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

  $|T_x|+|T_z|$ *running time* [Moschitti, EACL 2006]
- If T_x and T_z are generated by only one production rule $\Rightarrow$ $O(|T_x|\times|T_z|)\ldots$***Very Unlikely!!!!***

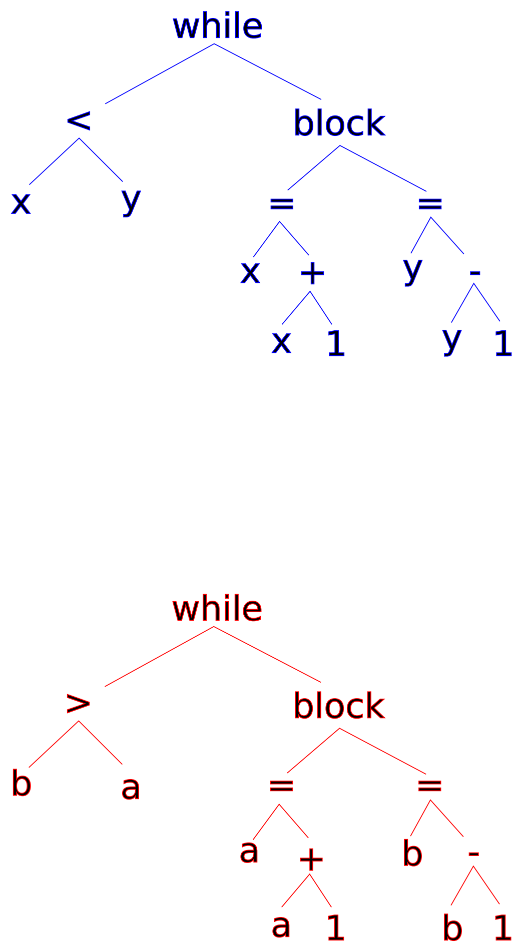
Trees can also be program derivation trees

CODE

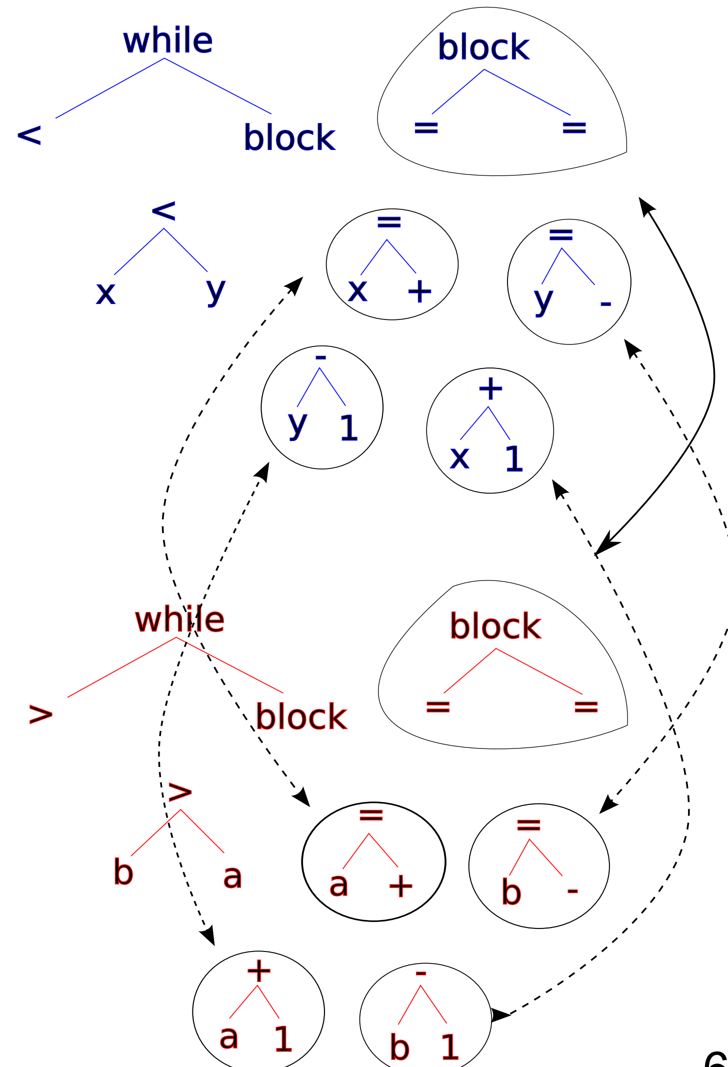
```
while (x < y) {
   x = x + 1
   y = y - 1
}
```

```
while (b > a) {
   a = a + 1
   b = b - 1
}
```

AST

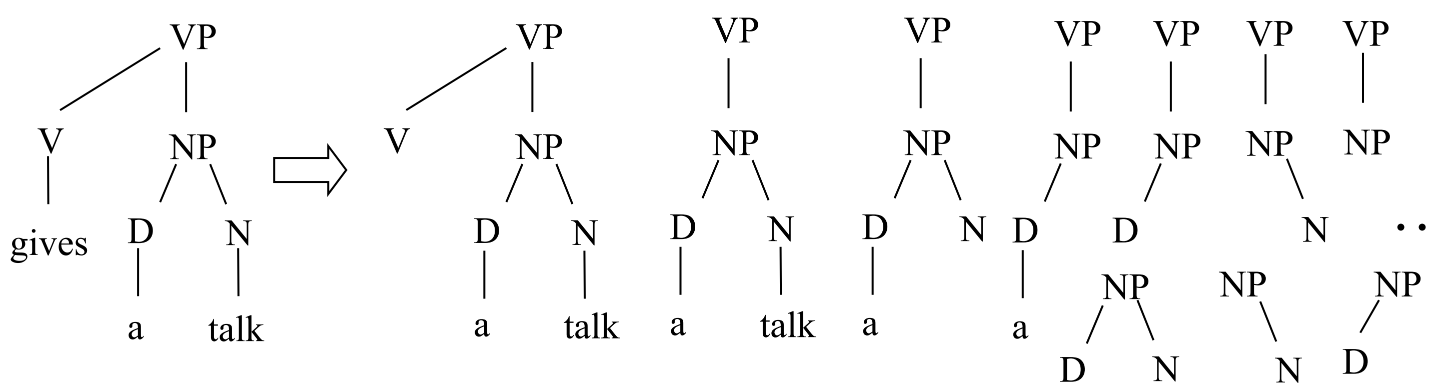


AST KERNEL

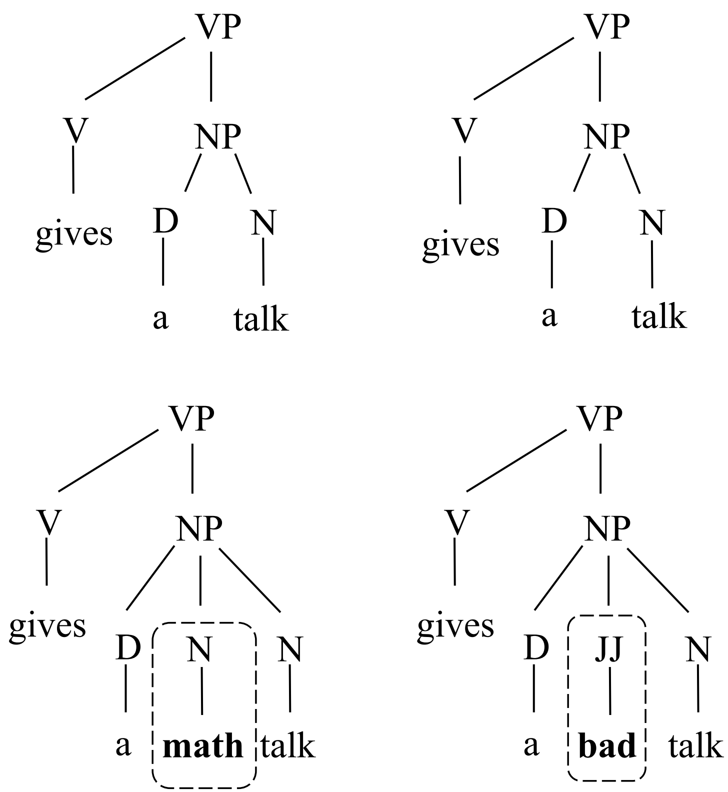


Labeled Ordered Tree Kernel

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



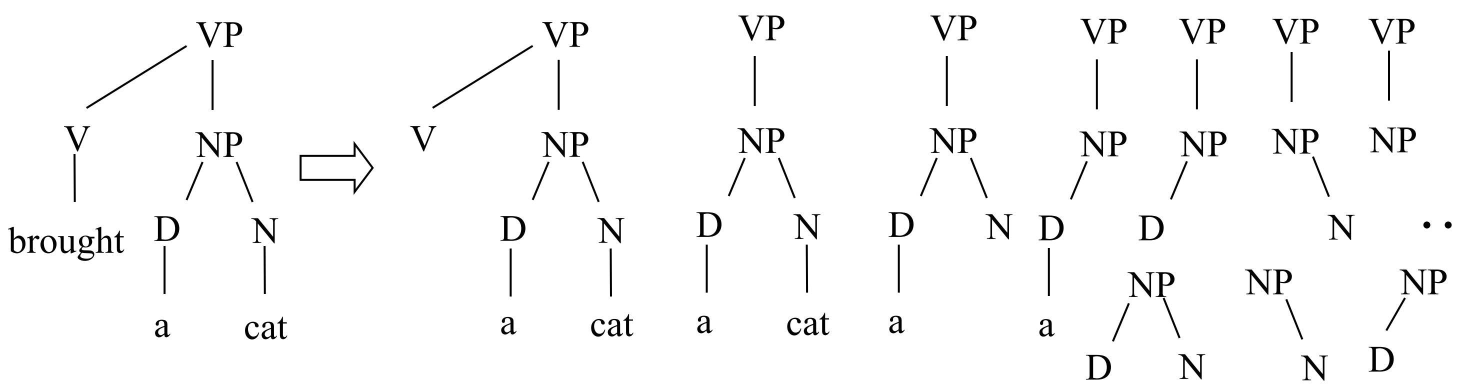
Weighting Problems



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

Partial Tree Kernel (PTK)
[Moschitti, ECML 2006]

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



Partial Tree Kernel - Definition

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

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

D_p

Efficient Evaluation (2)

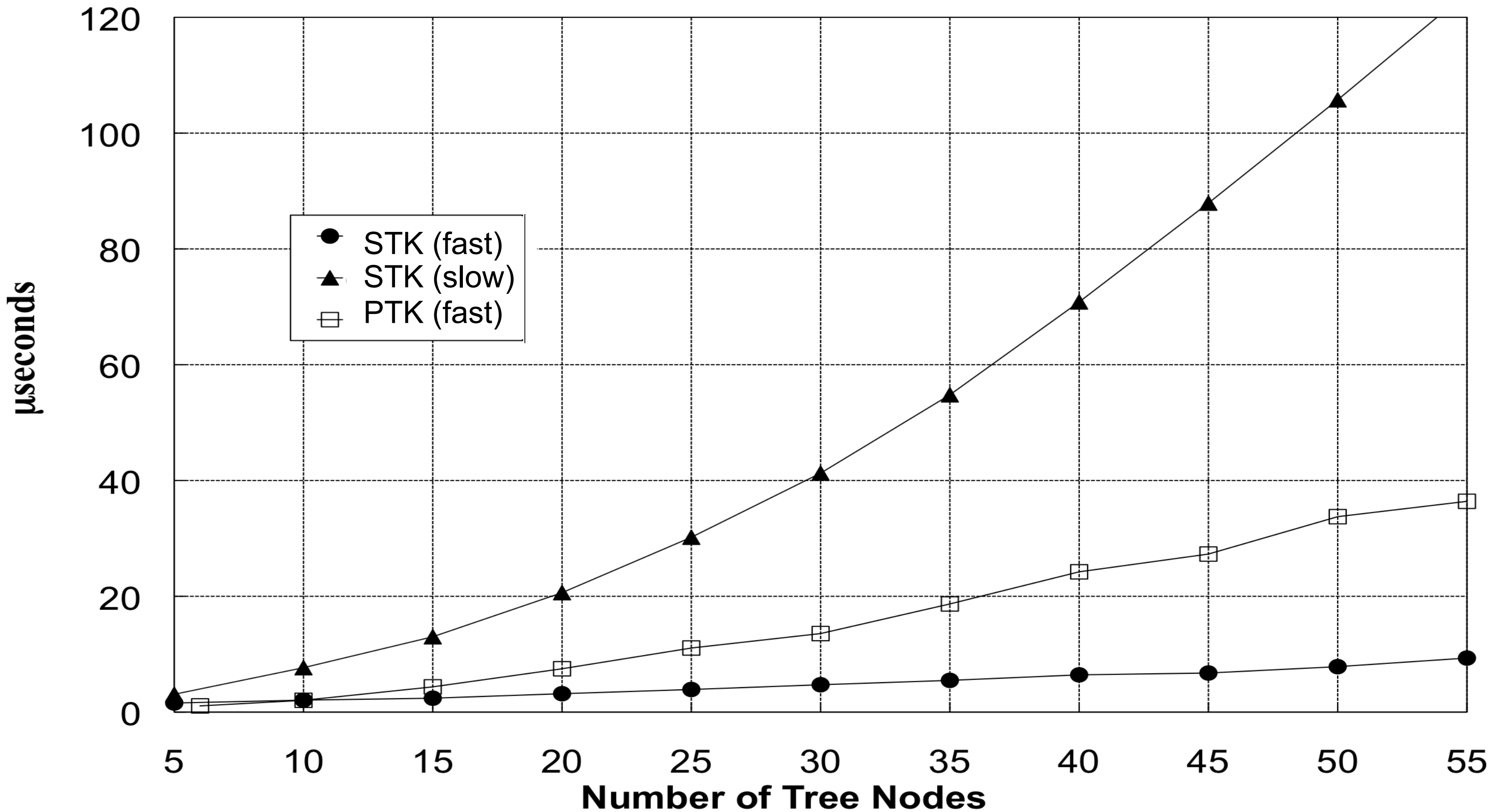
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

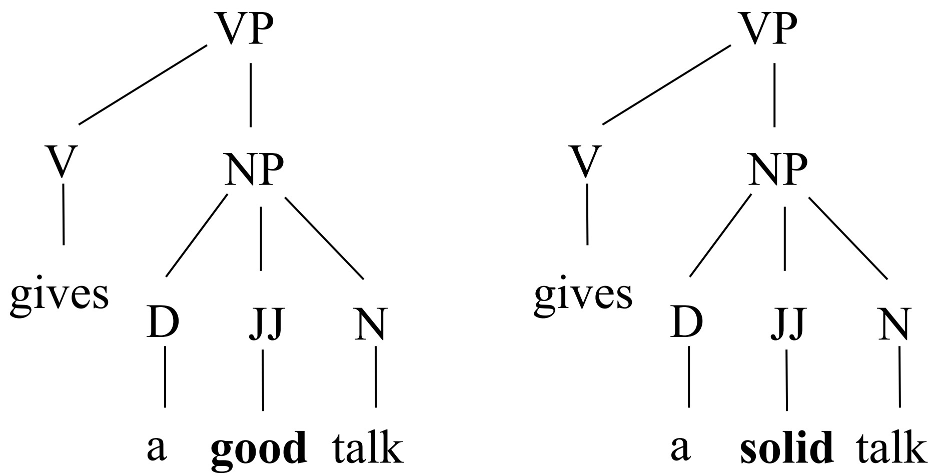
$$\begin{aligned} D_p(k,l) \quad &= \Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) \\ &+ \lambda D_p(k-1, l) + \lambda^2 D_p(k-1, l-1). \end{aligned}$$

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

Running Time of Tree Kernel Functions



Syntactic/Semantic Tree Kernels (SSTK)
[Bloehdorn & Moschitti, ECIR 2007 & CIKM 2007]



- Similarity between the fragment leaves
  - Tree kernel + Lexical Similarity Kernel

Equations of SSTK

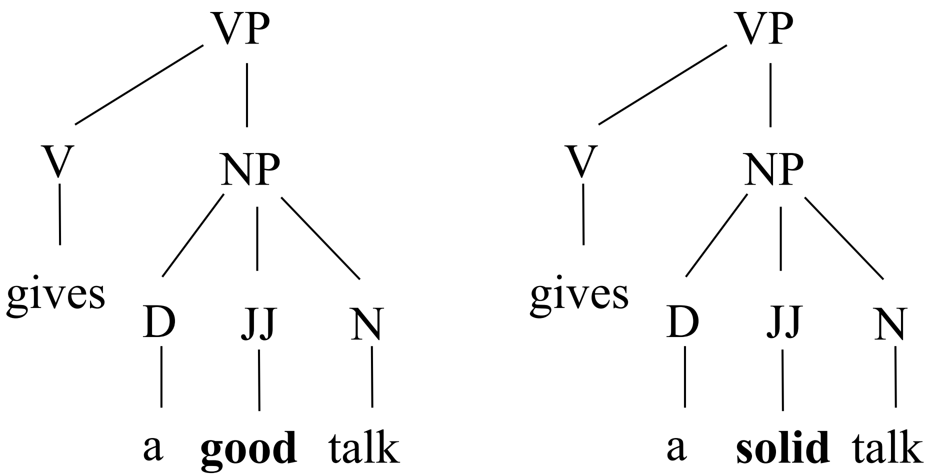
Definition 4 (Tree Fragment Similarity Kernel). *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[6]*:*

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

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

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

Example of an SSTK evaluation



K_S(gives,gives)*K_S(a,a)*
K_S(good,solid)*K_S(talk,talk)
= 1 * 1 * 0.5 * 1 = 0.5

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

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

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

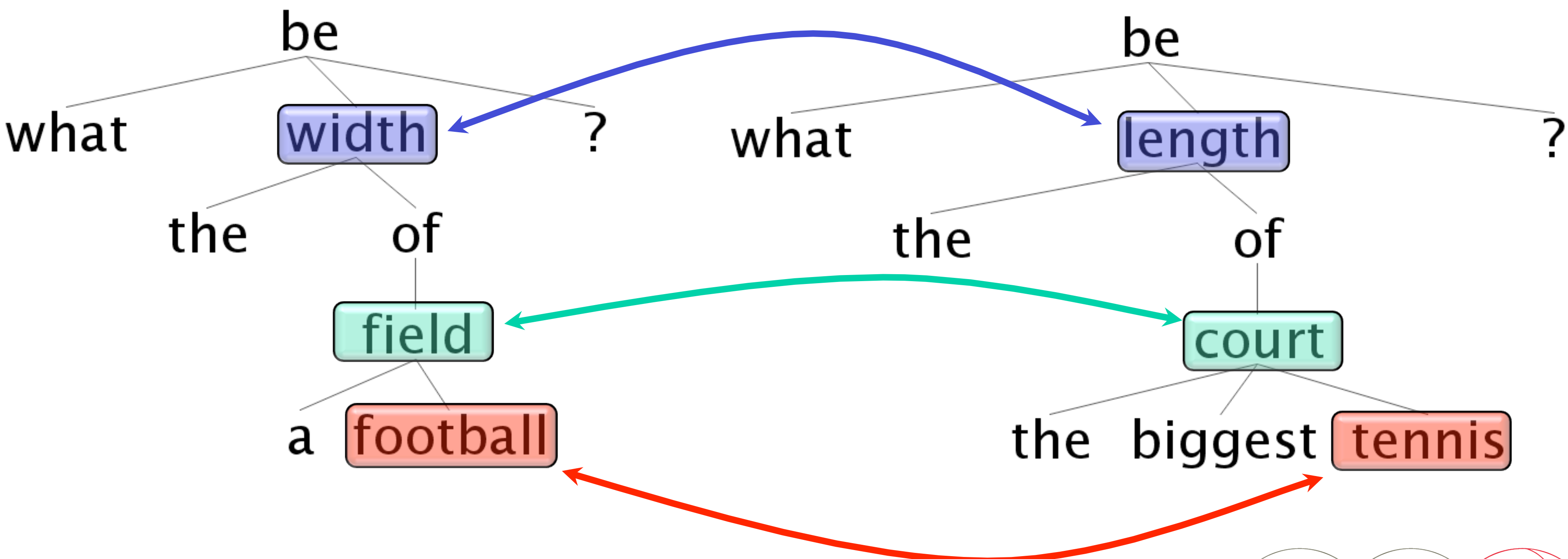
2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

Smoothed Partial Tree Kernels
[Moschitti, EACL 2009; Croce et al., 2011]

- Same idea of Syntactic Semantic Tree Kernel but the similarity is extended to any node of the tree
- The tree fragments are those generated by PTK
- Basically it extends PTK with similarities

Examples of Dependency Trees

- What is the width of a football field?
- What is the length of the biggest tennis court?



Equation of SPTK

If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \boxed{\mu\lambda\sigma(n_1, n_2)}$

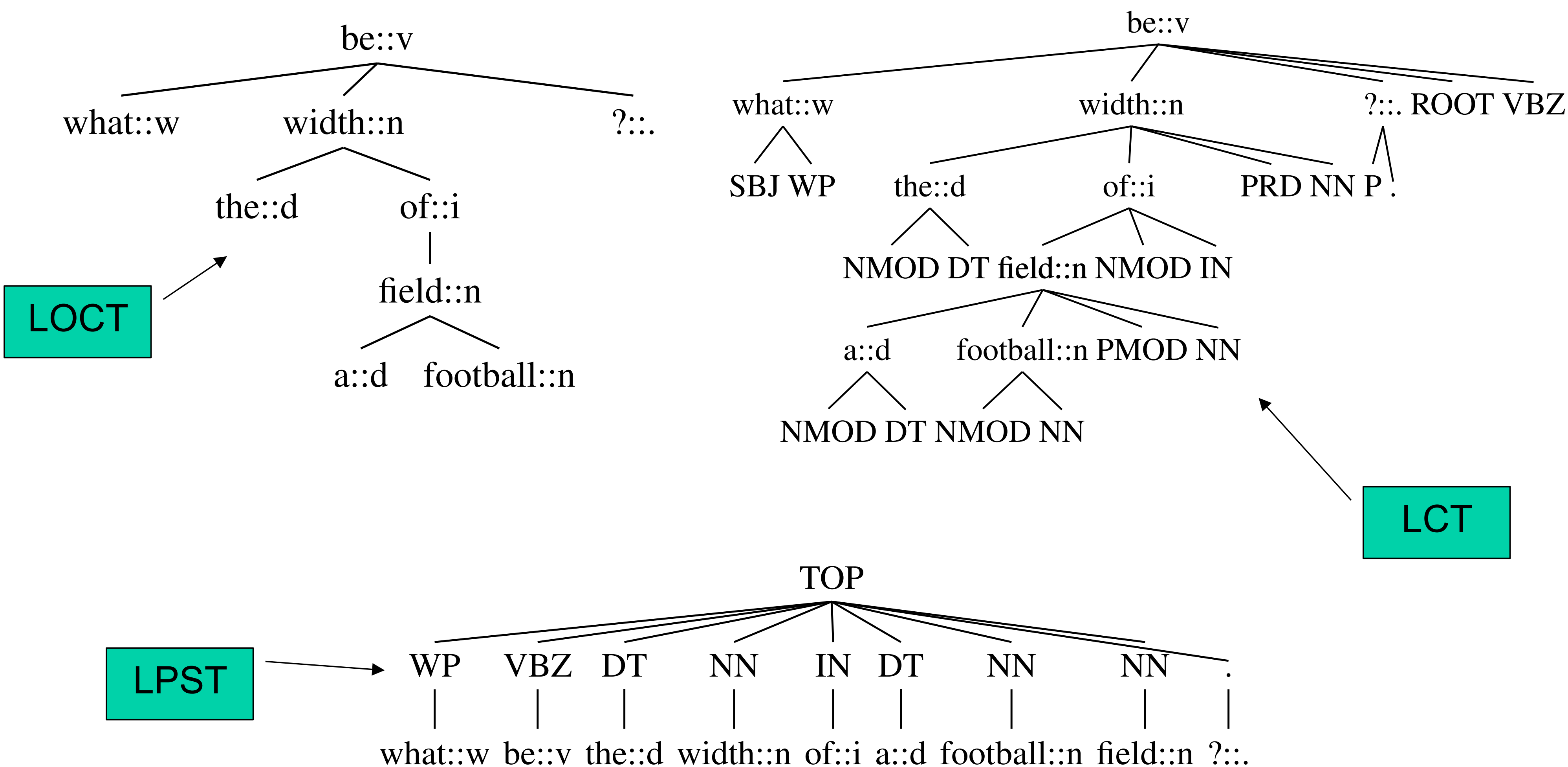
else

$$\Delta_\sigma(n_1, n_2) = \boxed{\mu\sigma(n_1, n_2)} \times \boxed{\left(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_\sigma(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\right)}$$

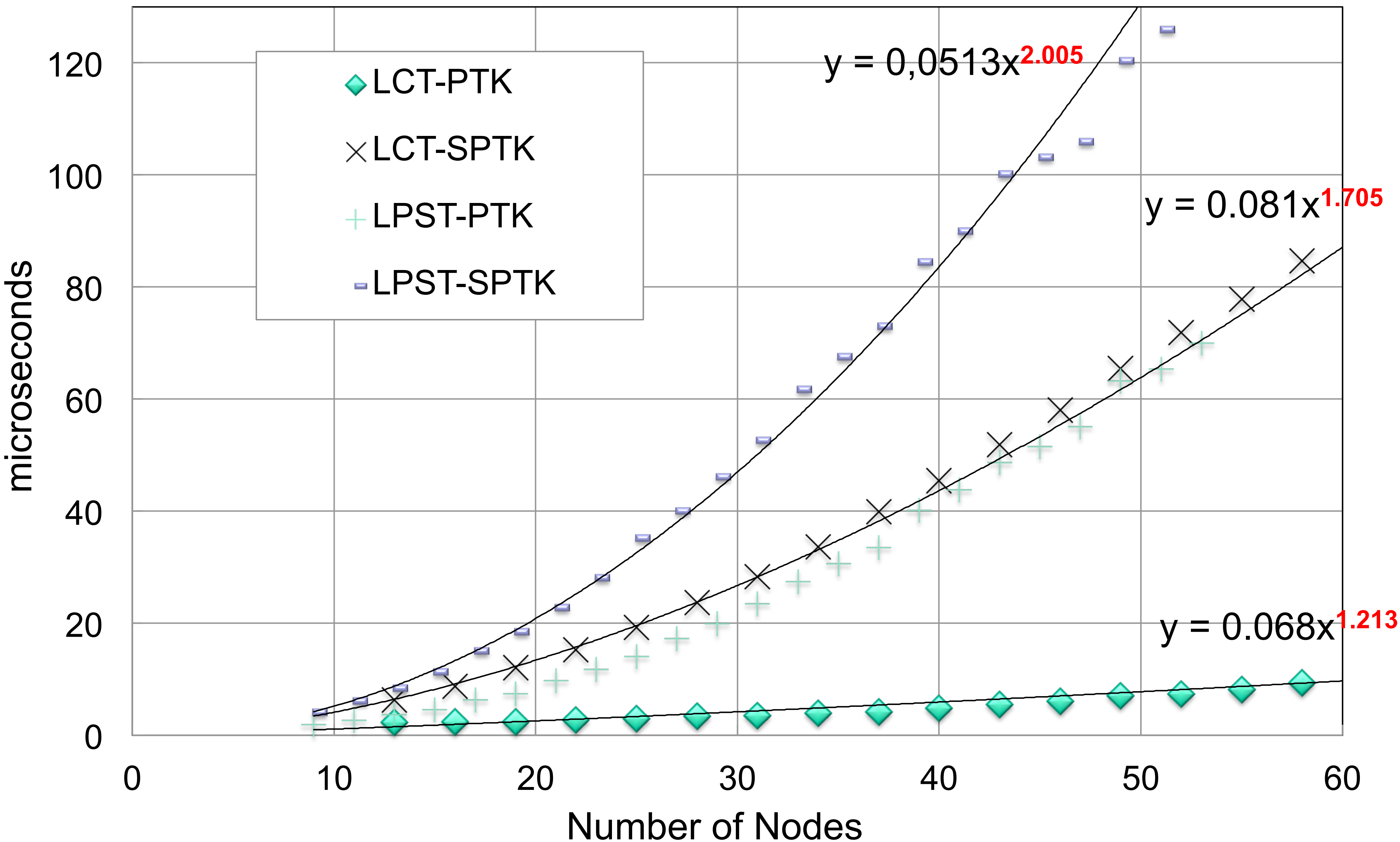
Lexical Similarity

PTK

Different versions of Computational Dependency Trees for PTK/SPTK



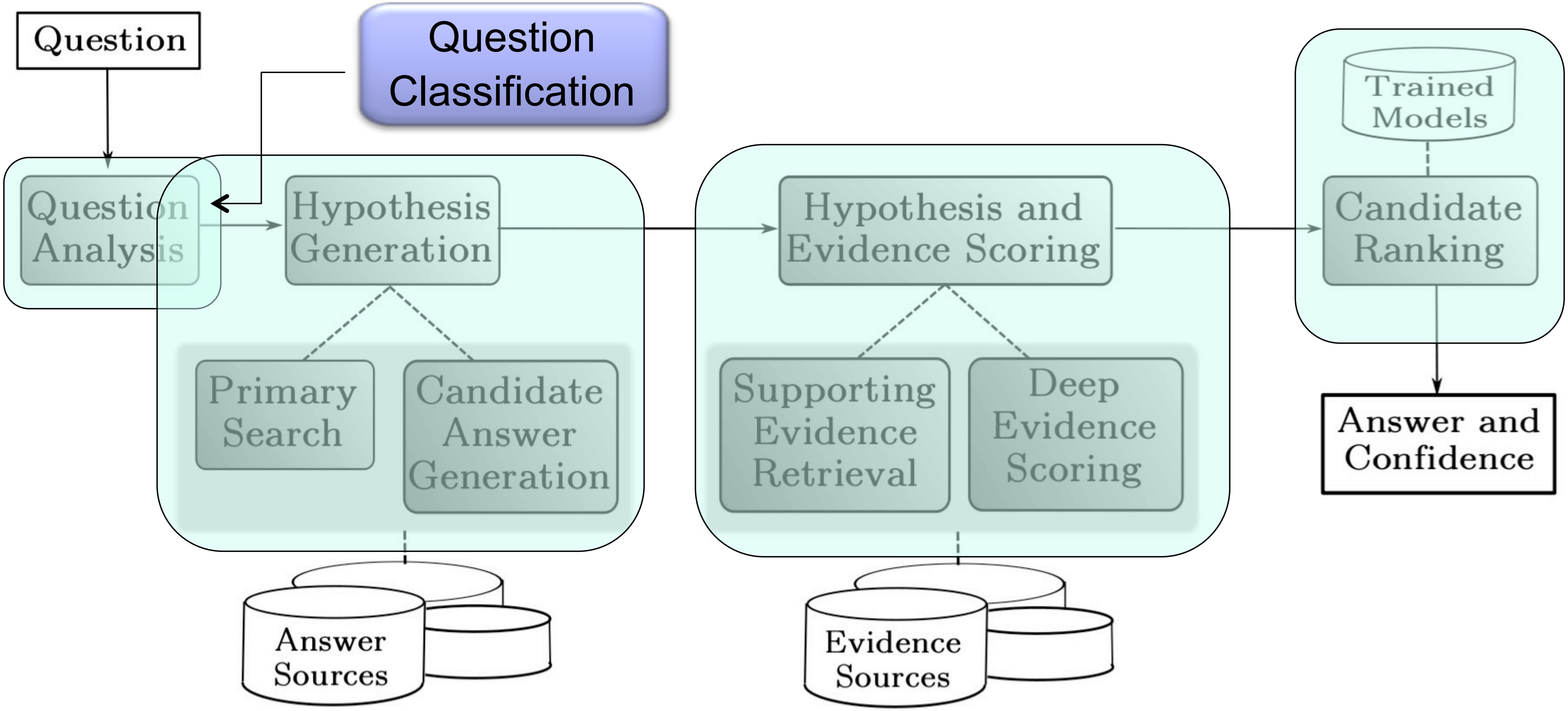
Tree Kernel Efficiency



Outline: Part I – Classification with Kernels

- Classification with Kernels (15 min)
  - Question Classification (QC) using constituency, dependency and semantic structures
  - Question Classification (QC) in Jeopardy!
  - Relation Extraction with kernels
  - Kernel-Based Coreference Resolution

IBM Watson (simplified) Pipeline



Question Classification

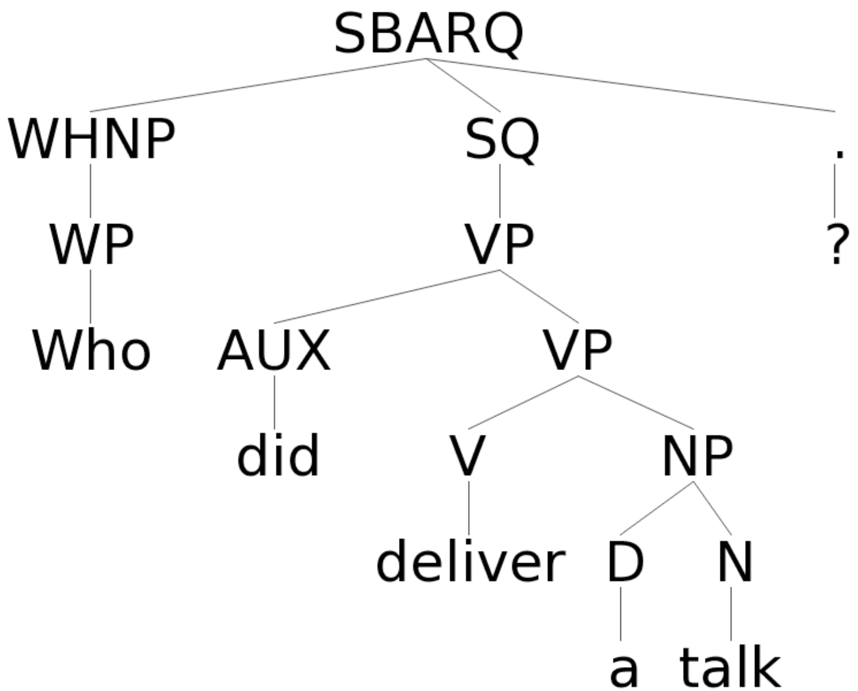
- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: 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

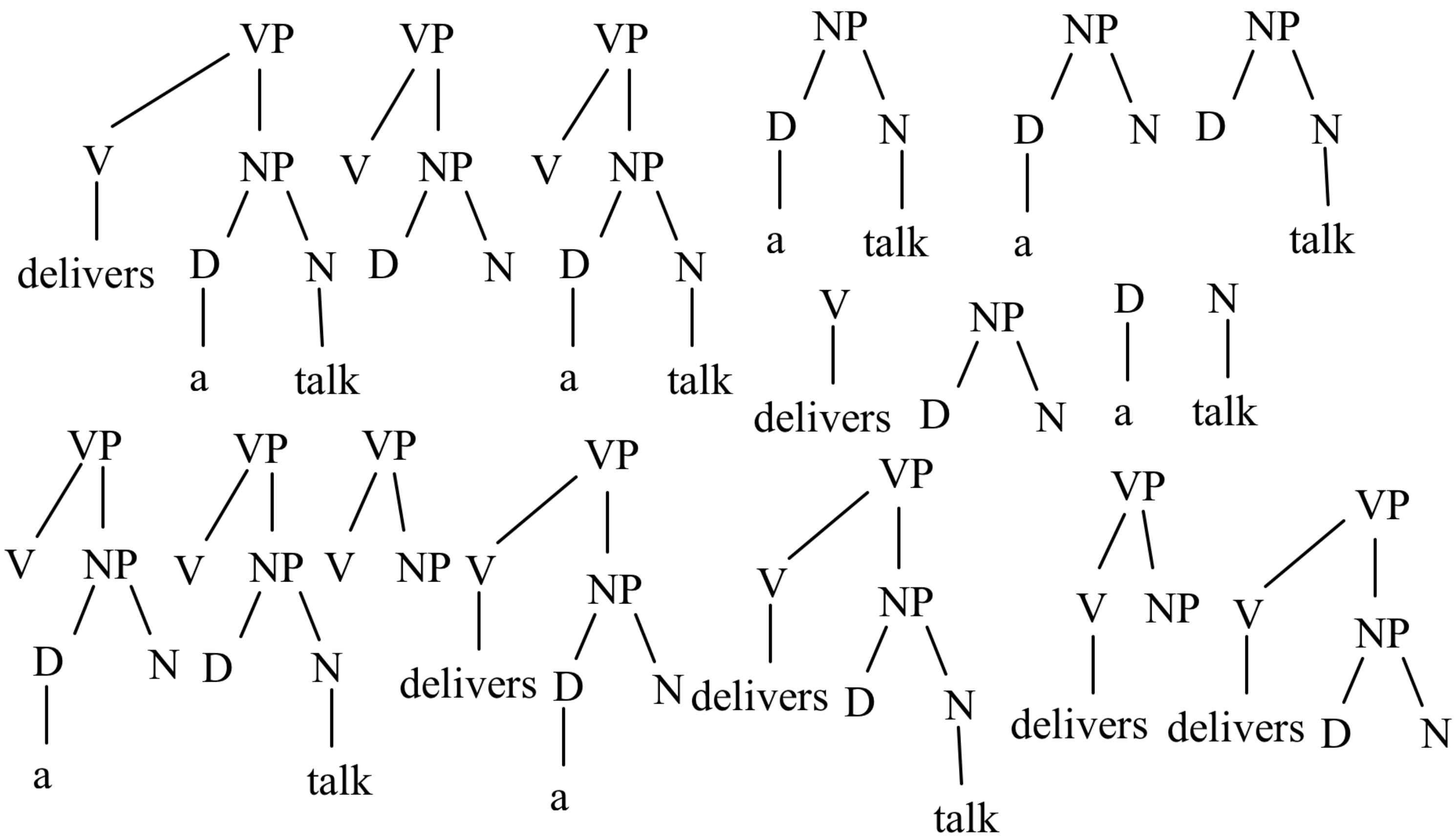
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Using the whole question parse trees
  - Constituent parsing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

“Who did deliver a talk?”

Syntactic Parse Trees (PT)

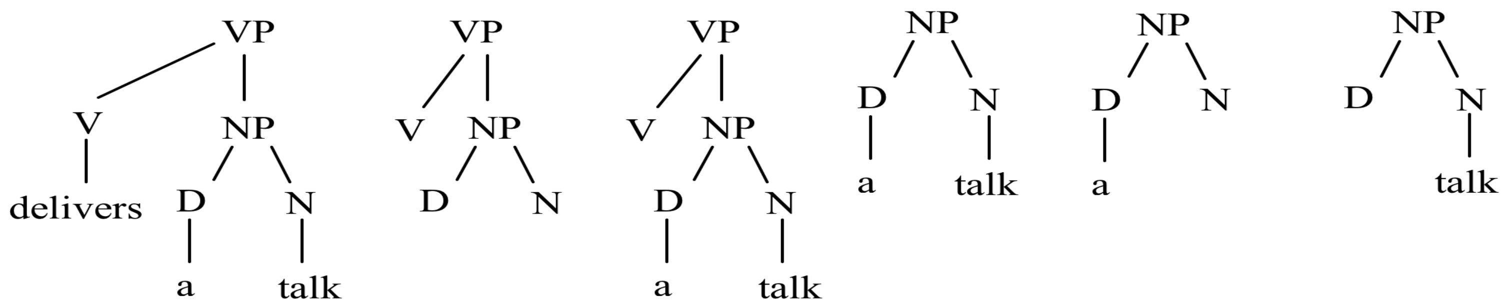


Some fragments from the VP subtree

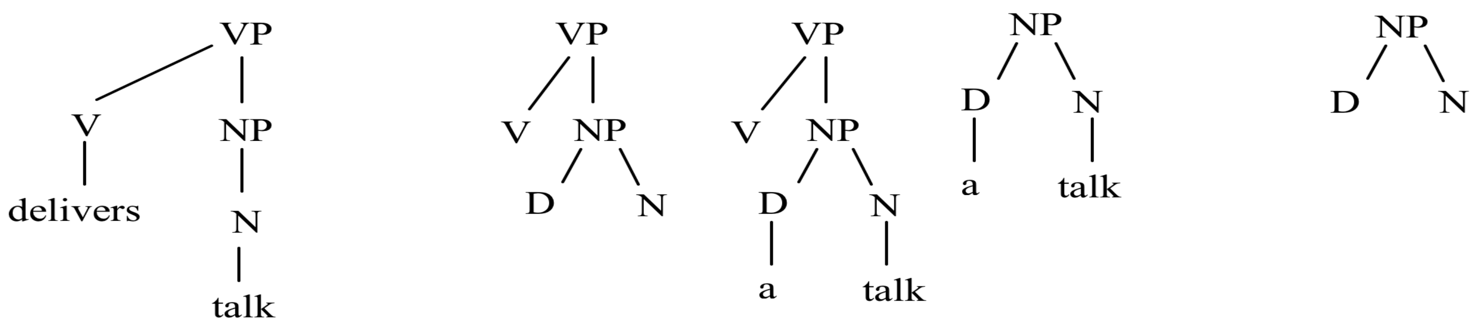


Explicit kernel space

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



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



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

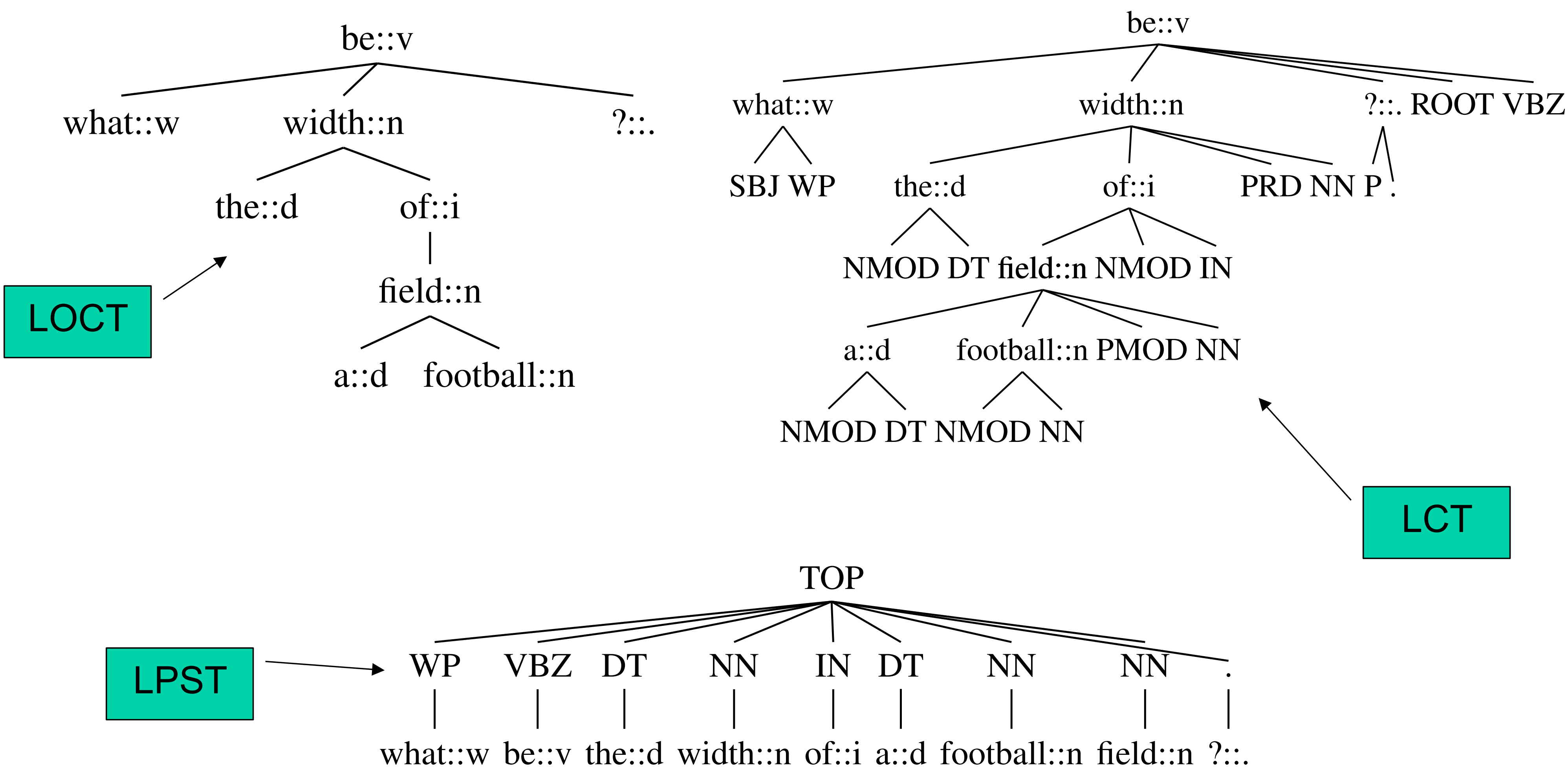
Question Classification with SSTK
[Blohedorn&Moschitti, CIKM2007]

Syntactic Tree Kernel
(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

Syntactic Tree Kernel
with similarities (SSTK)

Same Task with PTK, SPTK and Dependency Trees



State-of-the-art Results
[Croce et al., EMNLP 2011]

	STK	PTK	SPTK(LSA)
CT	91.20%	90.80%	91.00%
LOCT	-	89.20%	93.20%
LCT	-	90.80%	**94.80%**
LPST	-	89.40%	89.60%
BOW		88.80%	

Classification of Jeopardy! cues in definition vs. non definition


JEOPARDY!

Classification of Definition vs. non-Definition Questions in Jeopardy!

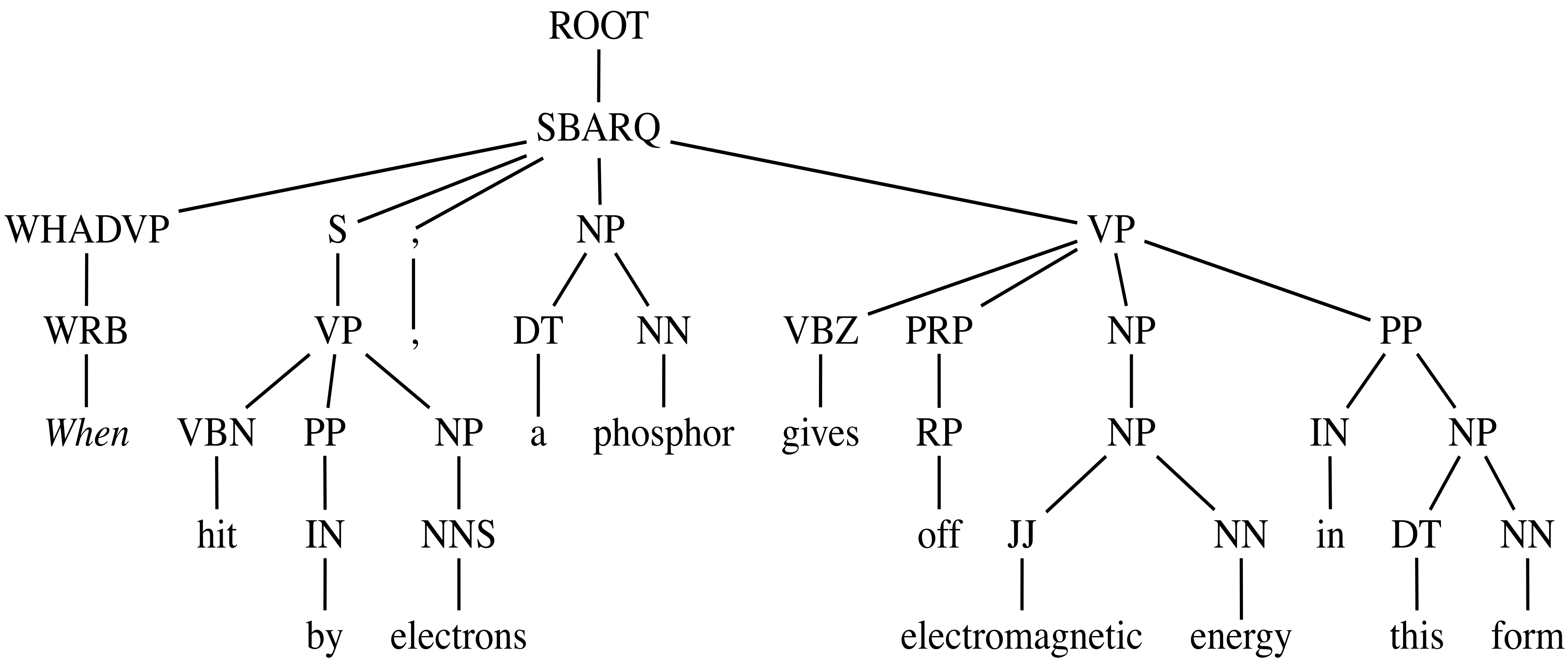
- **Definition:** *Usually, to do this is to lose a game without playing it*

  (solution: *forfeit*)
- **Non Definition:** *When hit by electrons, a phosphor gives off electromagnetic energy in this form*
- Complex linguistic problem: let us learn it from training examples using a syntactic similarity

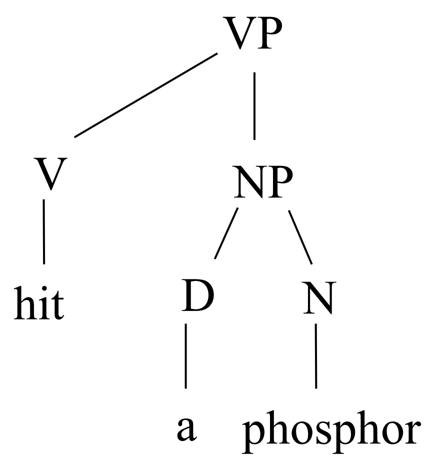
Automatic Learning of a Question Classifier

- Similarity between definition vs. non definition questions
- Instead of using features-based similarity we use kernels
- **Combining several linguistic structures** with several kernels for representing a question **q**:
  - $K_1(\langle q_1,q_2 \rangle)+K_2(\langle q_1,q_2 \rangle)+\ldots+K_n(\langle q_1,q_2 \rangle)$
- *n* tree kernels measure similarity between the *n* pairs of trees

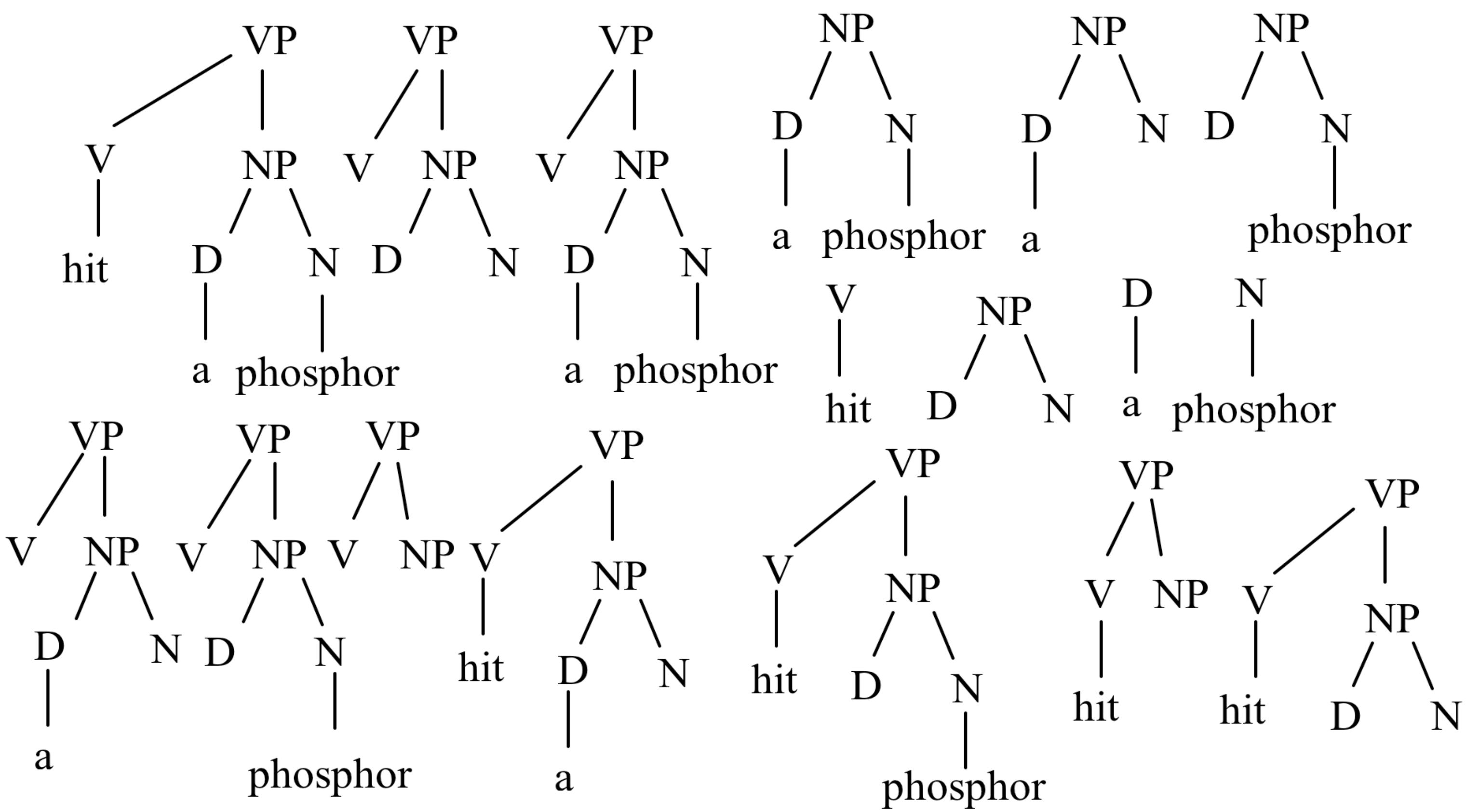
Constituency Tree (CT) – Apply STK



Syntactic Tree Kernel (STK)

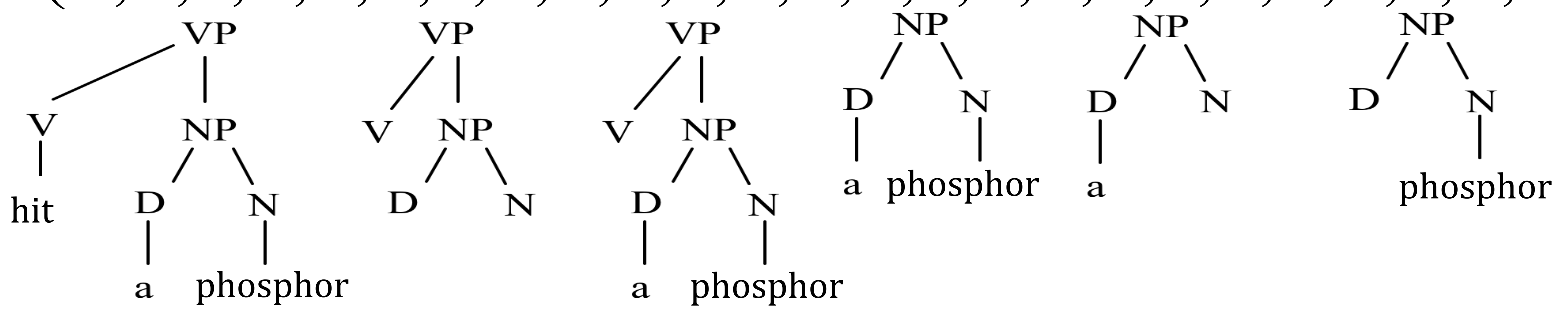


STK space

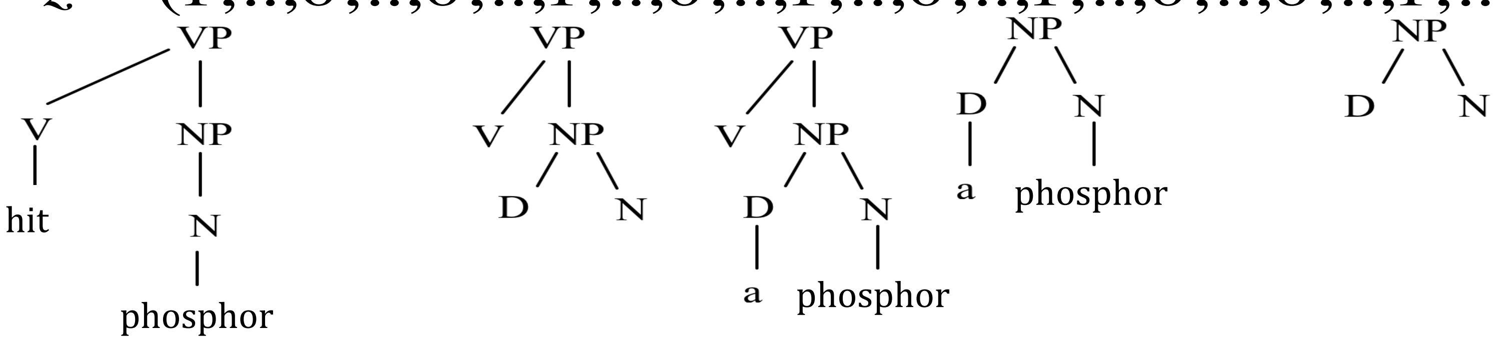


The explicit kernel space

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

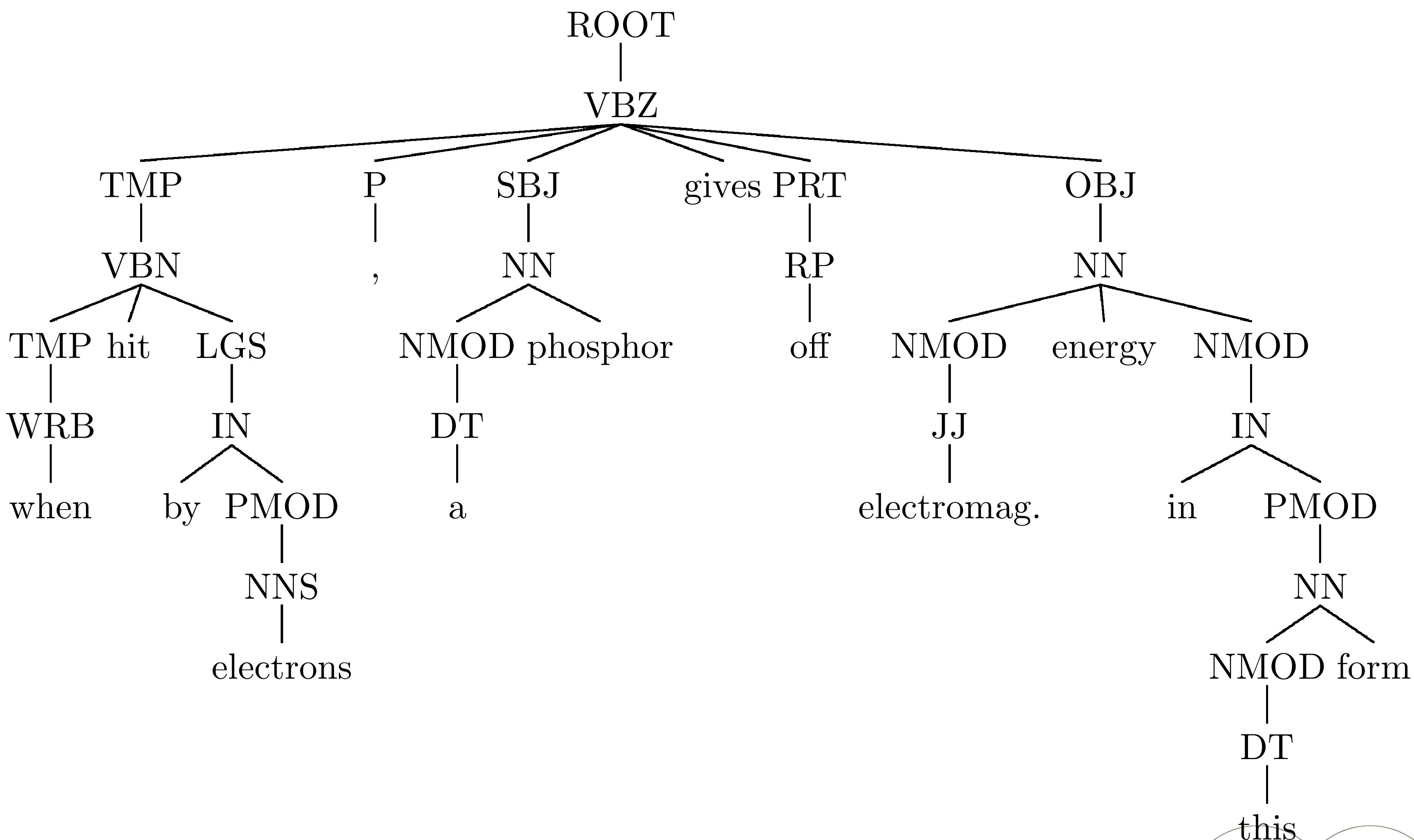


$$\phi(T_z) = \vec{z} = (1,..,0,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$

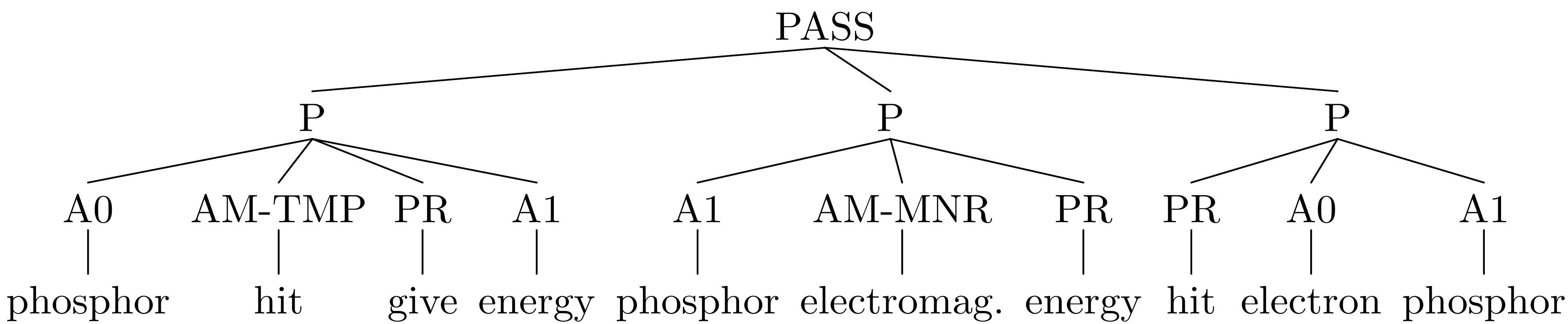


- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

Dependency Tree (DT) – Apply PTK



Predicate Argument Structure Set (PASS) – Apply PTK



Sequence Kernels on sequences of words and part-of-speech tags

WSK: [when][hit][by][electrons][,][a][phosphor][gives][off][electromagnetic][energy][in][this][form]

PSK: [wrb][vbn][in][nns][,][dt][nn][vbz][rp][jj][nn][in][dt][nn]

CSK: [general][science]
(category sequence kernel)

Experimental setup

- Corpus: a random sample from 33 Jeopardy! Games
- 306 definition and 4,964 non-definition clues
- Tools:
  - SVM-Light-TK
  - Charniak's constituency parser
  - Syntactic/Semantic parser by Johansson and Nugues (2008)
- Measures derived with leave-on-out

Individual models

Kernel Space	Prec.	Rec.	F1
RBC	28.27	70.59	**40.38**
BOW	47.67	46.73	47.20
WSK	47.11	50.65	48.82
STK-CT	50.51	32.35	39.44
PTK-CT	47.84	57.84	**52.37**
PTK-DT	44.81	57.84	50.50
PASS	33.50	21.90	26.49
PSK	39.88	45.10	42.33
CSK	39.07	77.12	**51.86**

Many Model Combinations

Kernel Space	Prec.	Rec.	F1
WSK+CSK	70.00	57.19	62.95
PTK-CT+CSK	69.43	60.13	64.45
PTK-CT+WSK+CSK	68.59	62.09	**65.18**
CSK+RBC	47.80	74.51	58.23
PTK-CT+CSK+RBC	59.33	74.84	65.79
BOW+CSK+RBC	60.65	73.53	66.47
PTK-CT+WSK+CSK+RBC	67.66	66.99	**67.32**
PTK-CT+PASS+CSK+RBC	62.46	71.24	66.56
WSK+CSK+RBC	69.26	66.99	**68.11**
ALL	61.42	67.65	64.38

Summary

Model	Precision	Recall	F1
RBC	28.27	70.59	**40.38**
BOW	46.55	50.65	48.51
CSK	39.07	77.12	51.86
PTK	47.84	57.84	52.37
PTK+CSK+RBC	67.66	66.99	**67.32**

- Rule Based Classifier (RBC)
- Only Word Overlap (BOW)
- Category Subsequences (CSK)
- Parse Tree (PTK)

Summary

Model	Precision	Recall	F1
RBC	28.27	70.59	**40.38**
PTK	47.84	57.84	52.37
PTK+CSK+RBC	67.66	66.99	**67.32**

66.7% relative improvement over RBC

- Rule Based Classifier (RBC)
- Only Word Overlap (BOW)
- Category Subsequences (CSK)
- Parse Tree (PTK)

Impact of QC in Watson

- Specific evaluation on definition questions
  - 1,000 unseen games (60,000 questions)
  - Two test sets of 1,606 and 1,875 questions derived with:
    - Statistical model (StatDef)
    - RBC (RuleDef)
  - Direct comparison only with NoDef

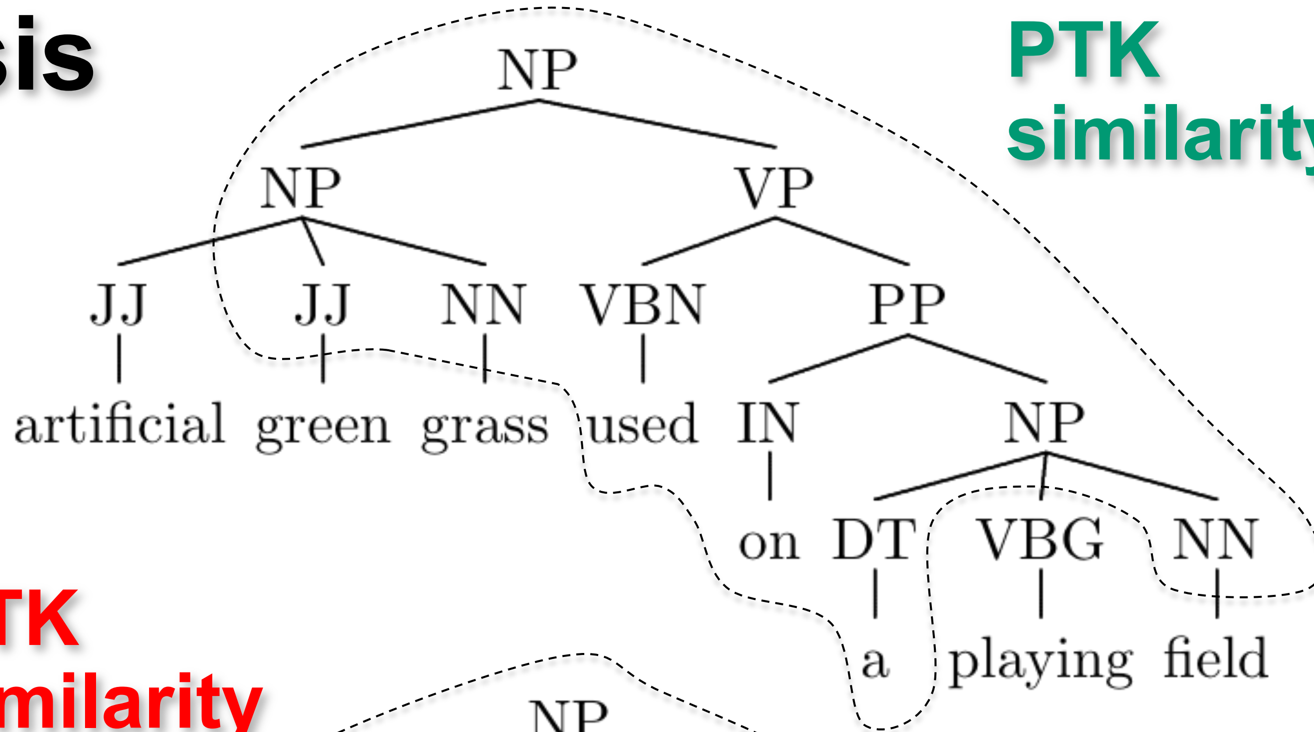
	# Def Q's	Accuracy	P@70	Earnings
NoDef	0	69.71%	86.79%	$24,818
RuleDef	480	69.23%	86.31%	$24,397
StatDef	131	69.85%	87.19%	$25,109

Error Analysis

Test Example

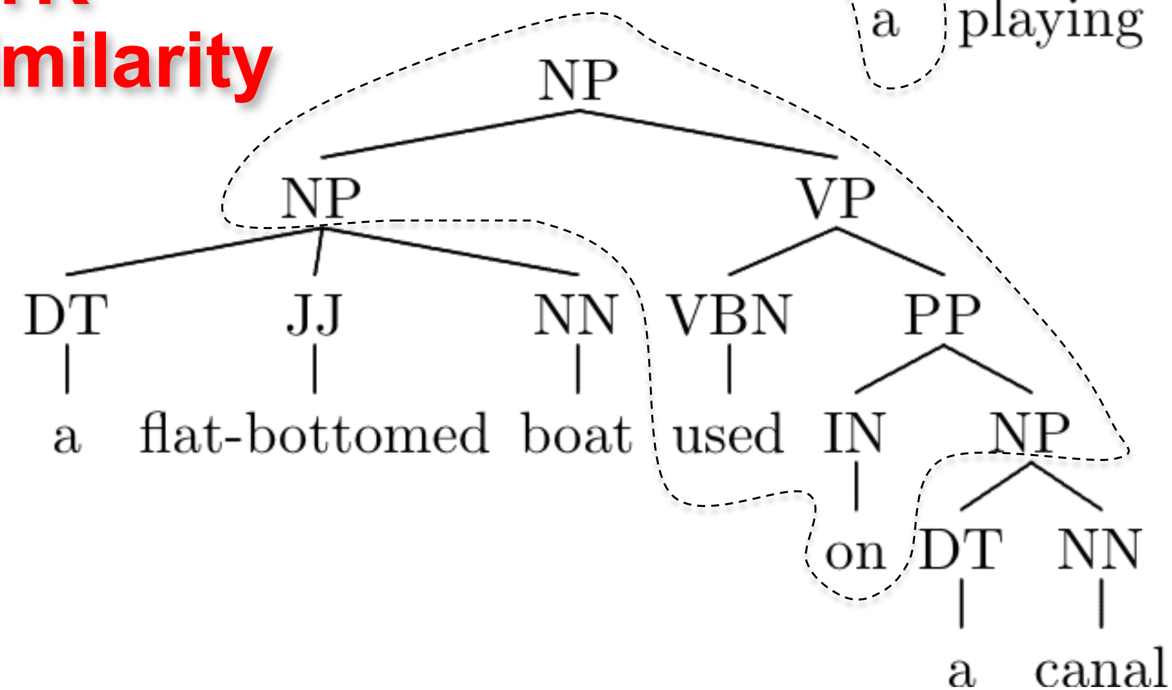
- PTK ok
- STK not ok

PTK similarity



STK similarity

Training Example



Outline: Part I – Classification with Kernels

- Classification with Kernels (30 min)
  - Question Classification (QC) using constituency, dependency and semantic structures
  - Question Classification (QC) in Jeopardy!
  - Relation Extraction with kernels
  - Kernel-Based Coreference Resolution

Relation Extraction

The Relation Extraction Problem

Last Wednesday, Eric Schmidt, the CEO of Google, defended the search engine's cooperation with Chinese censorship as he announced the creation of a research center in Beijing.

EMPLOYMENT

CEO ↔ Google

LOCATED

research center ↔ Beijing

Given a text with some available entities, how to recognize relations ?

Relation Extraction: The task

- Task definition: to label the semantic relation between pairs of entities in a sentence
  - The **governor** from **Connecticut**

M1
type: PER

M2
type: LOC

M := Entity Mention

  - Is there a relation between M1 and M2?
    If, so what kind of relation?

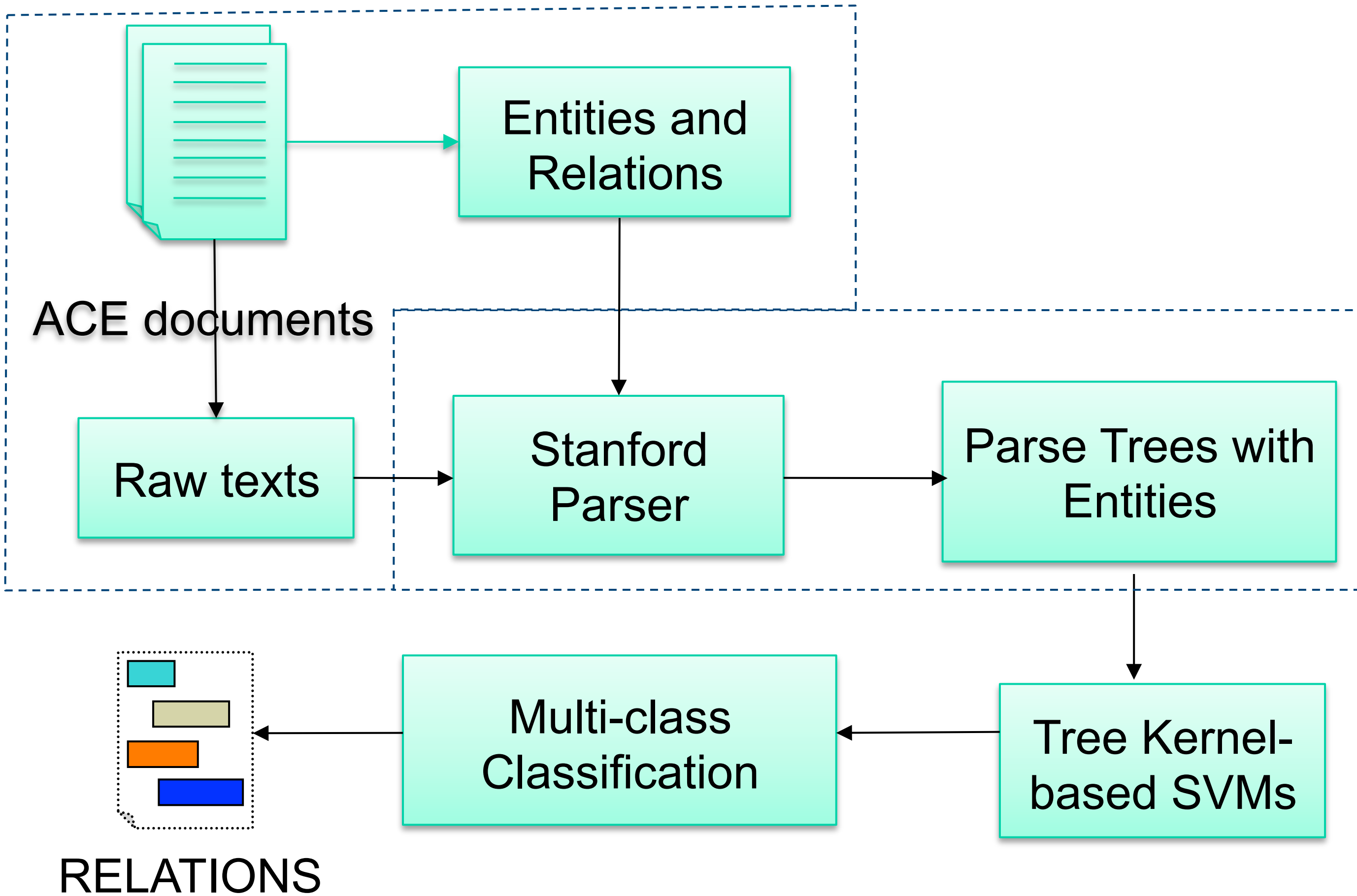
Relation Extraction defined in ACE

- Major relation types (from ACE 2004)

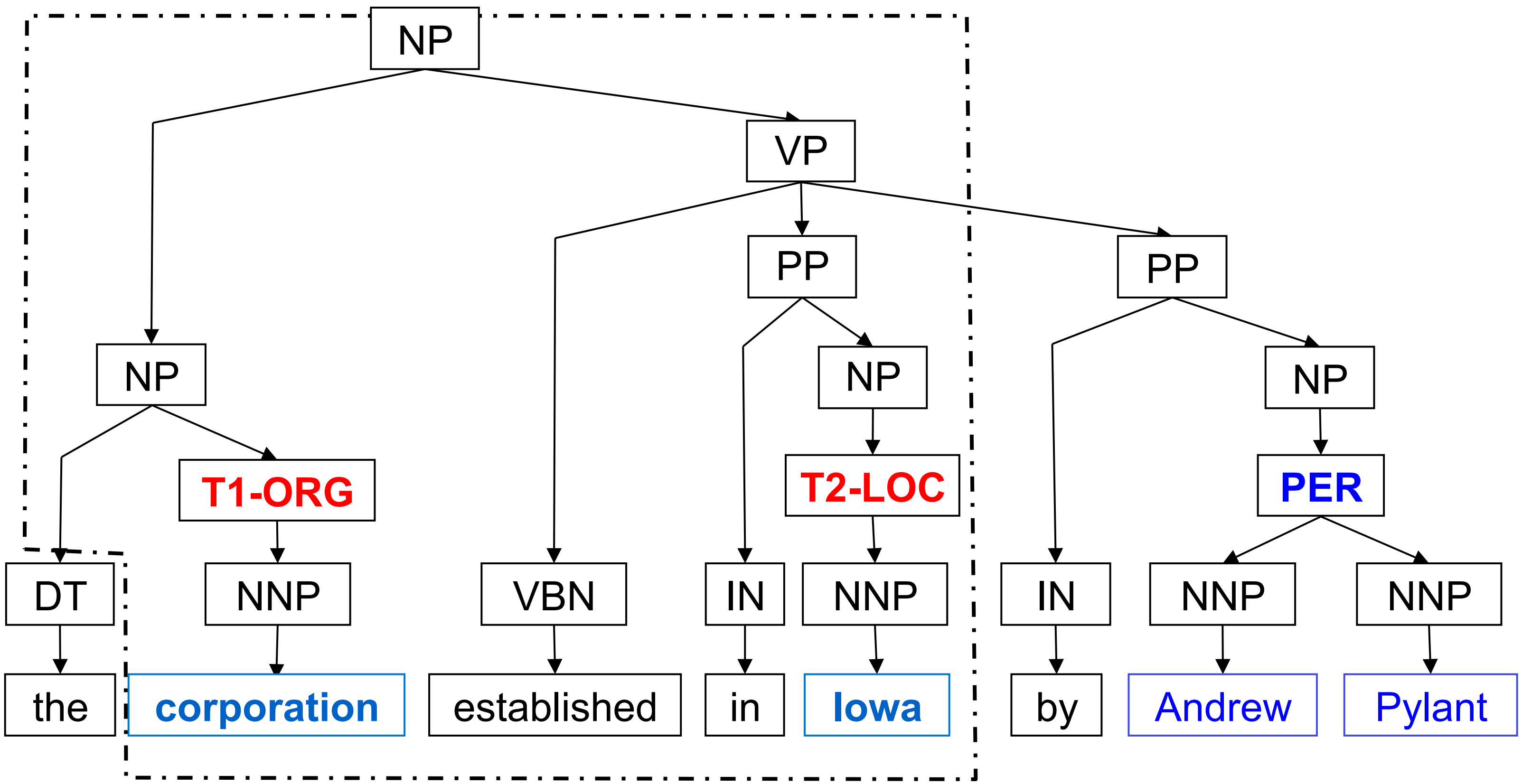
Type	Definition	Example
EMP-ORG	Employment	*US president*
PHYS	Located, near, part-whole	*a military base in Germany*
GPE-AFF	Affiliation	*U.S. businessman*
PER-SOC	Social	*a spokesman for the senator*
DISC	Discourse	*each of whom*
ART	User, owner, inventor …	*US helicopters*
OTHER-AFF	Ethnic, ideology …	*Cuban-American people*

- Entity types: PER, ORG, LOC, GPE, FAC, VEH, WEA

System Description [Nguyen et al, 2009]



Relation Representation
[Moschitti 2004; Zhang et al. 2006]



- The Path-enclosed tree captures the "PHYSICAL.LOCATED" relation between "**corporation**" and "**Iowa**"

Comparison

	Method	Data	P (%)	R (%)	F1 (%)
Zhang et al. (2006)	Composite Kernel (linear) with Context-Free Parse Tree	ACE 2004	73.5	67.0	70.1
Ours	Composite Kernel (linear) with Context-Free Parse Tree	ACE 2004	69.6	68.2	69.2

Both use the Path-Enclosed Tree for Relation Representation

Several Combination Kernels
[Nguyen et al, EMNLP 2009]

$$CK_1 = \alpha \cdot K_P + (1 - \alpha) \cdot K_x$$

$$CK_2 = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{SST} + K_{PTK})$$

$$CK_3 = \alpha \cdot K_{SST} + (1 - \alpha) \cdot (K_P + K_{PTK})$$

$$CK_4 = K_{PTK-DW} + K_{PTK-GR}$$

$$CK_5 = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{PTK-DW} + K_{PTK-GR})$$

$$SSK = \sum_{i=1,..,6} SK_i$$

$$CSK = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{SST} + SSK)$$

Results on ACE 2004

Kernel	P	R	F
$\mathbf{CK_1}$	**69.5**	**68.3**	**68.9**
SK_1	72.0	52.8	61.0
SK_2	61.7	60.0	60.8
SK_3	62.6	60.7	61.6
SK_4	73.1	50.3	59.7
SK_5	59.0	60.7	59.8
SK_6	57.7	61.8	59.7
$\mathbf{SK_3 + SK_4}$	**75.0**	**63.4**	**68.8**
$SK_3 + SK_6$	66.8	65.1	65.9
$\mathbf{SSK = \sum_i SK_i}$	**73.8**	**66.2**	**69.8**
$\mathbf{SST\ Kernel + SSK}$	**75.6**	**66.6**	**70.8**
$\mathbf{CK_1 + SSK}$	**76.6**	**67.0**	**71.5**
(Zhou et al., 2007) *CK_1 with Heuristics*	*82.2*	*70.2*	*75.8*

Coreference Resolution

Coreference Resolution

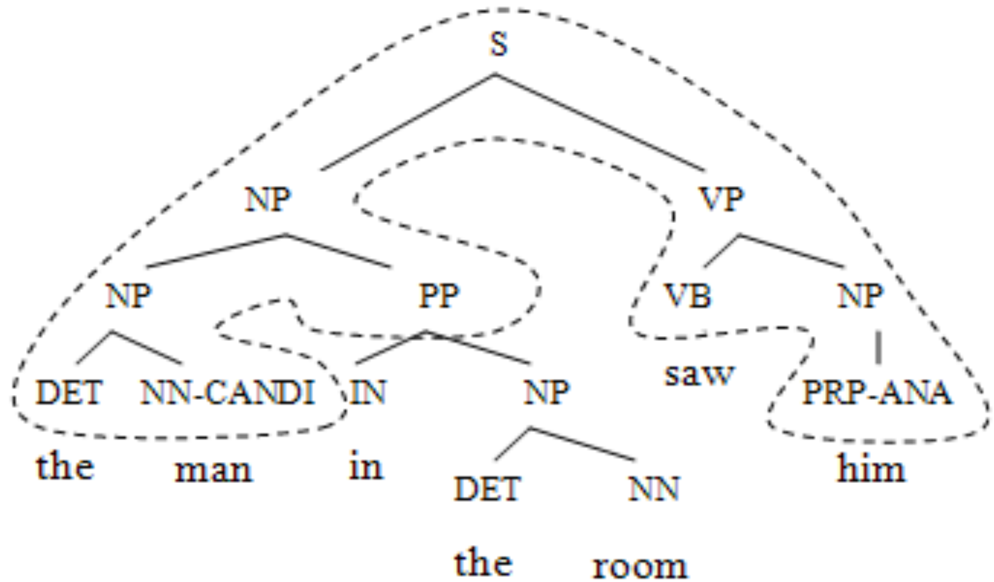
- Subtree that covers both anaphor and antecedent candidate

⇒ syntactic relations between anaphor & candidate (subject, object, c-commanding, predicate structure)

- Include the nodes in path between anaphor and candidate, as well as their first_level children

–“*the man* in the room saw *him*”

– inst(“the man”, “him”)



Context Sequence Feature

- A word sequence representing the mention expression and its context
  - Create a sequence for a mention

– “Even so, **Bill Gates** says that he just doesn’t understand our infatuation with thin client versions of Word ”

– (so)(,) (**Bill**)(**Gates**)(says)(that)

Composite Kernel

- Different kernels for different features
  - Poly Kernel for baseline flat features
  - Tree Kernel for syntax trees
  - Sequence Kernel for word sequences
- A composite kernel for all kinds of features
- Composite Kernel = TK*PolyK+PolyK+SK

Results for pronoun resolution
[Vesley et al, Coling 2008]

	MUC-6			ACE-02-BNews		
	R	P	F	R	P	F
All attribute value features	64.3	63.1	63.7	58.9	68.1	63.1
+ Syntactic Tree + Word Sequence	65.2	80.1	**71.9**	65.6	69.7	**67.6**

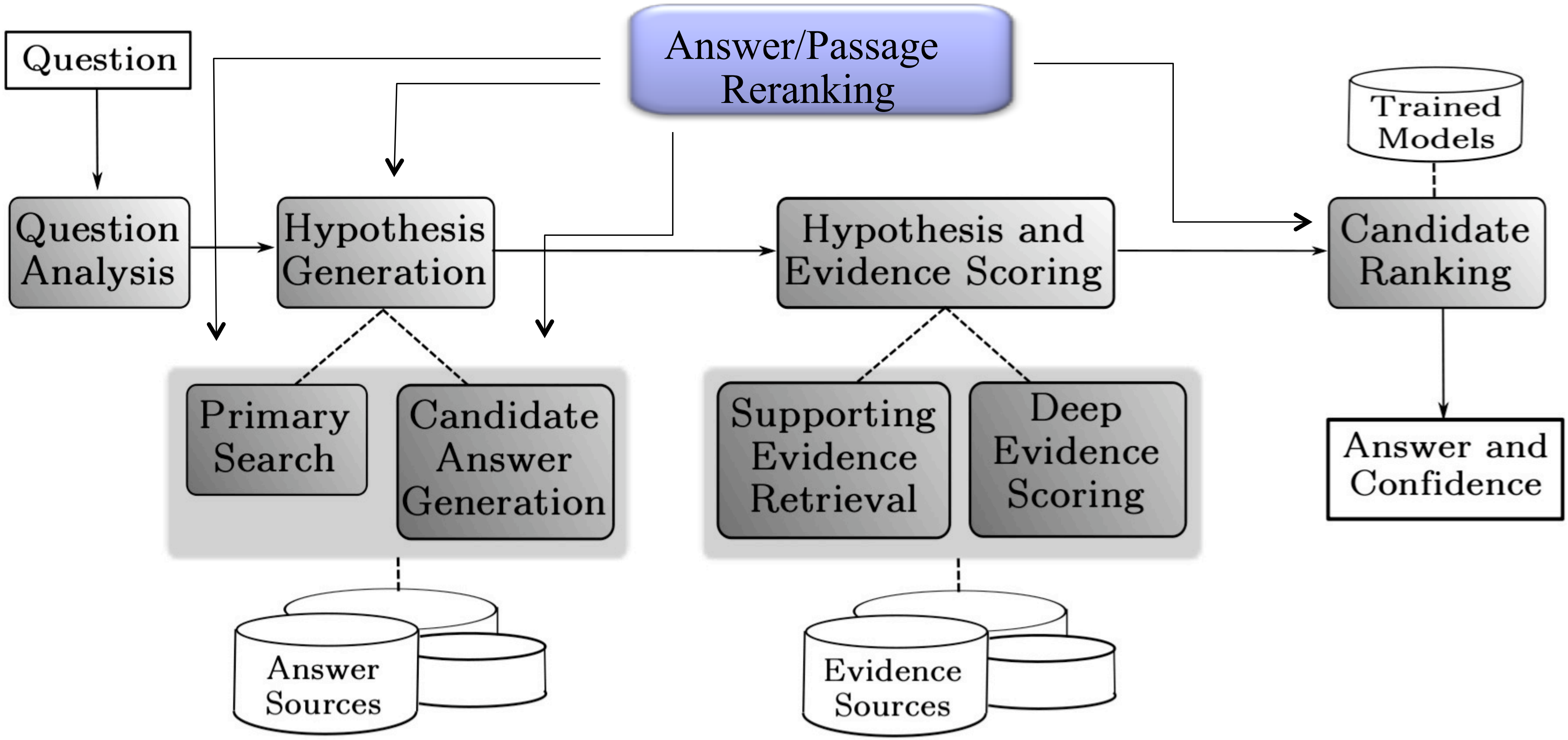
Results on the overall Coreference Resolution task using SVMs

	MUC-6			ACE02-BNews		
	R	P	F	R	P	F
Basic Features SVMs	61.5	67.2	64.2	54.8	66.1	59.9
Basic Features + Syntax Tree	63.4	67.5	65.4	56.6	66.0	60.9
Basic Features + Syntax Tree + Word Sequences	64.4	67.8	66.0	57.1	65.4	61.0
All Sources of Knowledge	60.1	76.2	67.2	60.0	65.4	63.0

Outline: Part II – Kernels for Ranking

- Reranking with kernels (40 min)
  - Classification of Question/Answer (QA) pairs
  - Preference Reranking Kernel
  - Reranking NLP tasks
    - Named Entities in a sentence
    - Predicate Argument Structures
    - Segmentation and labeling of Speech Transcriptions
  - Reranking the output of a hierarchical text classifier
  - Reranking Passages with relational representations: the IBM Watson system case

Answer/Passage Reranking



Reranking with a QA classifier

Reranking framework

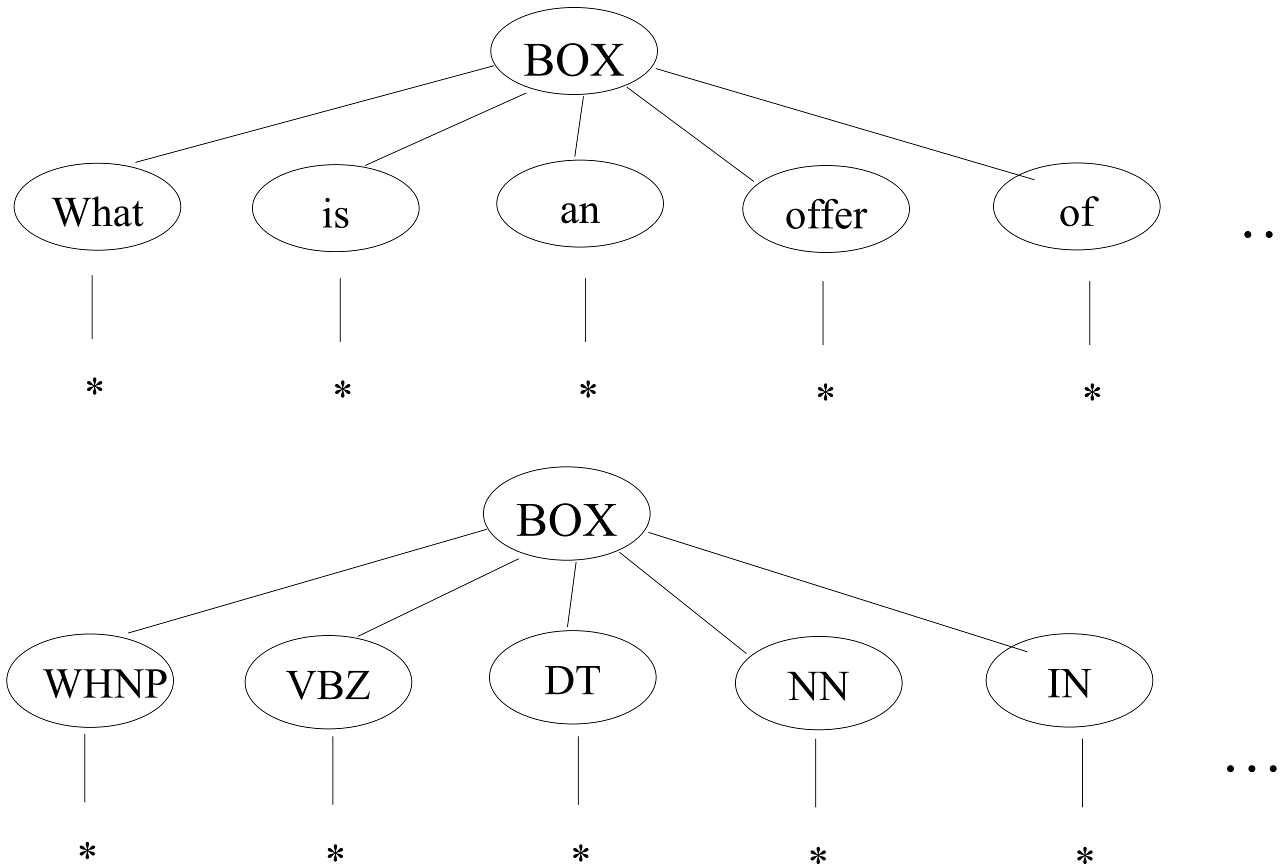
$H_1 = (q_1,p_1)$		-1		$H_2 = (q_2,p_2)$
$H_2 = (q_2,p_2)$	**Pair Classifier** →	+1	**Reranking with scores** →	$H_n = (q_n,p_n)$
.		.		.
.		.		.
.		.		.
$H_n = (q_n,p_n)$		+1		$H_1 = (q_1,p_1)$

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

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

- To save time, apply tree kernels to these trees:



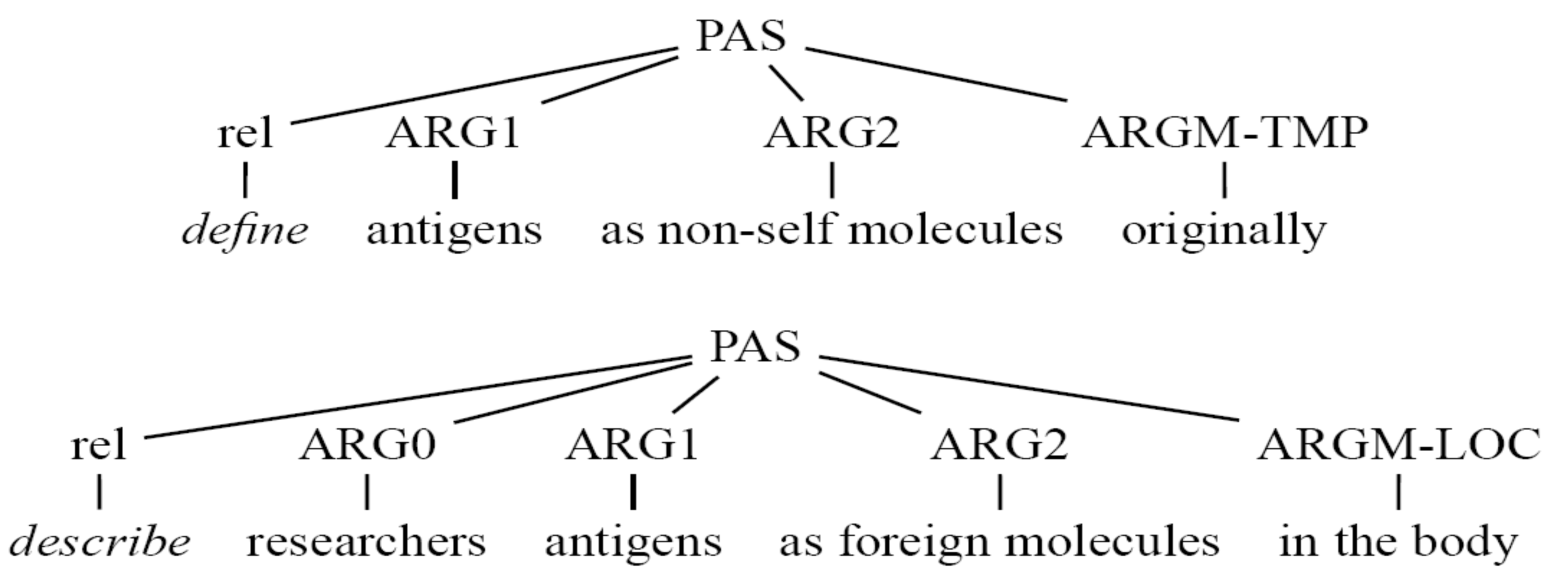
Word and POS Sequences

- 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`

Predicate Argument Structures for describing answers (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]



Dataset 2: TREC data

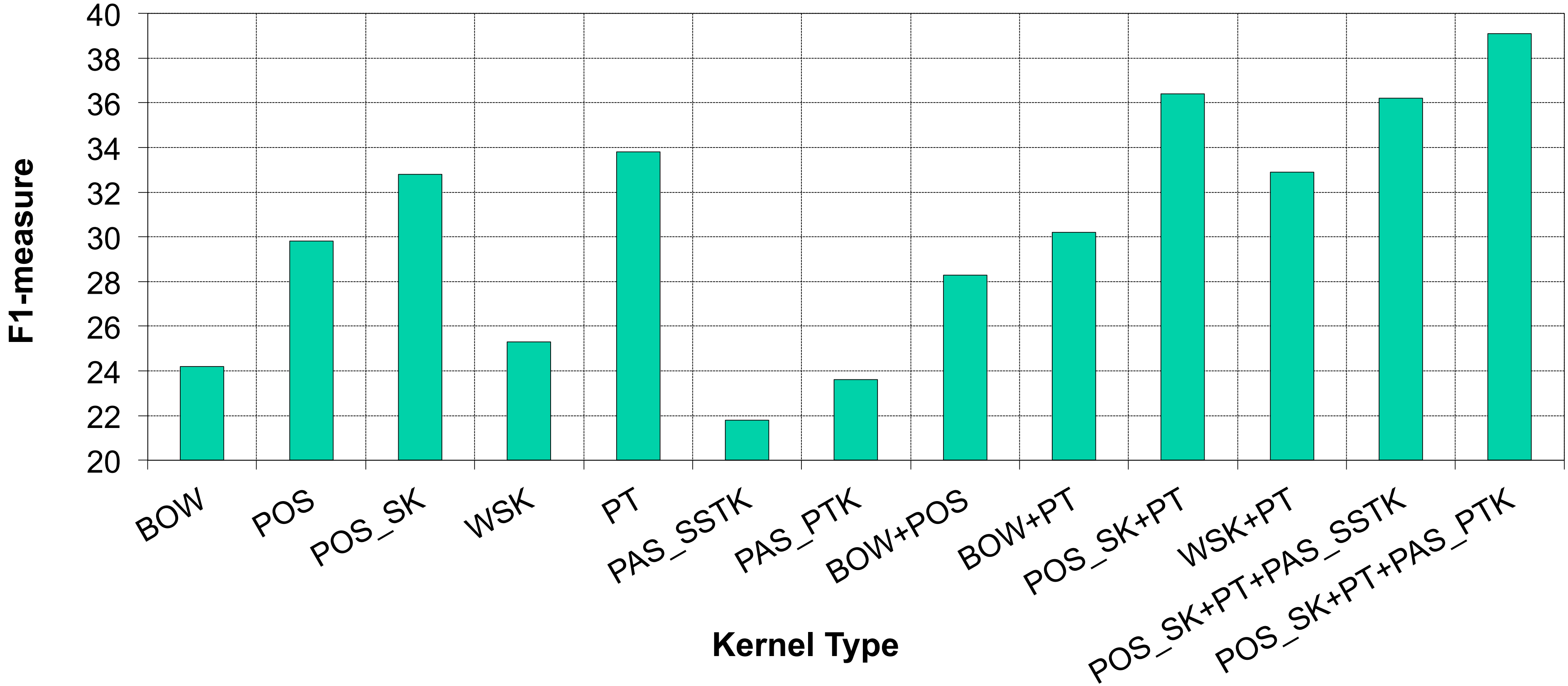
- 138 TREC 2001 test questions labeled as “description”
- 2,256 sentences, extracted from the best ranked paragraphs (using a basic QA system based on Lucene search engine on TREC dataset)
- 216 of which labeled as correct by one annotator

Kernels and Combinations

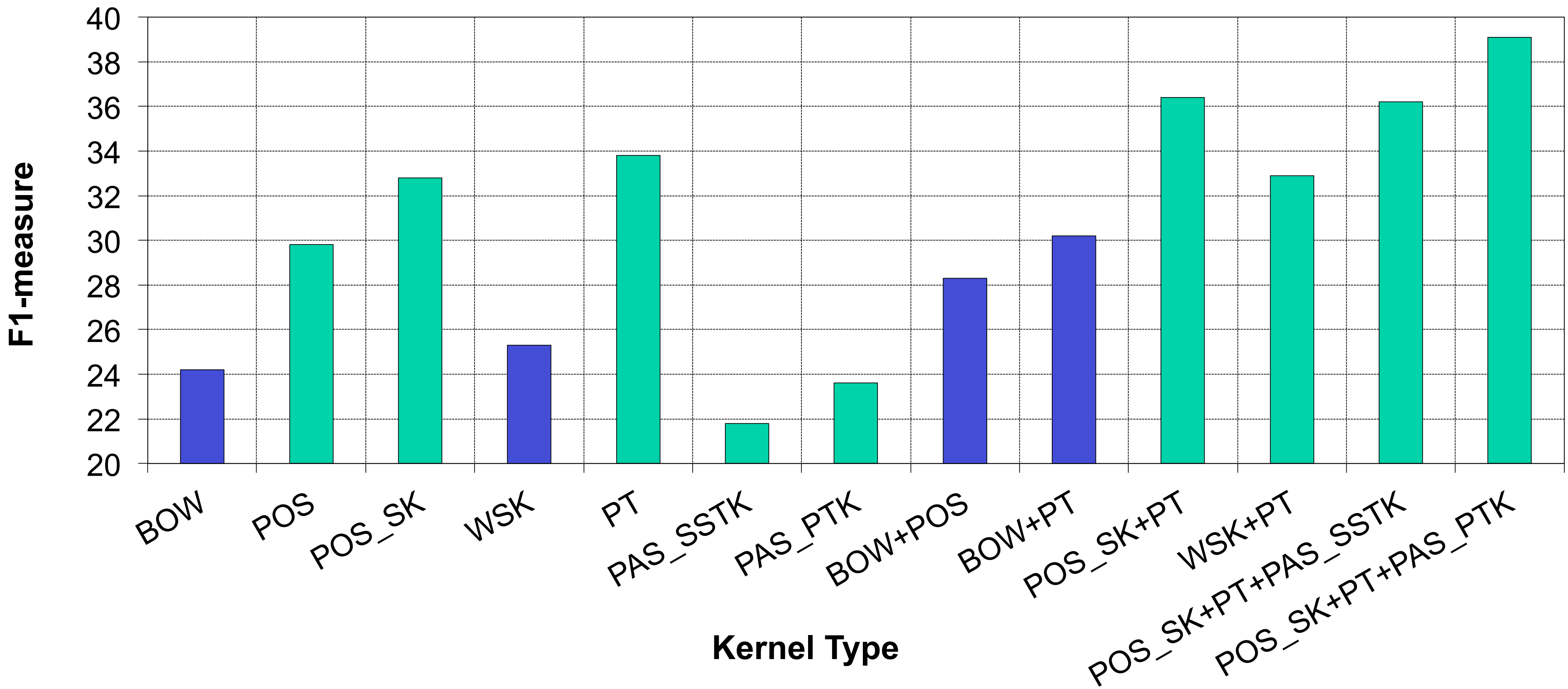
- Exploiting the property: $k(x,z) = k_1(x,z)+k_2(x,z)$
- Given: BOW, POS, WSK, POSSK, PT, PAS_{PTK}

$\Rightarrow$ BOW+POS, BOW+PT, PT+POS, …

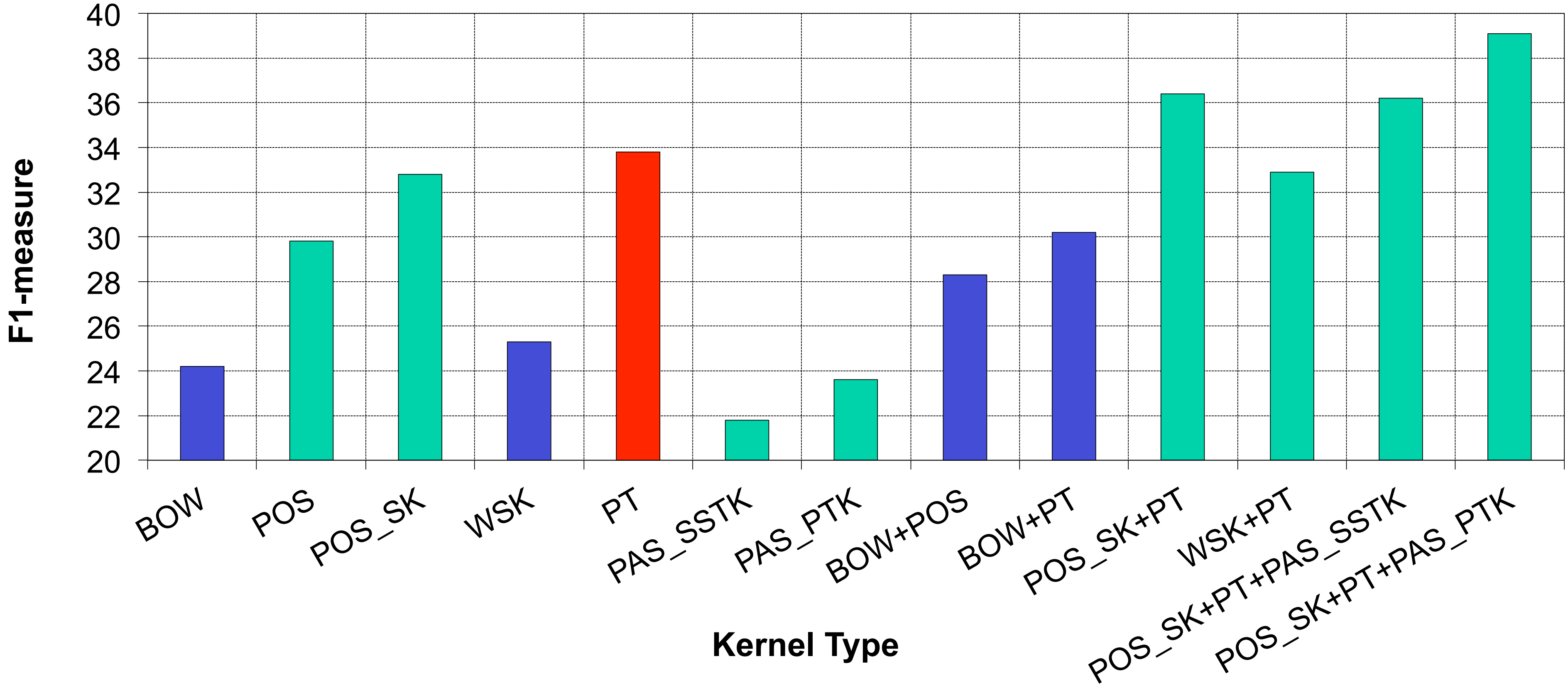
Results on TREC Data
(5 folds cross validation)



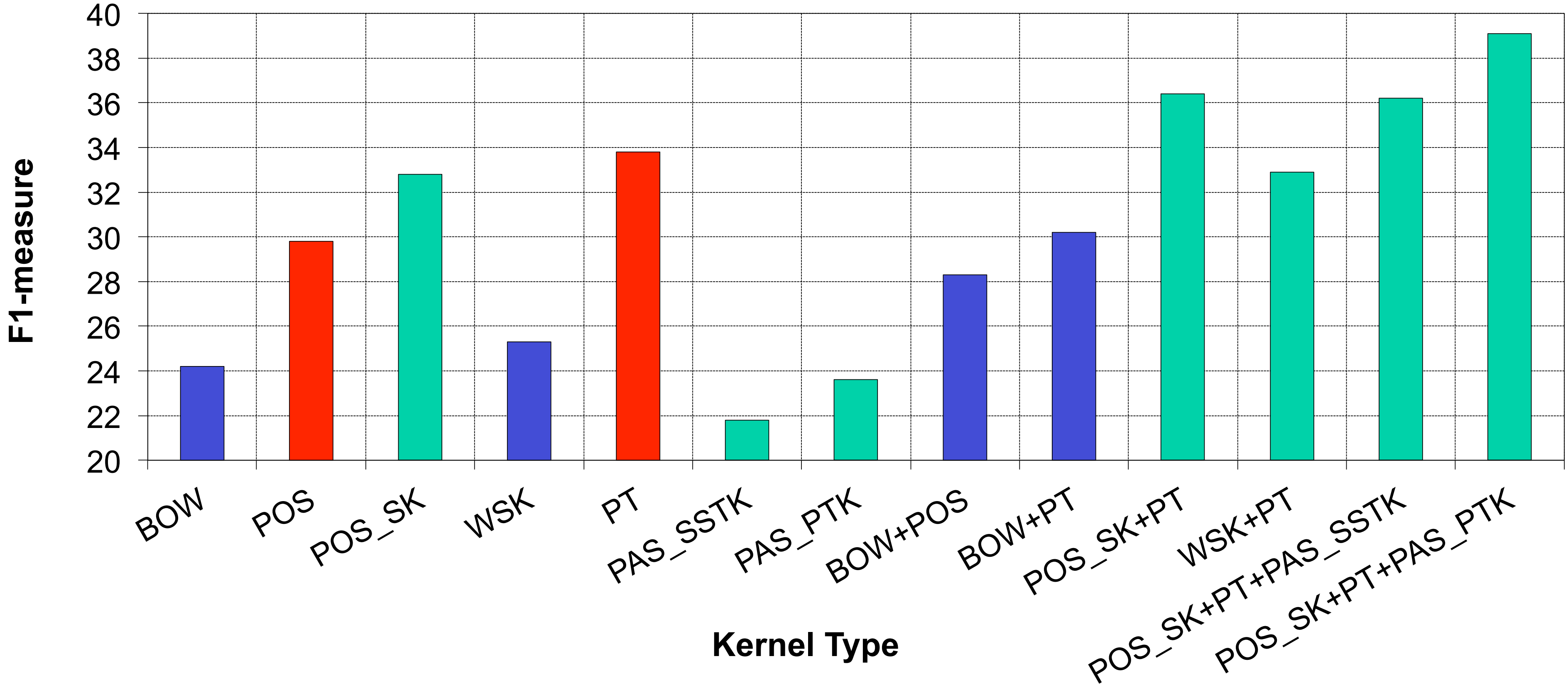
Results on TREC Data
(5 folds cross validation)



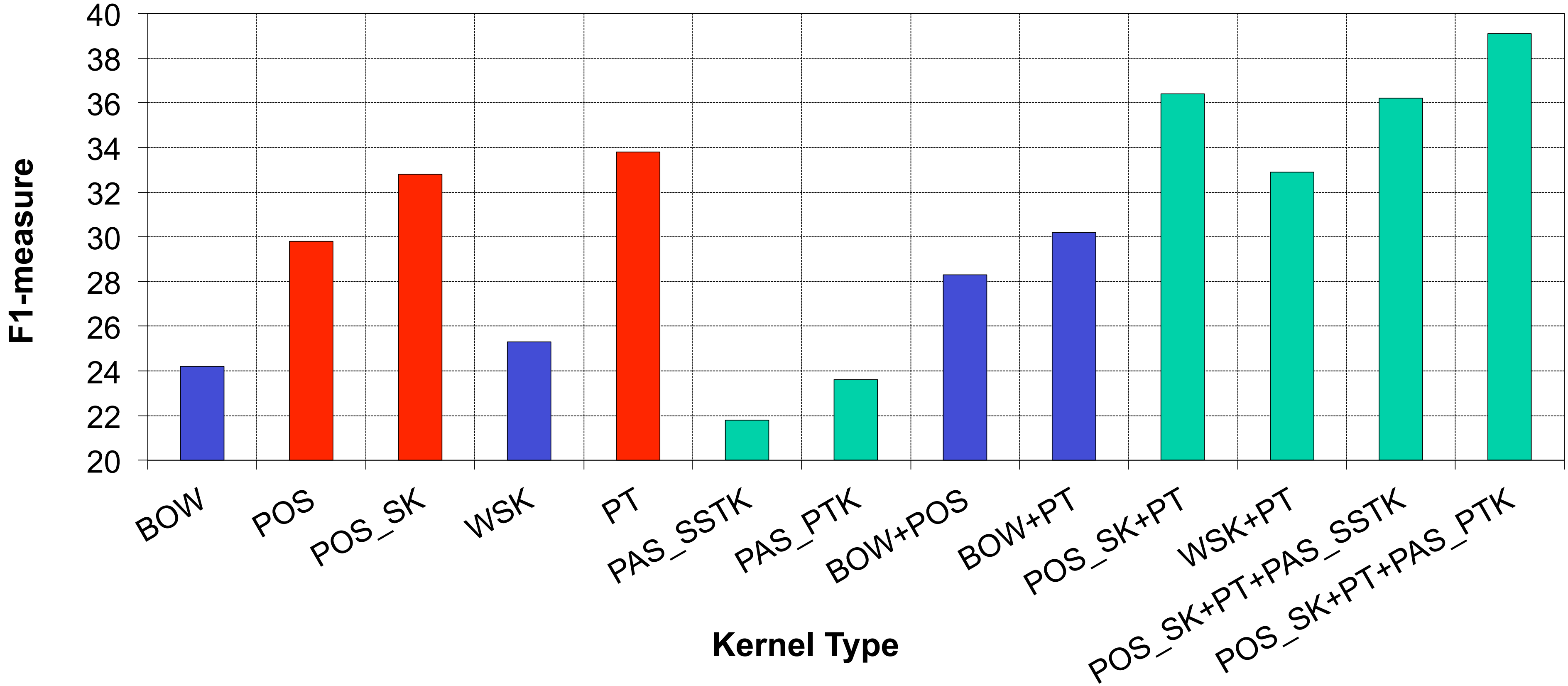
Results on TREC Data
(5 folds cross validation)



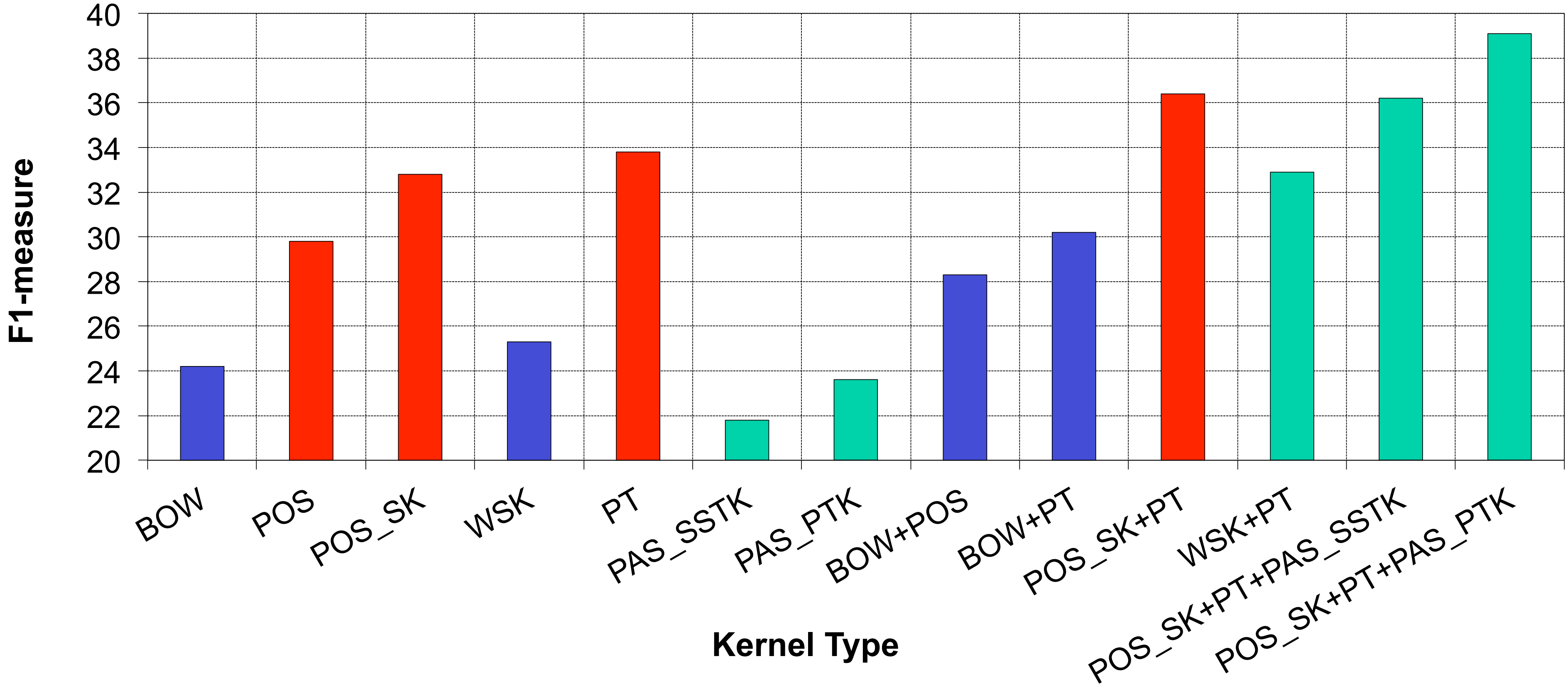
Results on TREC Data
(5 folds cross validation)



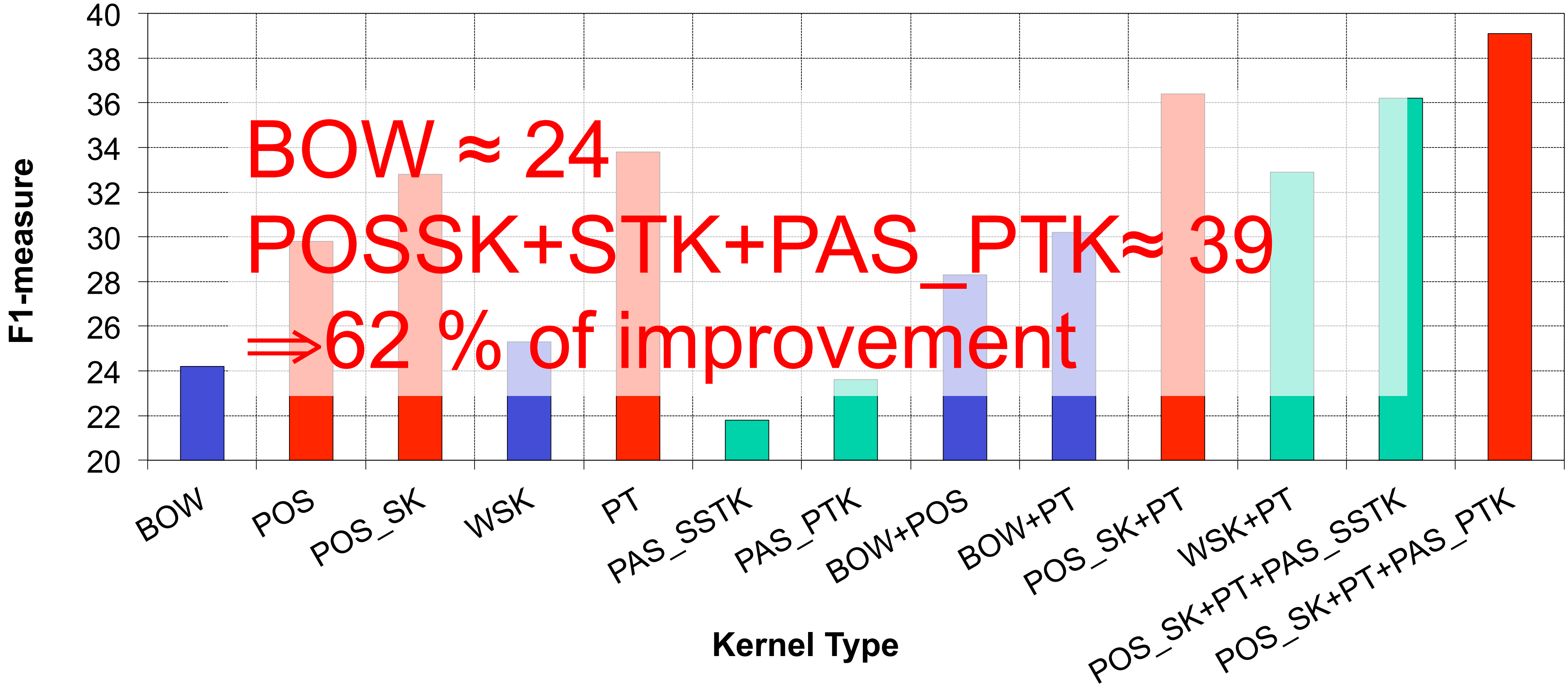
Results on TREC Data
(5 folds cross validation)



Results on TREC Data
(5 folds cross validation)

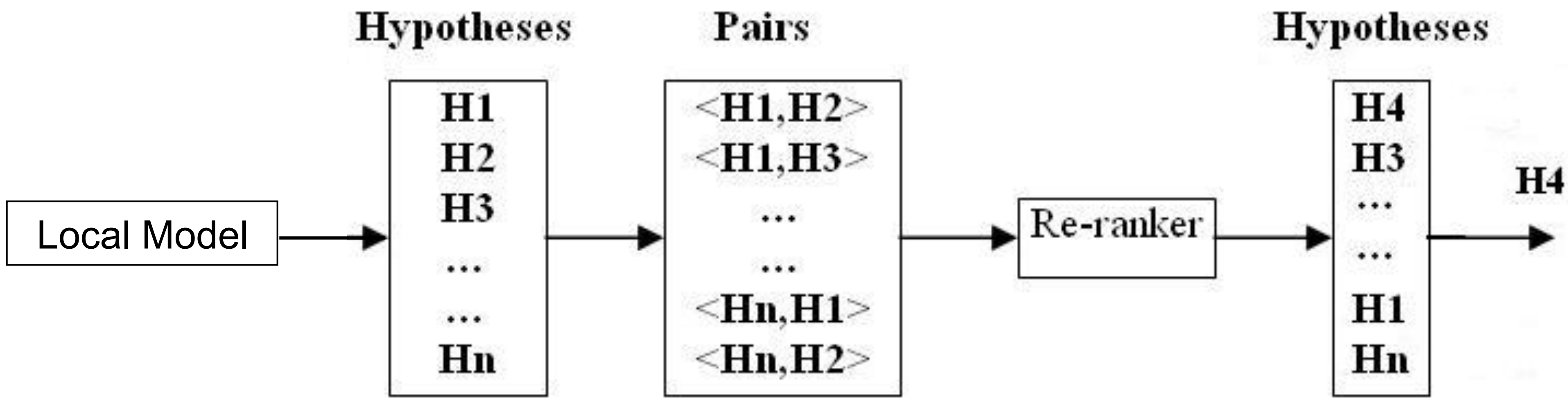


Results on TREC Data
(5 folds cross validation)



Preference Reranking

Framework of Preference Reranking



- The local model is a system providing the initial rank
- Preference reranking is superior to ranking with an instance classifier since it compares pairs of hypotheses

More formally

- Build a set of hypotheses: Q and A pairs
- These are used to build pairs of pairs, $\langle H^i , H^j \rangle$
  - positive instances if H^i is correct and H^j is not correct
- A binary classifier decides if H^i is more probable than H^j
- Each candidate annotation H^i is described by a structural representation
- This way kernels can exploit all dependencies between features and labels

Preference Reranking Kernel

$H_1 > H_2$ and $H_3 > H_4$ then consider training vectors:

$\vec{Z}_1 = \phi(H_1) - \phi(H_2)$ and $\vec{Z}_2 = \phi(H_3) - \phi(H_4) \Rightarrow$ the dot product is:

$$\vec{Z}_1 \bullet \vec{Z}_2 = \left(\phi(H_1) - \phi(H_2)\right) \bullet \left(\phi(H_3) - \phi(H_4)\right) =$$

$$\phi(H_1) \bullet \phi(H_3) - \phi(H_1) \bullet \phi(H_4) - \phi(H_2) \bullet \phi(H_3) + \phi(H_2) \bullet \phi(H_4)$$

$$= K(H_1, H_3) - K(H_1, H_4) - K(H_2, H_3) + K(H_2, H_4)$$

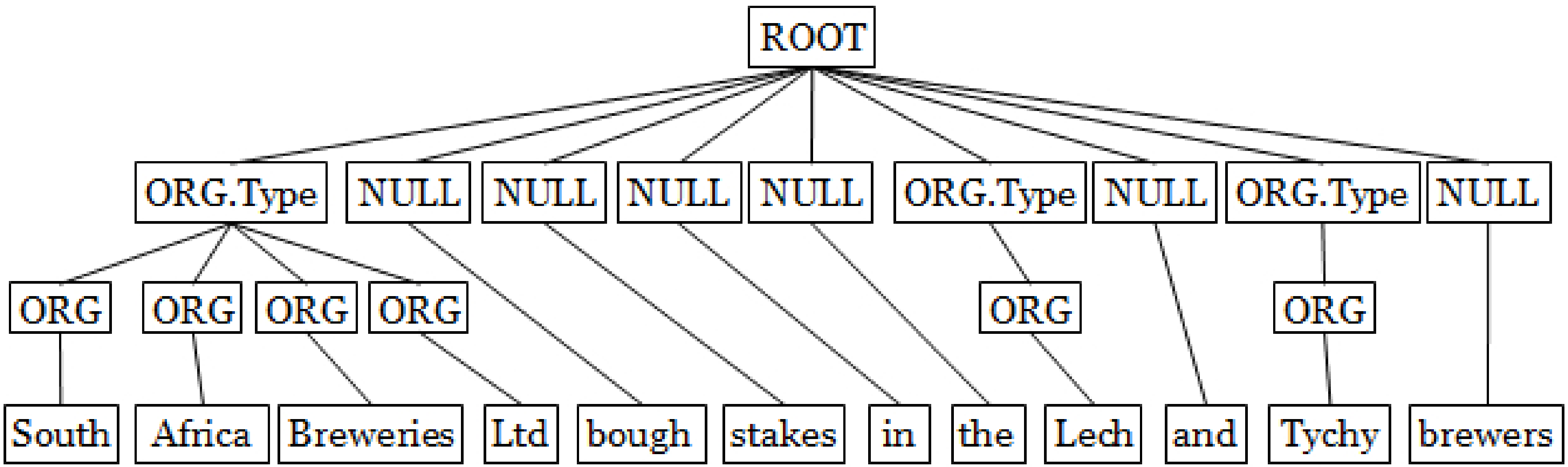
Let $H_i = \langle q_i, a_i \rangle$, $H_j = \langle q_j, a_j \rangle$

$$K(H_i, H_j) = PTK(q_i, q_j) + PTK(a_i, a_j)$$

Syntactic Parsing Reranking

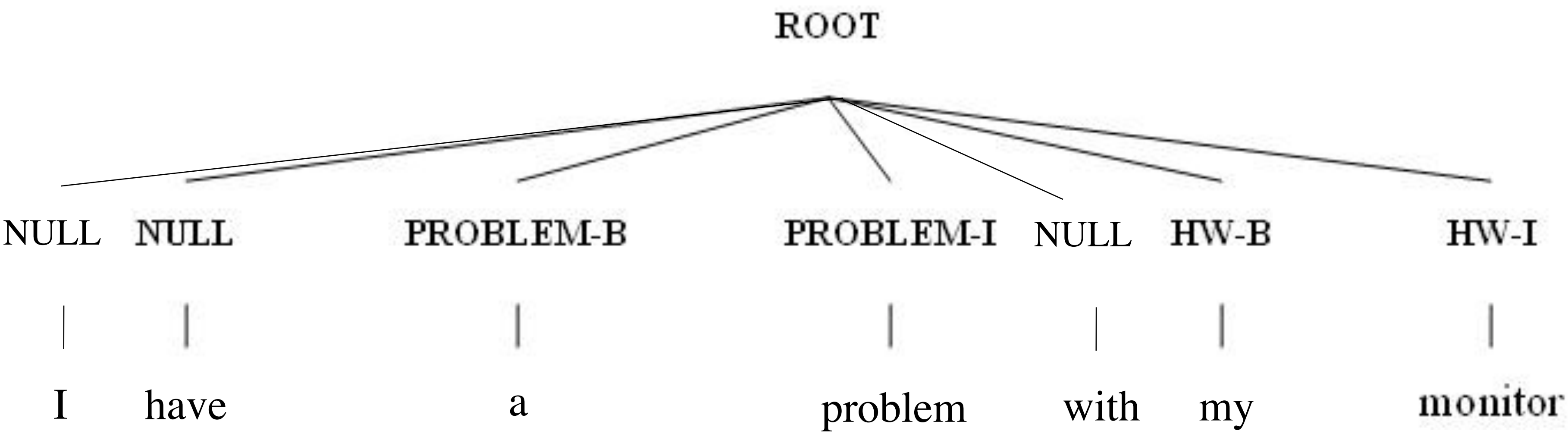
- Pairs of parse trees (Collins and Duffy, 2002)
- N-best parse generated by the Collins' parser
- Re-ranking using STK in a perceptron algorithm

Reranking for Named-Entity Recognition
[Nguyen and Moschitti, AI Journal 2013]



- CRF F1 from 84.86 to 88.16
- Best Italian system F1 82.0, improved to 84.33

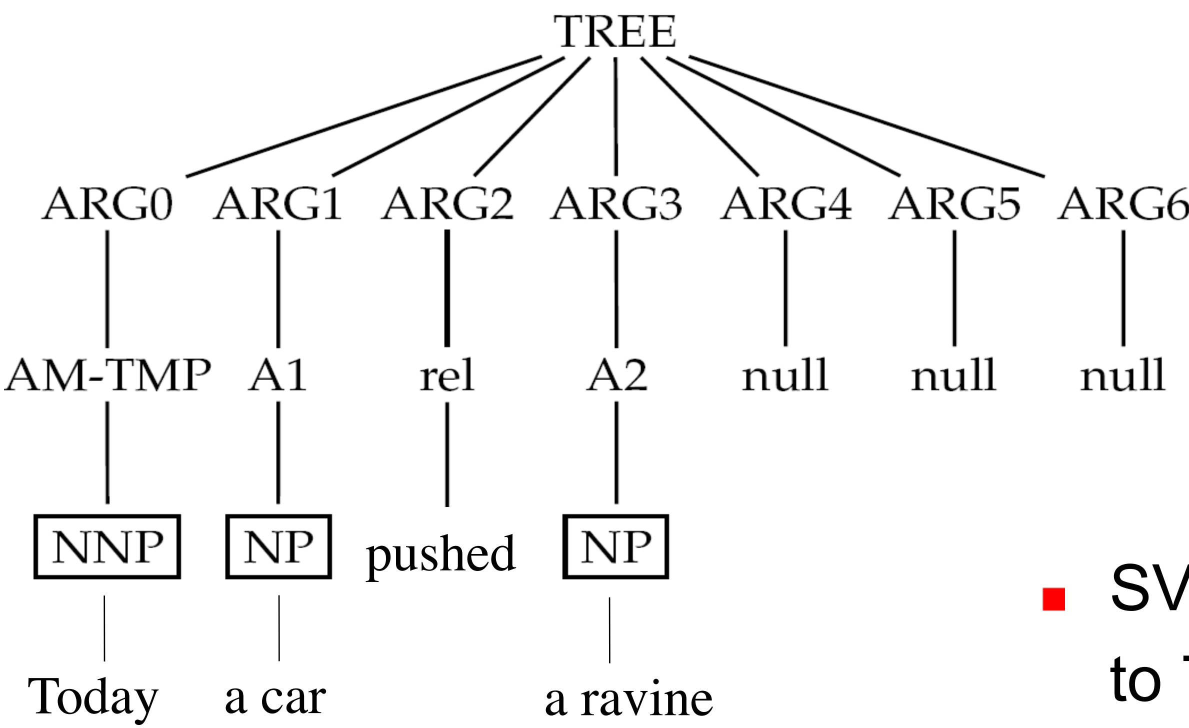
Reranking segmentation and labeling of speech transcription [Dinarelli et al, TASLP 2012]



- It improved the state of the art (CRF) by 2 and 3 points on automatic transcriptions for English and Italian, respectively

Reranking Predicate Argument Structures
[Moschitti et al, CoNLL 2006]

- Today, a car was pushed into a ravine.



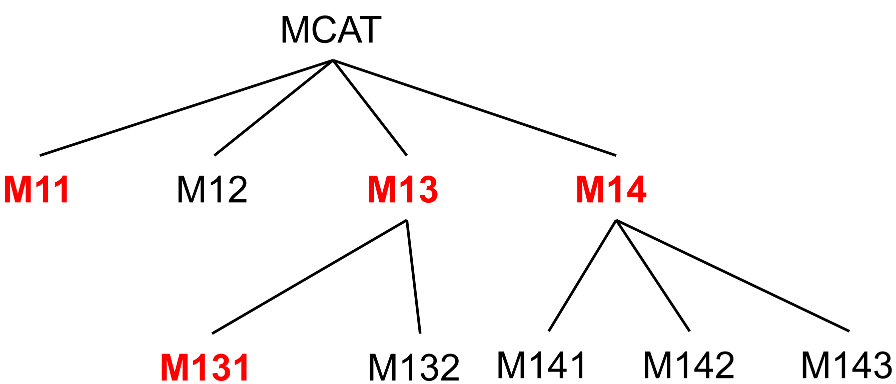
- SVMs F1 from 75.89 to 77.25

Reranking the output of a hierarchical text classifier [Ju et al, ECIR 2013]

- A basic flat multi-label, multi-class classifier builds a set hypotheses
- Represent them with trees
- Apply a tree kernel-based reranker on such trees

Generation with a simple joint probability

- For SVMs, convert scores to probabilities [Platt, 2000]
- Joint probability = product of all category probabilities:
- In red assigned labels (M11, M13, M14, M131)

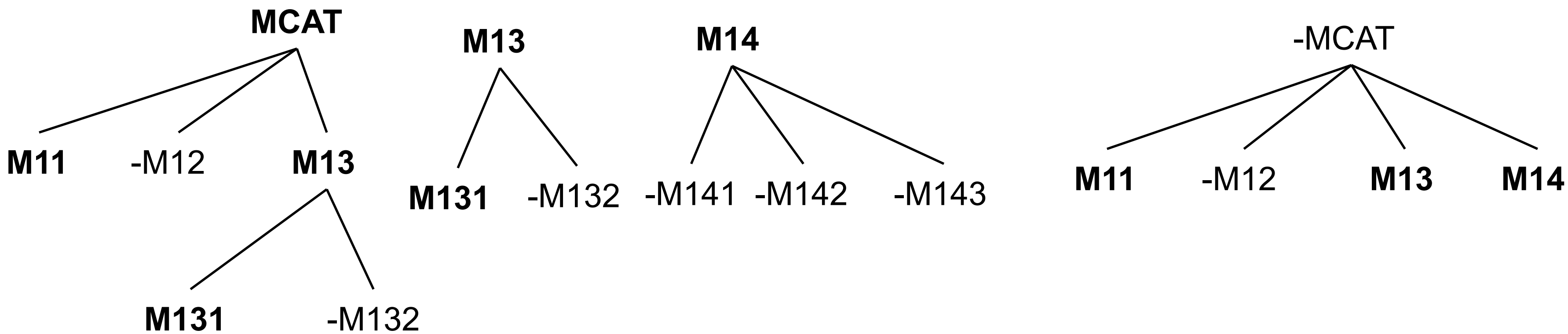


$$P(H) = (1 - p_{MCAT}) \times p_{M_{11}} \times (1 - p_{M_{12}})$$
$$\times p_{M_{13}} \times p_{M_{131}} \times (1 - p_{M_{132}}) \times p_{M_{14}}$$
$$\times (1 - p_{M_{141}}) \times (1 - p_{M_{142}}) \times (1 - p_{M_{143}})$$

- where, p_{M_i} is the probability output by the basic multiclassifier

Vector Generated by Tree Kernels from a Hierarchy Tree

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



- The character "-" is used to indicate that a category was not selected by the flat model

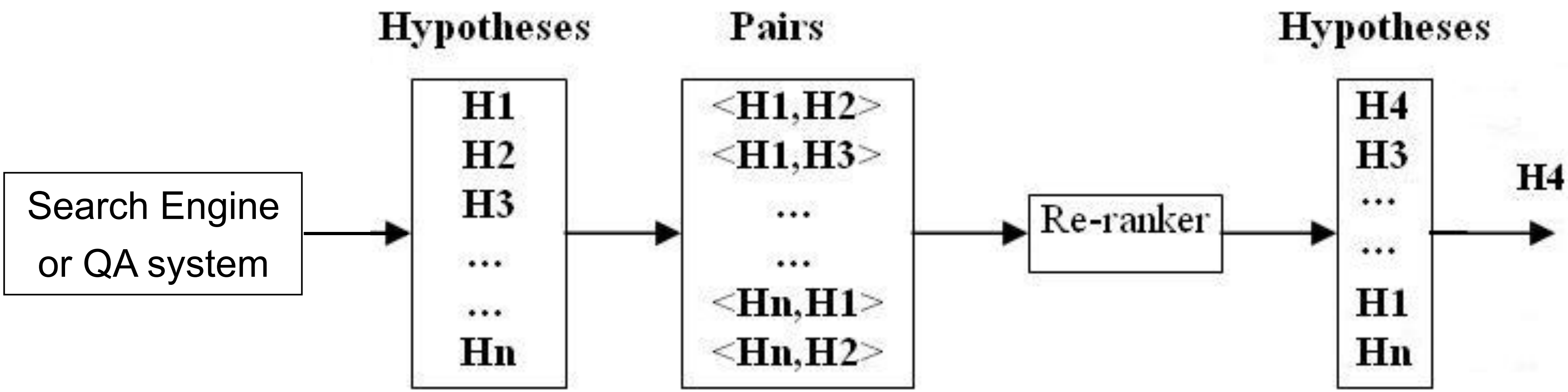
Multi-label, Multi-classification Models

- Baselines
  - **Lewis, flat**: results of one-vs-all from (Lewis' et al, 2004)
  - **Ours, flat**: reimplementation of (Lewis' et al, 2004)
  - **Ours, hier:** implementation of Top-Down model
- Rerankers
  - **SeqRR:** label sequences as features (sequence kernel)
  - **FRR**: Tree Kernels on flat generated hypotheses
  - **HRR**: Tree Kernels on hierarchical generated hypotheses

F1	baseline			our Rerankers		
	Lewis, flat	Ours, flat	Ours, hier	SeqRR	FRR	HRR
Micro-F1	0.816	0.815	0.819	0.828	0.849	0.855
Macro-F1	0.567	0.566	0.578	0.590	0.615	0.634

Relational Kernels for Answer Reranking

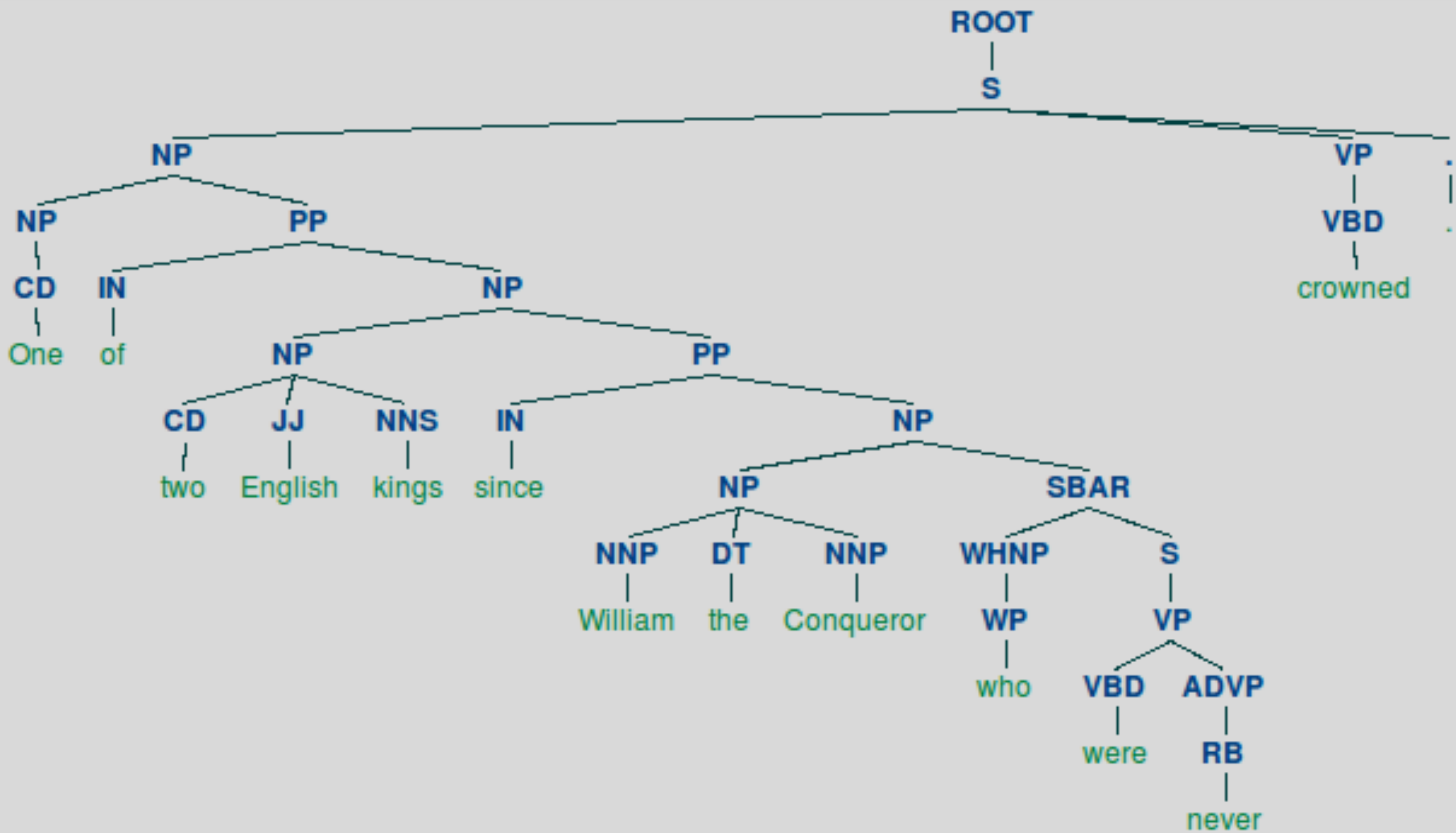
Preference Reranking for documents/ passages

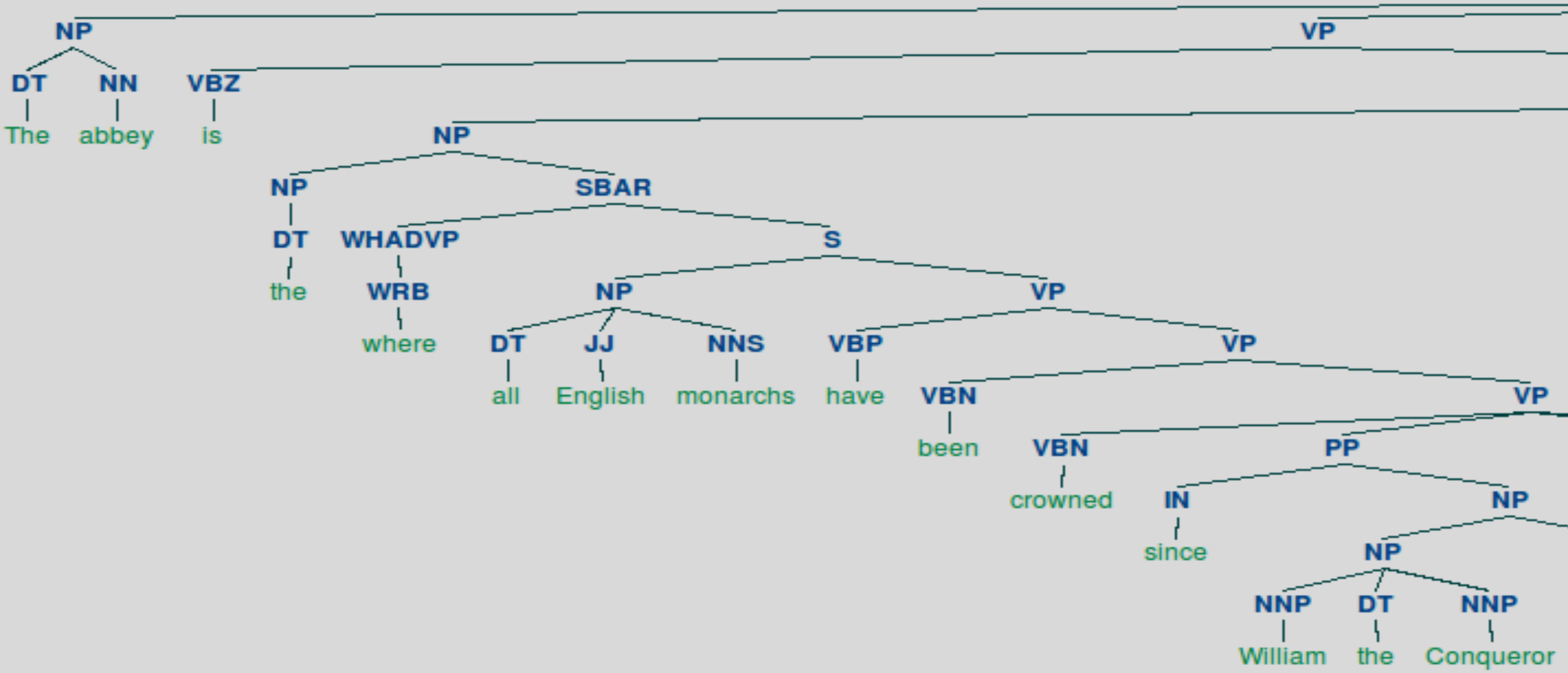


- The initial rank is provided by a search engine (or also a powerful QA system)
- New idea: a boost can be achieved by capturing the relation between question and answer passage

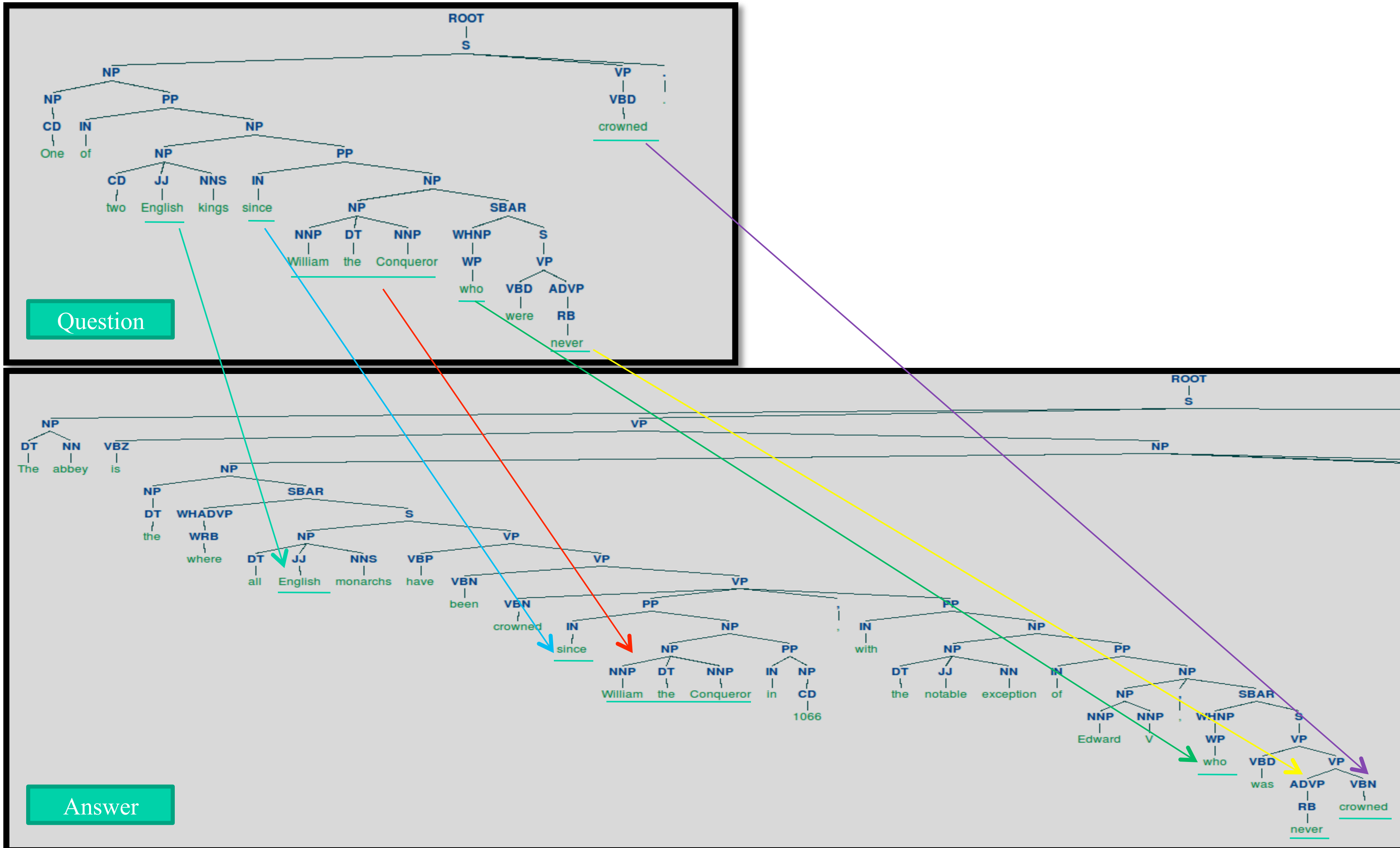

JEOPARDY!

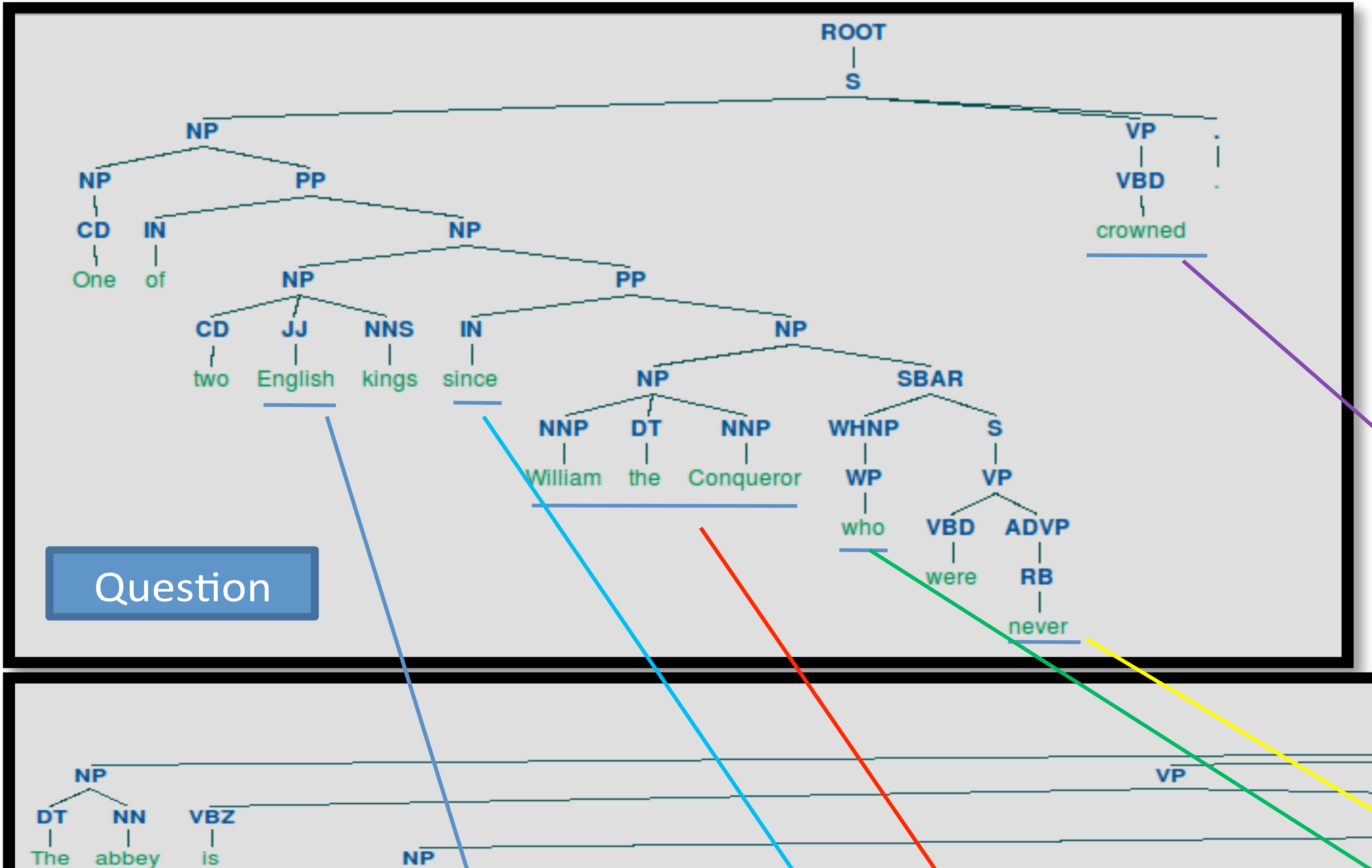
An example of Jeopardy! Question




NP
VP
DT
NN
VBZ
The
abbey
is
NP
NP
SBAR
DT
WHADVP
S
the
WRB
NP
VP
where
DT
JJ
NNS
VBP
VP
all
English
monarchs
have
VBN
VP
been
VBN
PP
crowned
IN
NP
since
NP
NNP
DT
NNP
William
the
Conqueror

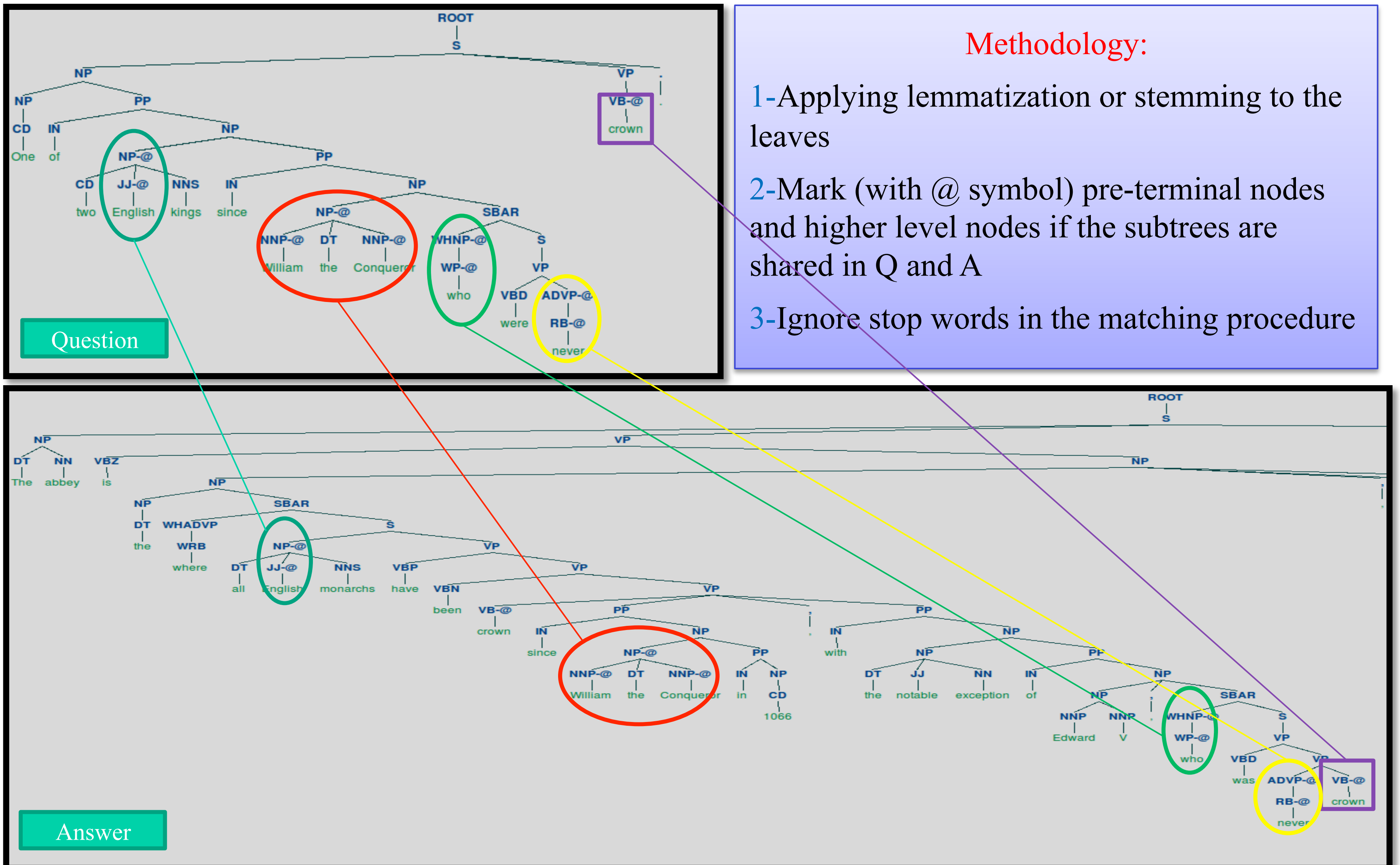
Adding Relational Links





ROOT
S
NP
VP
.
NP
PP
VBD
.
CD
IN
NP
crowned
One
of
NP
PP
CD
JJ
NNS
IN
NP
two
English
kings
since
NP
SBAR
NNP
DT
NNP
WHNP
S
William
the
Conqueror
WP
VP
who
VBD
ADVP
were
RB
never
Question
NP
VP
DT
NN
VBZ
The
abbey
is
NP

Links can be encoded marking tree nodes

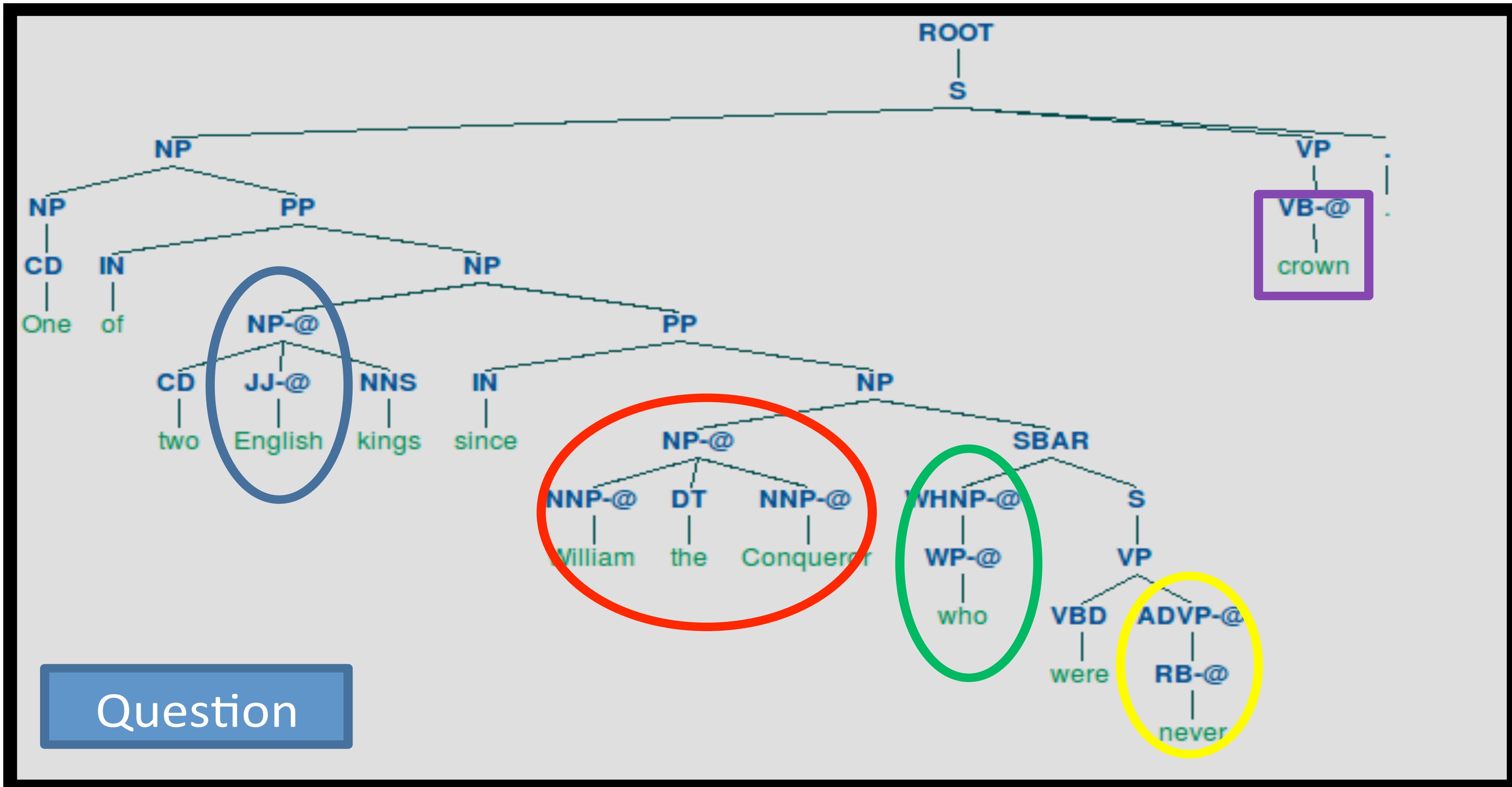


Methodology:

1-Applying lemmatization or stemming to the leaves

2-Mark (with @ symbol) pre-terminal nodes and higher level nodes if the subtrees are shared in Q and A

3-Ignore stop words in the matching procedure


ROOT
S
NP
VP
.
NP
PP
VB-@
crown
.
CD
IN
NP
One
of
NP-@
PP
CD
JJ-@
NNS
IN
NP
two
English
kings
since
NP-@
SBAR
NNP-@
DT
NNP-@
WHNP-@
S
William
the
Conqueror
WP-@
VP
who
VBD
ADVP-@
were
RB-@
never
Question

Representation Issues

- Very large sentences
- The Jeopardy! cues can be constituted by more than one sentence
- The answer is typically composed by several sentences
- Too large structures cause inaccuracies in the kernel similarity and the learning algorithm looses some of its power

Running example from Answerbag

Question: Is movie theater popcorn vegan?

Answer:

(01) Any movie theater popcorn that includes butter -- and therefore dairy products -- is not vegan.

(02) However, the popcorn kernels alone can be considered vegan if popped using canola, coconut or other plant oils which some theaters offer as an alternative to standard popcorn.

Shallow models for Reranking:
[Severyn & Moschitti, SIGIR 2012]

Question

SQ

VBZ NN NN JJ NN

is movie theater popcorn vegan

bag of pos tags

bag of words

and their combination

(is) (movie) (theater) (popcorn) (vegan)

(VBZ) (NN) (NN) (JJ) (NN)

Answer

S

DT NN NN NN WDT VBZ NN CC RB JJ NNS VBZ RB NN

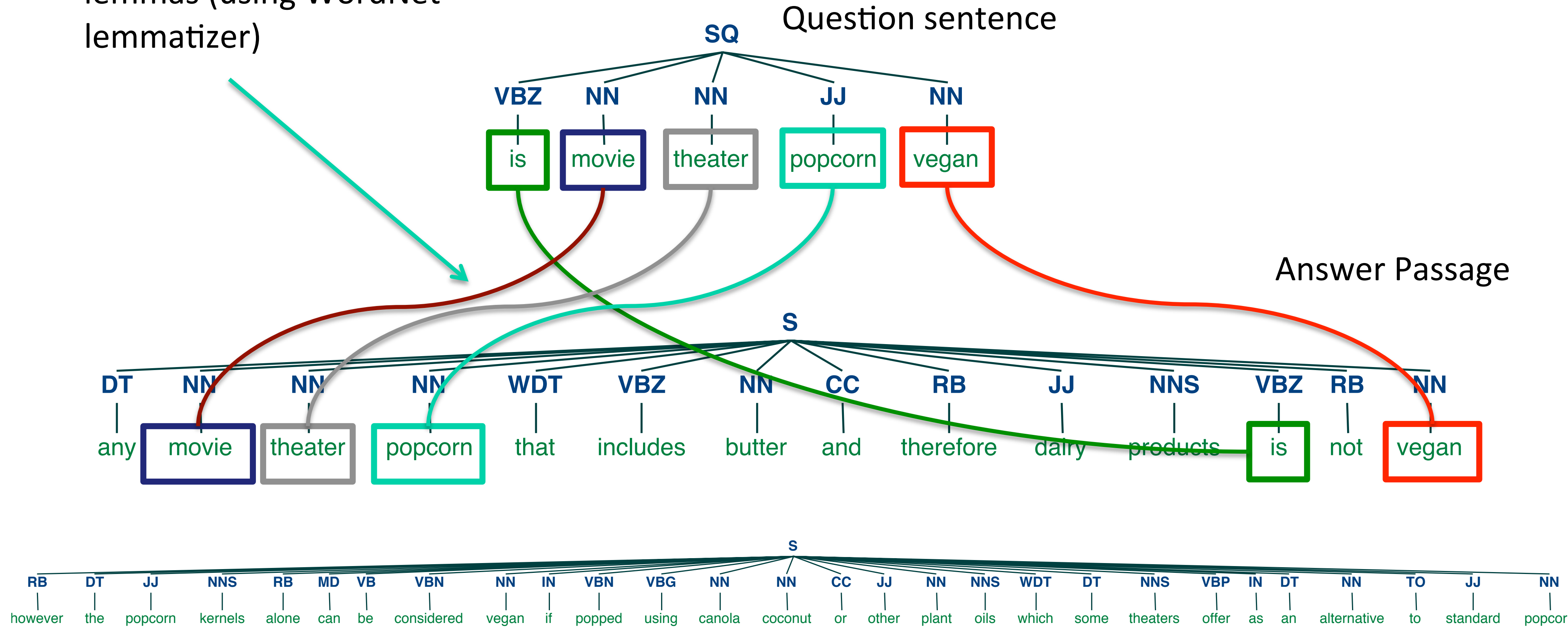
any movie theater popcorn that includes butter and therefore dairy products is not vegan

(any) (movie) (theater) (popcorn) (that) (includes) (butter) (and) (therefore) (dairy) (products) (is) (not) (vegan)

(DT) (NN) (NN) (NN) (WDT) (VBZ) (NN) (CC) (RB) (JJ) (NNS) (VBZ) (RB) (NN)

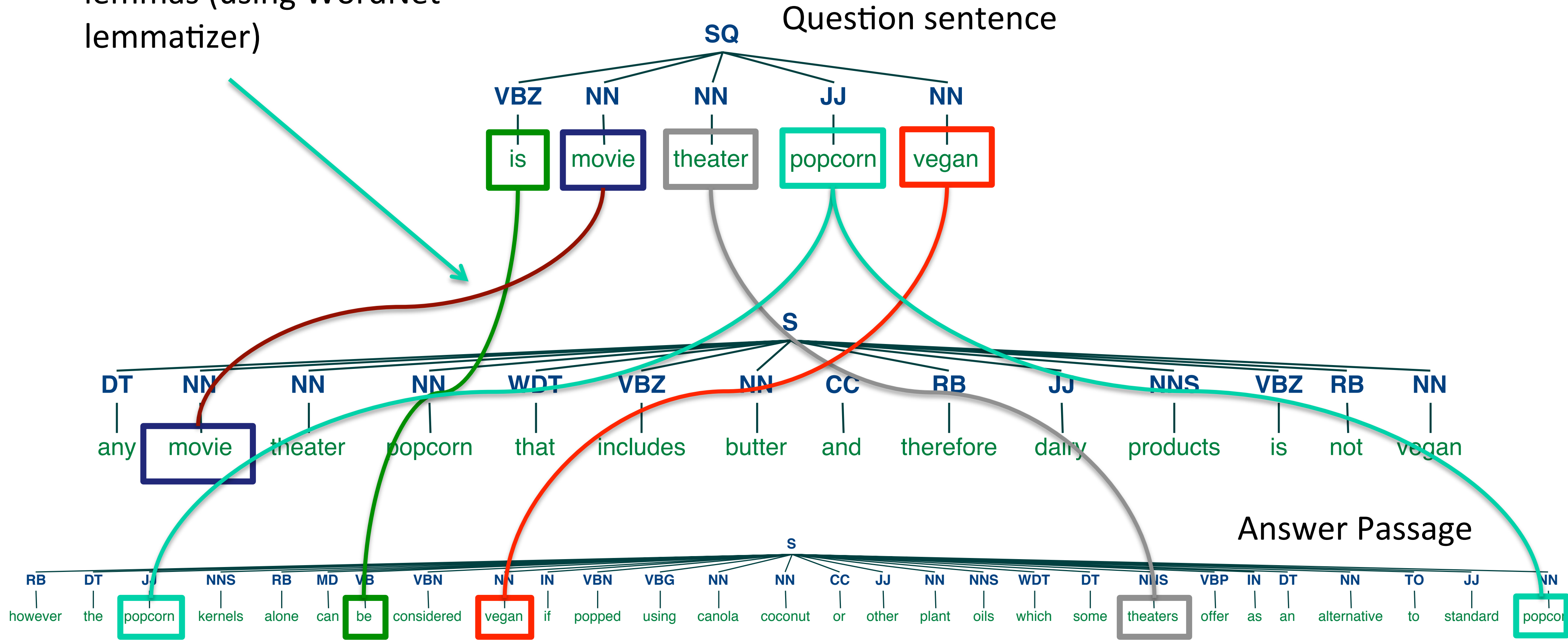
Linking question with the answer 01

Lexical matching is on word lemmas (using WordNet lemmatizer)



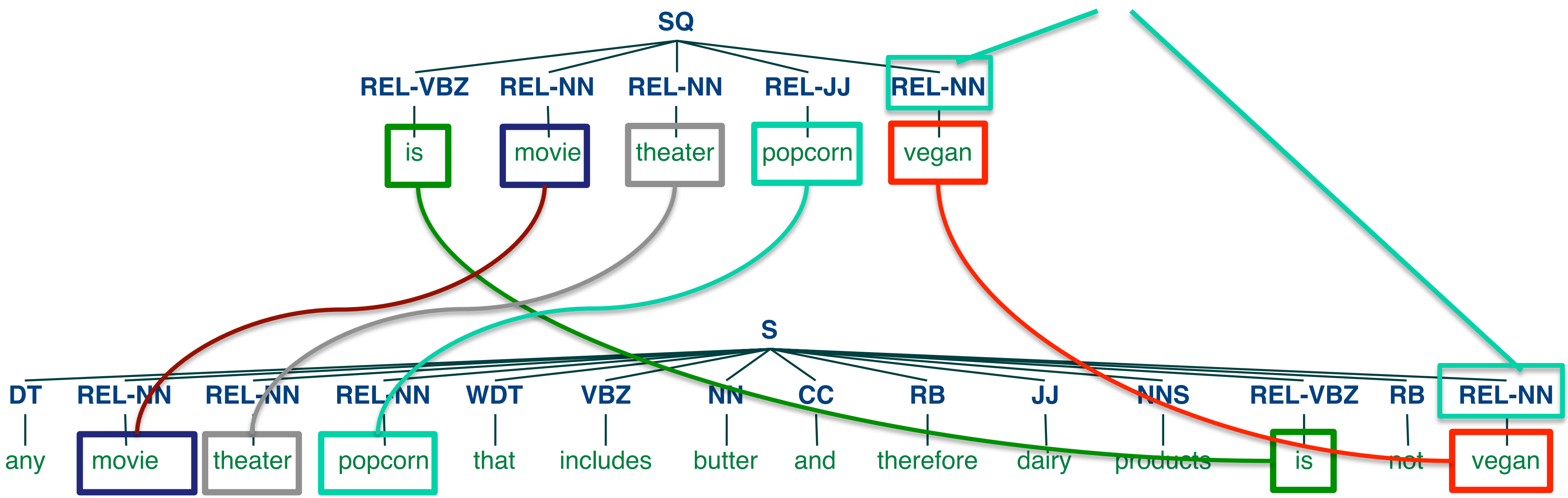
Linking question with the answer 02

Lexical matching is on word lemmas (using WordNet lemmatizer)



Linking question and its answer passages using a relational tag

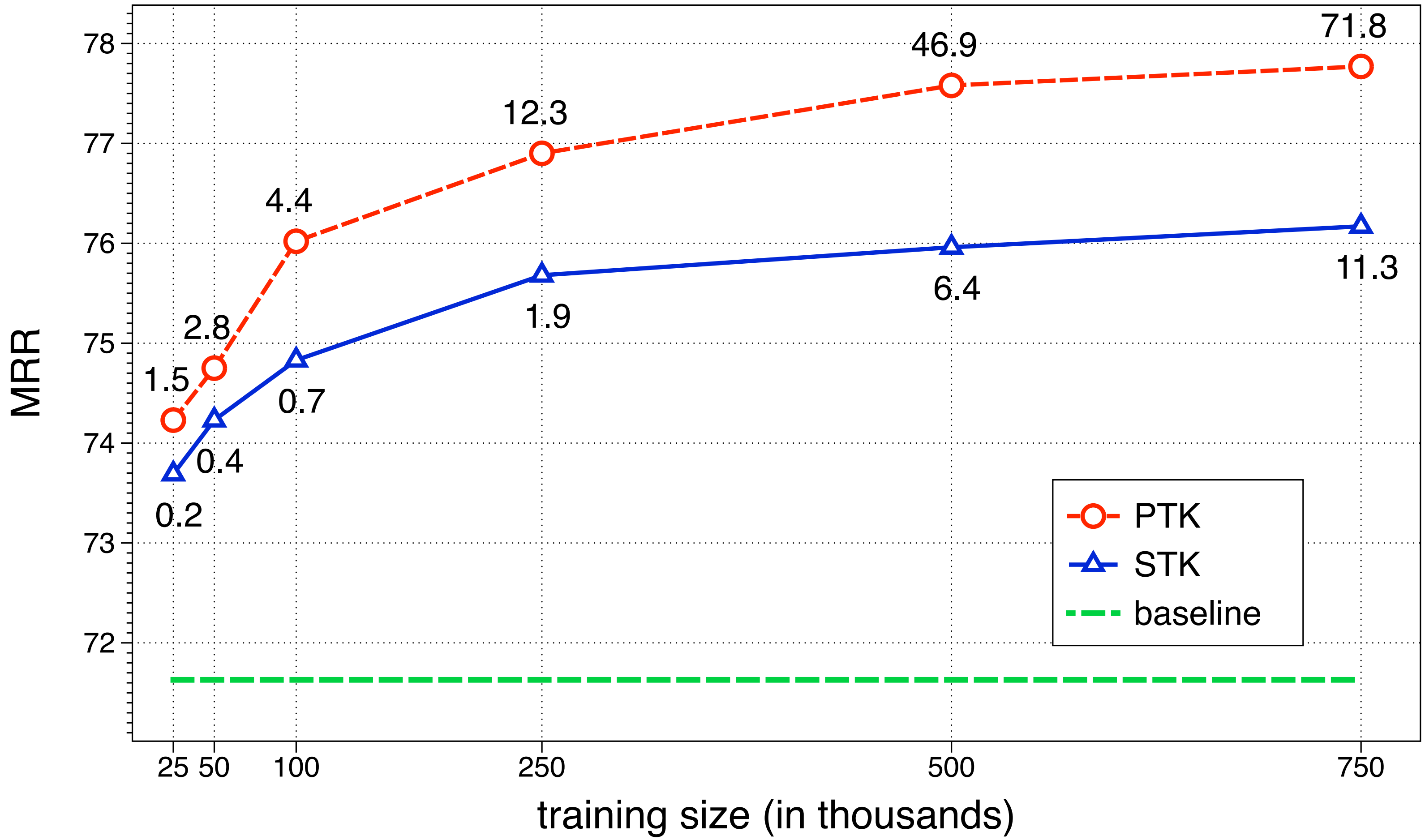
Marking pos tags of the aligned words by a relational tag: “REL”



Answerbag data

- www.answerbag.com: professional question answer interactions
- Divided in 30 categories, Art, education, culture,…
- 180,000 question-answer pairs

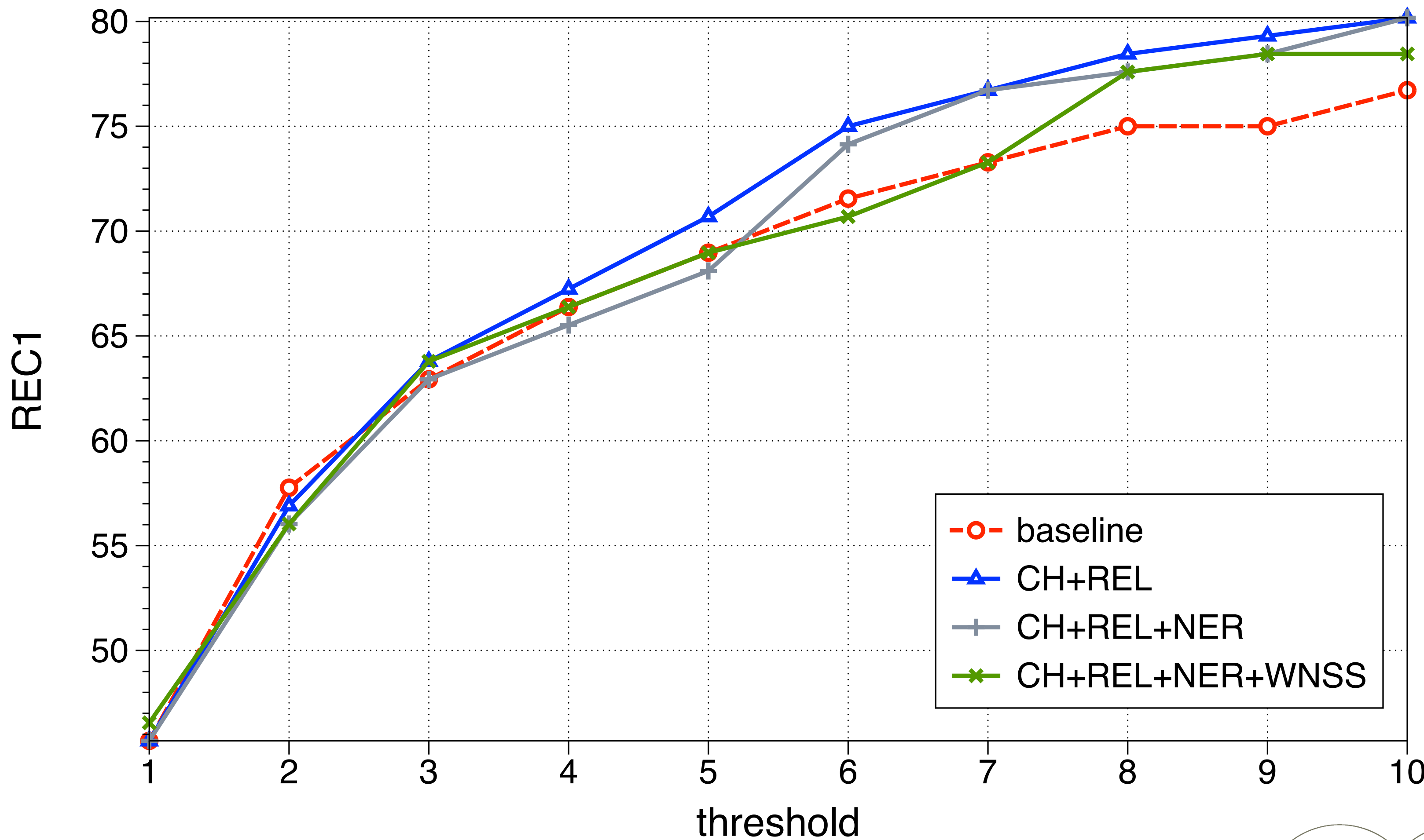
Learning Curve for Answerbag



Jeopardy! data (T9)

- Total number of questions: 517
- 50+ candidate answer passages per question
- Questions with at least one correct answer: 375
- Use only questions with at least one correct answer
- Split the data:
  - Train 70% (259 questions): 63,361 examples for re-ranker
  - Test 30% (116 question): 5,706 examples for re-ranker

Jeopardy! data



Outline: Part III – Advanced Topics

- Large-scale learning with kernels (15 min)
  - Cutting Plane Algorithm for SVMs
  - Sampling methods (uSVMs)
  - Compacting space with DAGs
- Reverse Kernel Engineering (15 min)
  - Model linearization
  - Semantic Role Labeling
  - Question Classification
- Conclusions and Future Directions (5 min)

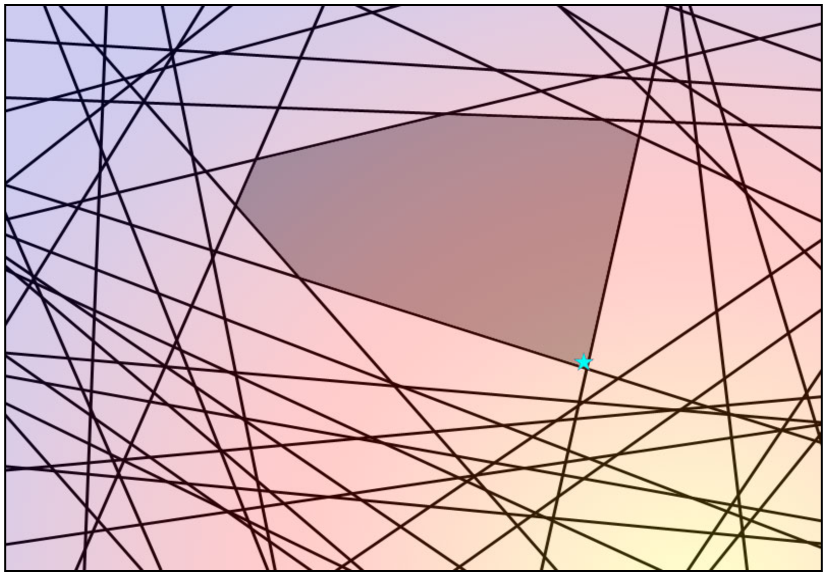
Efficiency Issue

- Working in dual space with SVMs implies quadratic complexity
- Our solutions:
  - Cutting-plane algorithm with sampling uSVMs

    [Yu & Joachims, 2009] [Severyn&Moschitti, ECML PKDD 2010]
  - Compacting SVM models with DAGs

    [Severyn&Moschitti, ECML PKDD 2011]
  - Compacting SVM models with DAGs in online models [Aiolli et al, CIDM 2007]

CPA in a nutshell



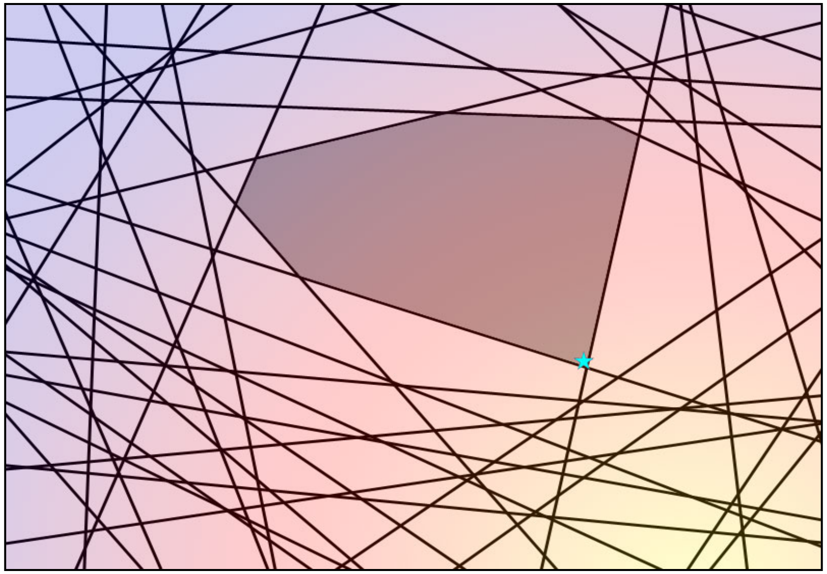
Original SVM Problem

- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

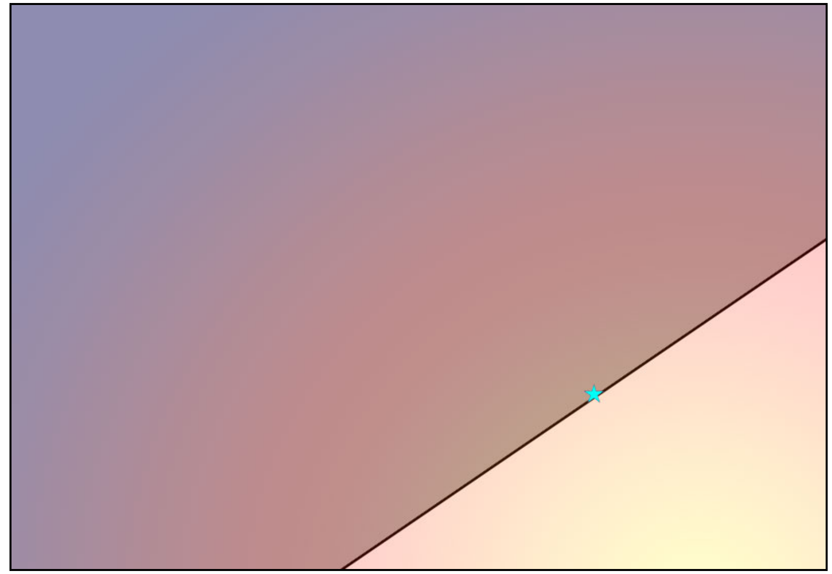
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

CPA in a nutshell



Original SVM Problem

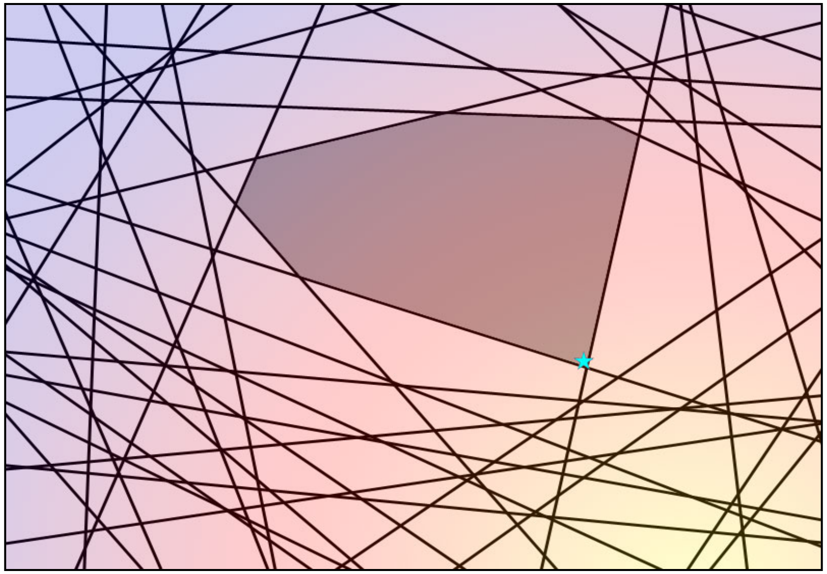
- Exponential constraints
- Most are dominated by a small set of "important" constraints

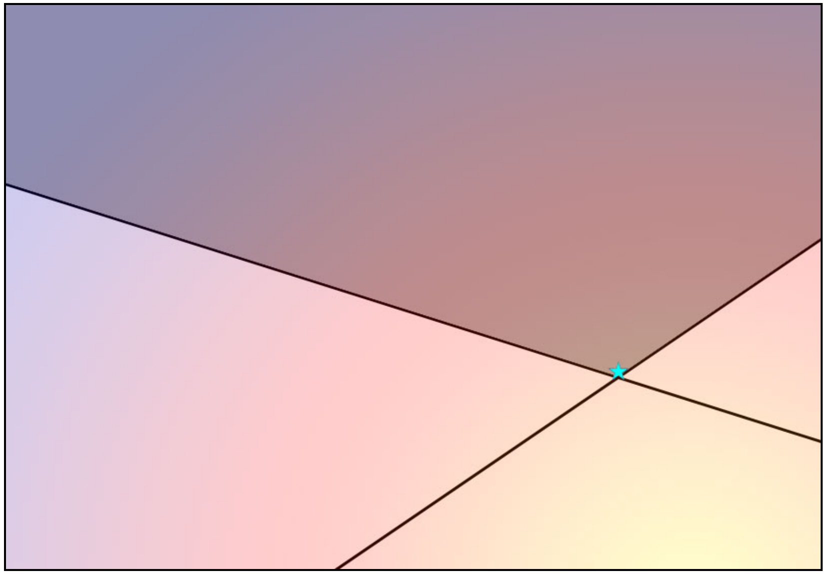


CPA SVM Approach

- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

CPA in a nutshell





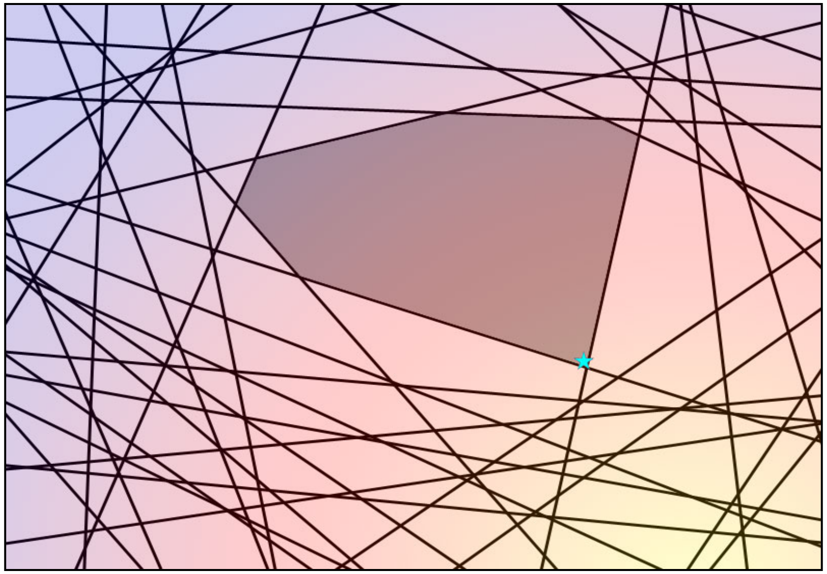
Original SVM Problem

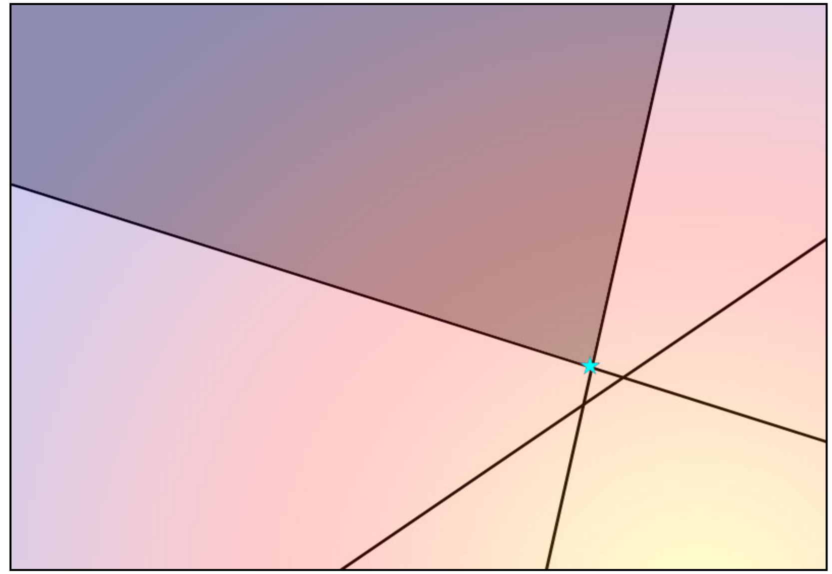
- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

CPA in a nutshell





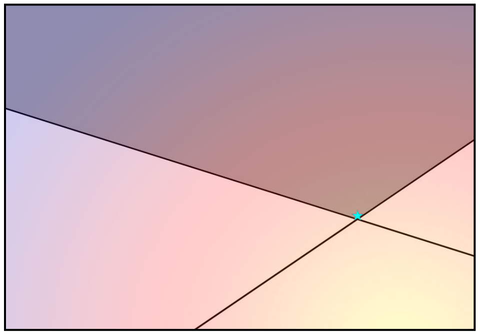
Original SVM Problem

- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

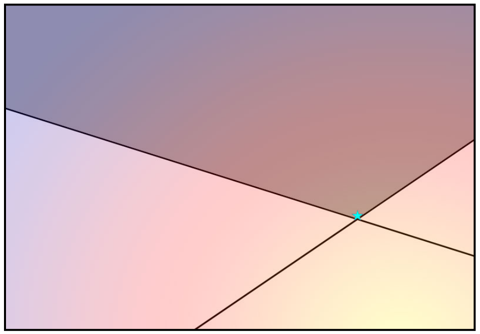
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

Computing most violated constraint (MVC)



$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \vec{g}^{(j)} \cdot \phi(\vec{x}_i)$$

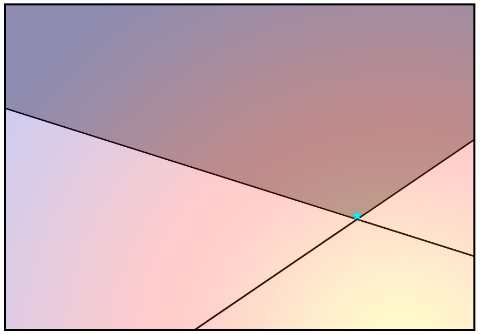
Computing most violated constraint (MVC)



$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \vec{g}^{(j)} \cdot \phi(\vec{x}_i)$$

$$g^{\overrightarrow{(j)}} = \frac{1}{n} \sum_{k=1}^{n} c_k^{(j)} y_k \phi(\vec{x}_k)$$

Computing most violated constraint (MVC)



$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \vec{g}^{(j)} \cdot \phi(\vec{x}_i)$$

$$\vec{g^{(j)}} = \frac{1}{n} \sum_{k=1}^{n} c_k^{(j)} y_k \phi(\vec{x}_k)$$

$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{n} \left(\frac{1}{n} c_k^{(j)} y_k \right) K(\vec{x}_i, \vec{x}_k)$$

Approximate CPA [Yu & Joachims, 2009]

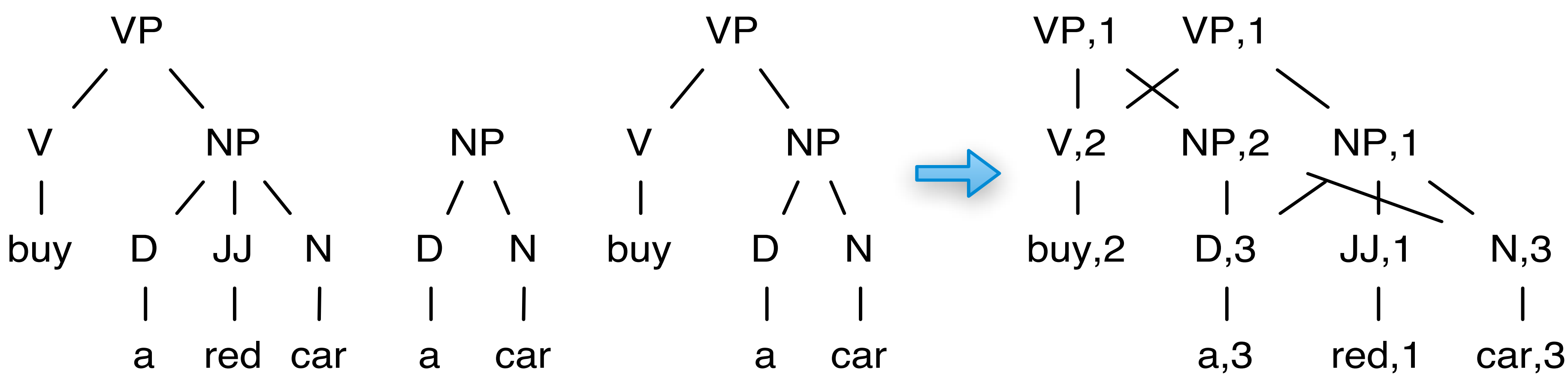
- Main bottleneck to apply kernels comes from the inner product:

$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{n} \left(\frac{1}{n} c_k^{(j)} y_k \right) K(\vec{x}_i, \vec{x}_k)$$

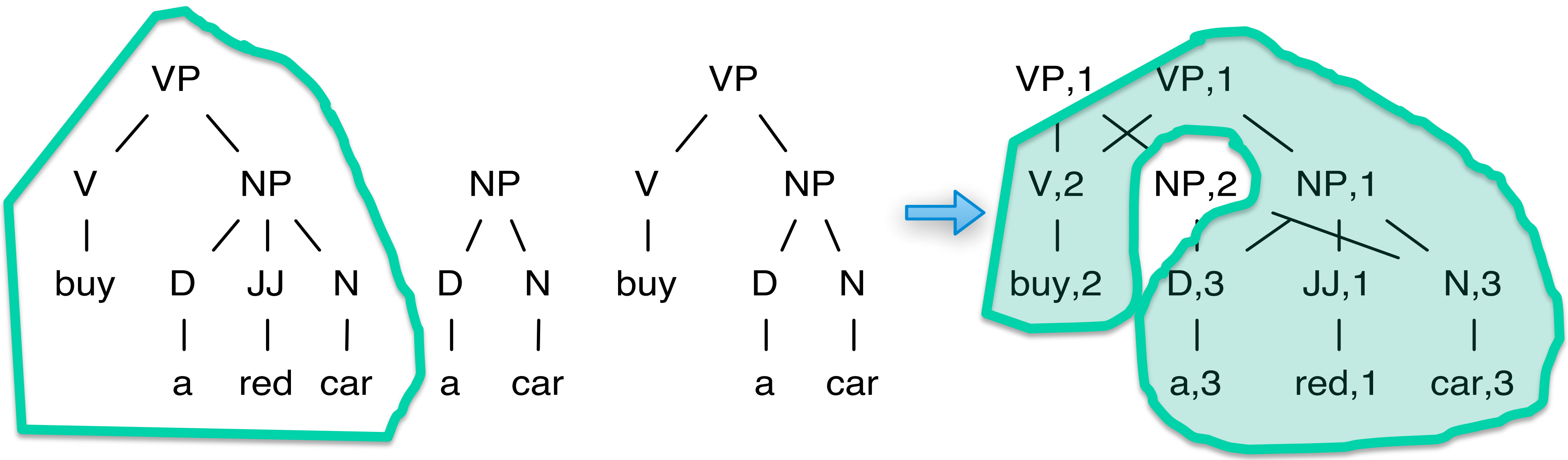
- Use sampling to approximate exact cutting plane models

$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{r} \left(\frac{1}{r} c_k^{(j)} y_k \right) K(\vec{x}_i, \vec{x}_k)$$

Three syntactic trees and the resulting DAG



Three syntactic trees and the resulting DAG



SDAG [Severyn & Moschitti, 2011]

- Compacts each CPA model into a single DAG

$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \boxed{\sum_{k=1}^{r} \left(\frac{1}{r} c_k^{(j)} y_k \right) K(\vec{x}_i, \vec{x}_k)}$$

$$\Downarrow$$

$$\vec{w} \cdot \phi(\vec{x}_i) = \sum_{j=1}^{t} \alpha_j \boxed{K_{dag}(\vec{dag}_{(j)}, \vec{x}_i)}$$

SDAG+

- Compacts all CPA models in the working set into a single DAG

$$\vec{w} \cdot \phi(\vec{x}_i) = \boxed{\sum_{j=1}^{t} \alpha_j \sum_{k=1}^{r} \left(\frac{1}{r} c_k^{(j)} y_k\right) K(\vec{x}_i, \vec{x}_k)}$$

$$\Downarrow$$

$$\vec{w} \cdot \phi(\vec{x}_i) = \boxed{K_{dag}(\widehat{\vec{dag}_{(t)}}, \vec{x}_i)}$$

Outline: Part III – Advanced Topics

- Large-scale learning with kernels (15 min)
  - Cutting Plane Algorithm for SVMs
  - Sampling methods (uSVMs)
  - Compacting space with DAGs
- Reverse Kernel Engineering (15 min)
  - Model linearization
  - Question Classification
- Conclusions and Future Directions (5 min)

Reverse Kernel Engineering
[Pighin & Moschitti, CoNLL 2010 & EMNLP 2009]

- **Input**: an SVM model, i.e., $\vec{w}$
- **Output**: a ranked list of tree fragments
- Intuitively the more a fragment is important the higher is its weight
- Mine tree structures with higher weight first
  - Start from the smallest structures
  - Add nodes to them
  - Stop when reached the max size of the list
- More in detail…

Algorithm 2.1: MINE_MODEL(M, L, E, λ)

$prev \leftarrow \emptyset$; CLEAR_INDEX()
for each $\langle \alpha y, t \rangle \in M$
  do $\begin{cases} T_i \leftarrow \alpha \cdot y / \|t\| \\ \textbf{for each } n \in \mathcal{N}_t \\ \quad \textbf{do} \begin{cases} f \leftarrow \text{FRAG}(n) \;;\; rel = \lambda \cdot T_i \\ prev \leftarrow prev \cup \{f, rel\} \\ \text{PUT}(f, rel) \end{cases} \end{cases}$
$best\_pr \leftarrow$ BEST(L) ;
while **true**
  do $\begin{cases} next \leftarrow \emptyset \\ \textbf{for each } \langle f, rel \rangle \in prev \textbf{ if } f \in best\_pr \\ \quad \textbf{do} \begin{cases} \mathcal{X} = \text{EXPAND}(f, E) \\ rel\_exp \leftarrow \lambda \cdot rel \\ \textbf{for each } frag \in \mathcal{X} \\ \quad \textbf{do} \begin{cases} temp = \{frag, rel\_exp\} \\ next \leftarrow next \cup temp \\ \text{PUT}(frag, rel\_exp) \end{cases} \end{cases} \\ best \leftarrow \text{BEST}(L) \\ \textbf{if not } \text{CHANGED}() \\ \quad \textbf{then break} \\ best\_pr \leftarrow best;\; prev \leftarrow next \end{cases}$
return $(\mathcal{F}_L)$

- Greedy, small to large fragment, recursive exploration of a tree's fragment space
- Basic assumption: consider fragments that span k levels of the tree only if there was at least one fragment spanning $k-1$ levels that is more relevant than those spanning from 0 to $k-2$ levels.
- Basic operations:
  - FRAG(n)
  - EXPAND(f, E)
- Parameters:
  - maxexp (E)
  - threshold value (L)

Mining the weight of a fragment

For a linear SVM:

- Gradient of the hyperplane is: $\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i} = [w^{(1)}, \ldots, w^{(N)}]$
- Cumulative relevance $w^{(j)}$ of the j-th feature: $|w^{(j)}| = \left|\sum_{i=1}^{n} \alpha_i y_i x_i^{(j)}\right|$

For a tree kernel function (i.e.: features $\rightarrow$ fragments):

$$x_i^{(j)} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\|t_i\|} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\sqrt{\sum_{k=1}^{N}(t_{i,k}\lambda^{\ell(f_k)})^2}} \Rightarrow |w^{(j)}| = \left|\sum_{i=1}^{n} \frac{\alpha_i y_i t_{i,j}\lambda^{\ell(f_j)}}{\|t_i\|}\right|$$

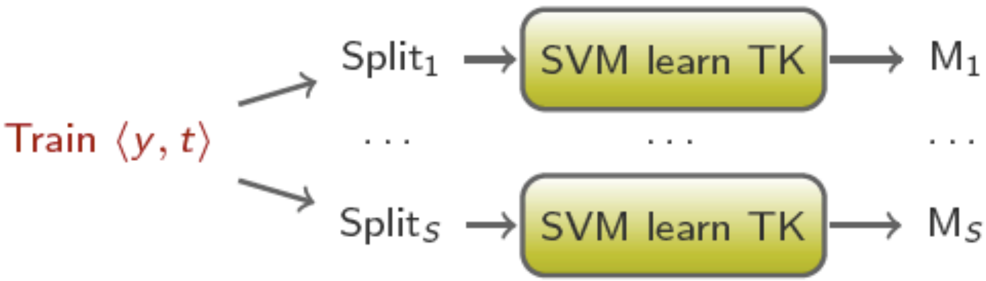
where:

t_i	is the i-th tree in the model
α_i	is the SVM-estimated weight for the tree (and hence, for its fragments)
y_i	is the training label of the tree
f_J	is the fragment associated with the j-th dimension of the feature space
$t_{i,j}$	is the number of occurrences of f_j in t_i
λ	is the kernel decay factor
$\ell(f_j)$	is the depth (number of levels) of the fragment

Reverse Engineering Framework

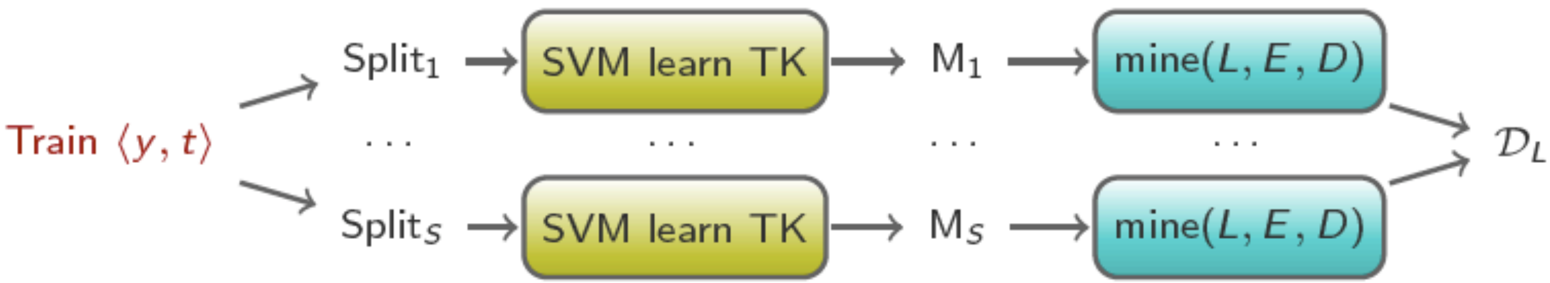
$$\begin{array}{ccc} & \nearrow & \mathsf{Split}_1 \\ \mathsf{Train}\ \langle y, t\rangle & & \cdots \\ & \searrow & \mathsf{Split}_S \end{array}$$

Reverse Engineering Framework



FSL = Fragment Space Learning

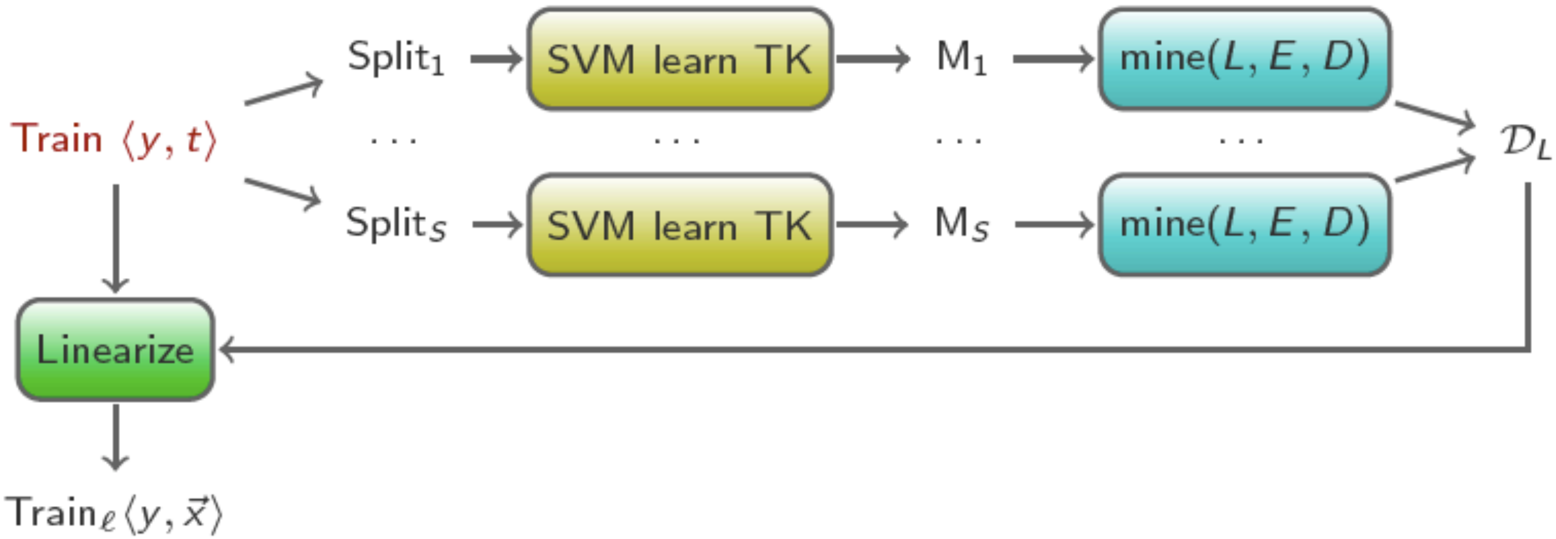
Reverse Engineering Framework



FSL = Fragment Space Learning

FMI = Fragment Mining and Indexing

Reverse Engineering Framework

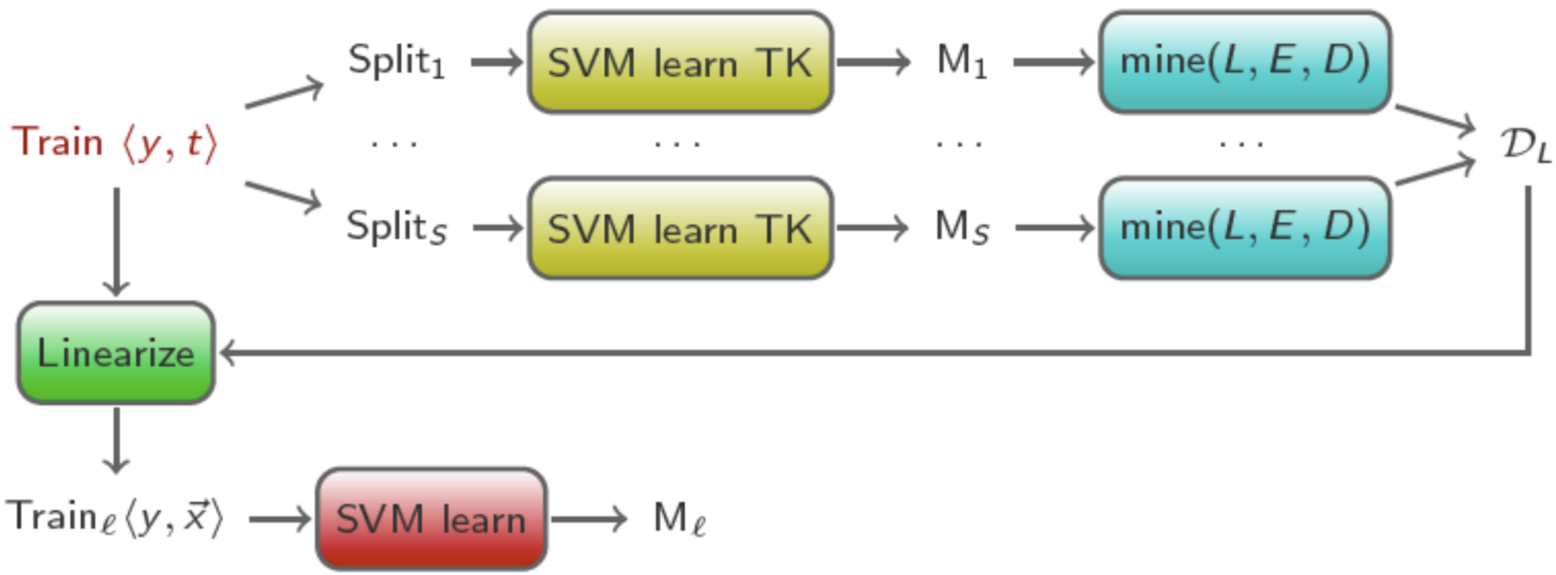


FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

FMI = Fragment Mining and Indexing

Reverse Engineering Framework



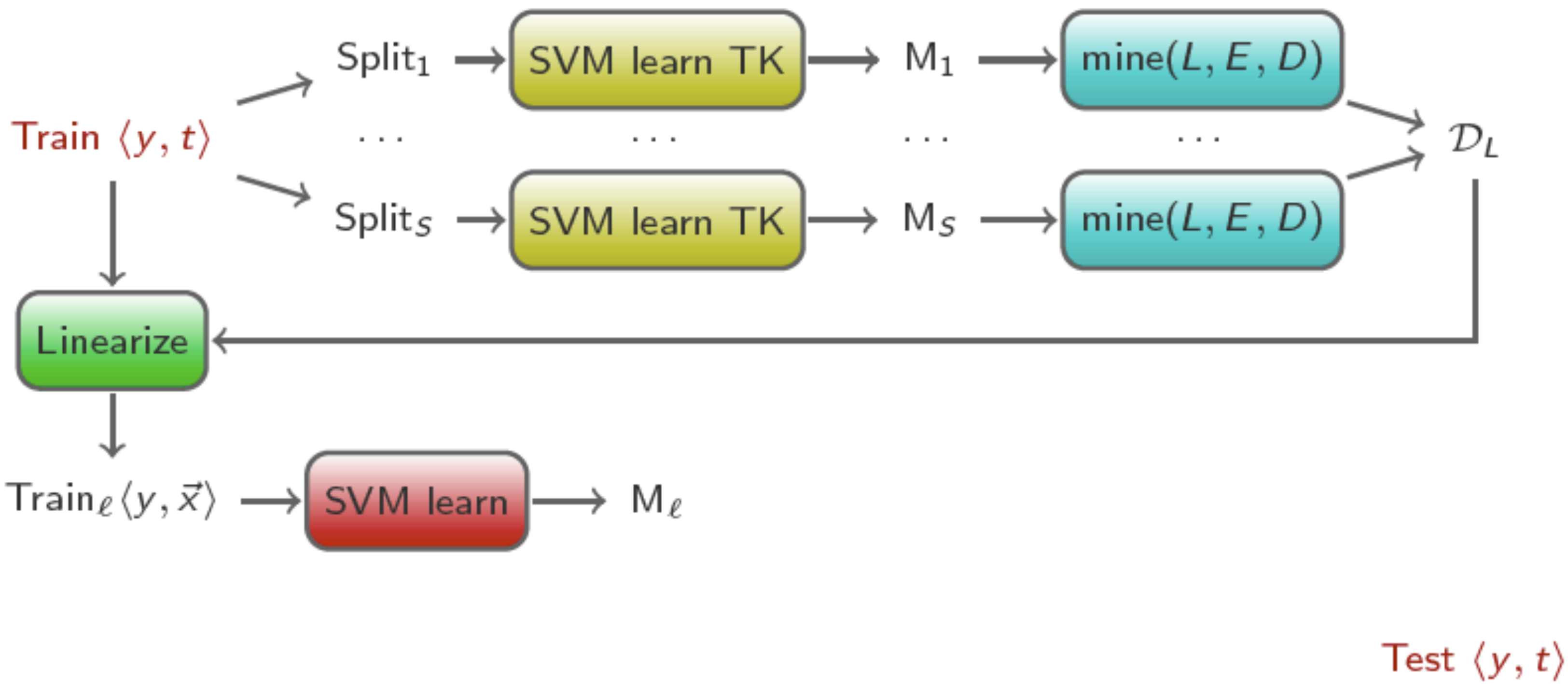
FSL = Fragment Space Learning

FMI = Fragment Mining and Indexing

TFX = Tree Fragment eXtraction

ESL = Explicit Space Learning

Reverse Engineering Framework



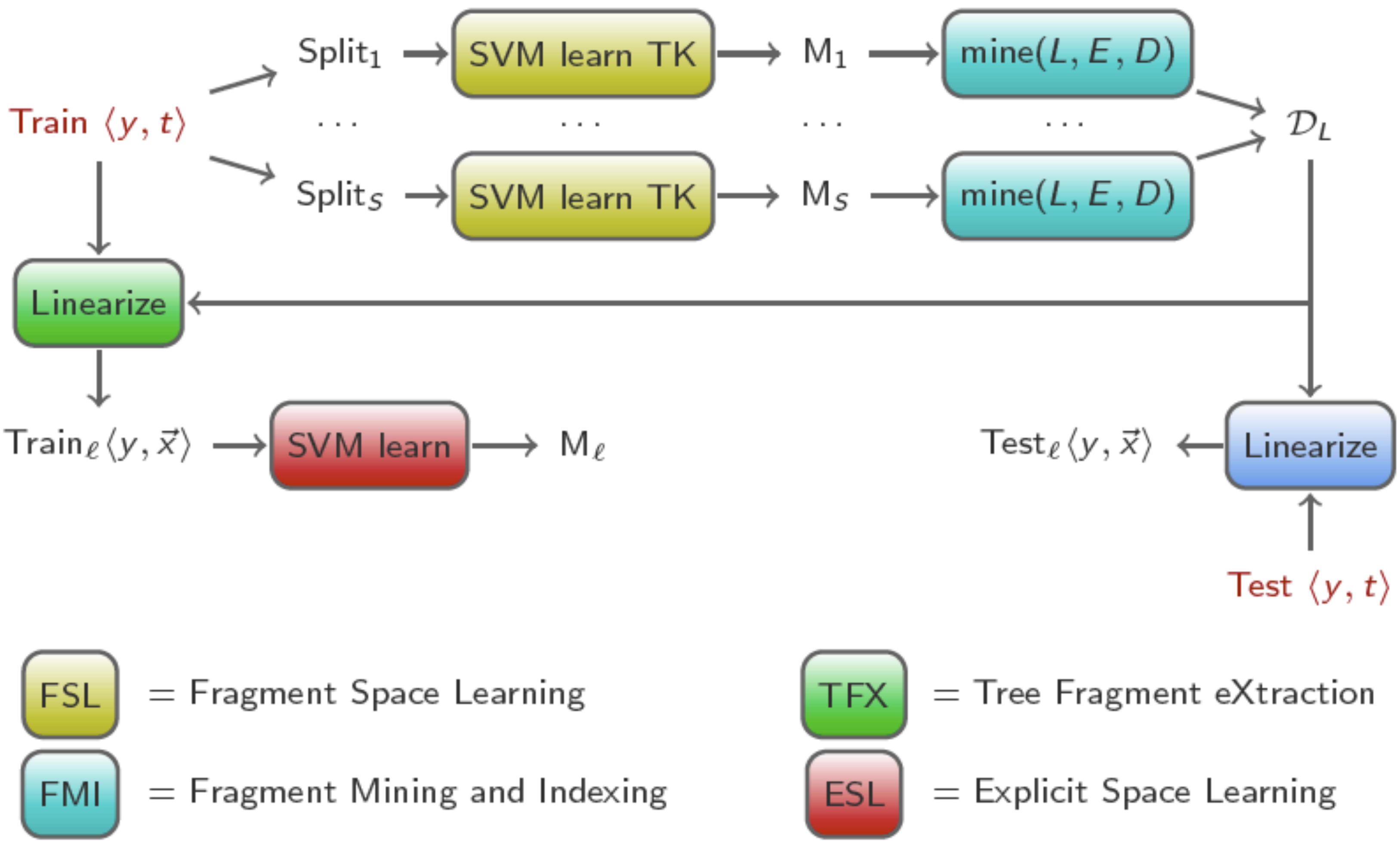
FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

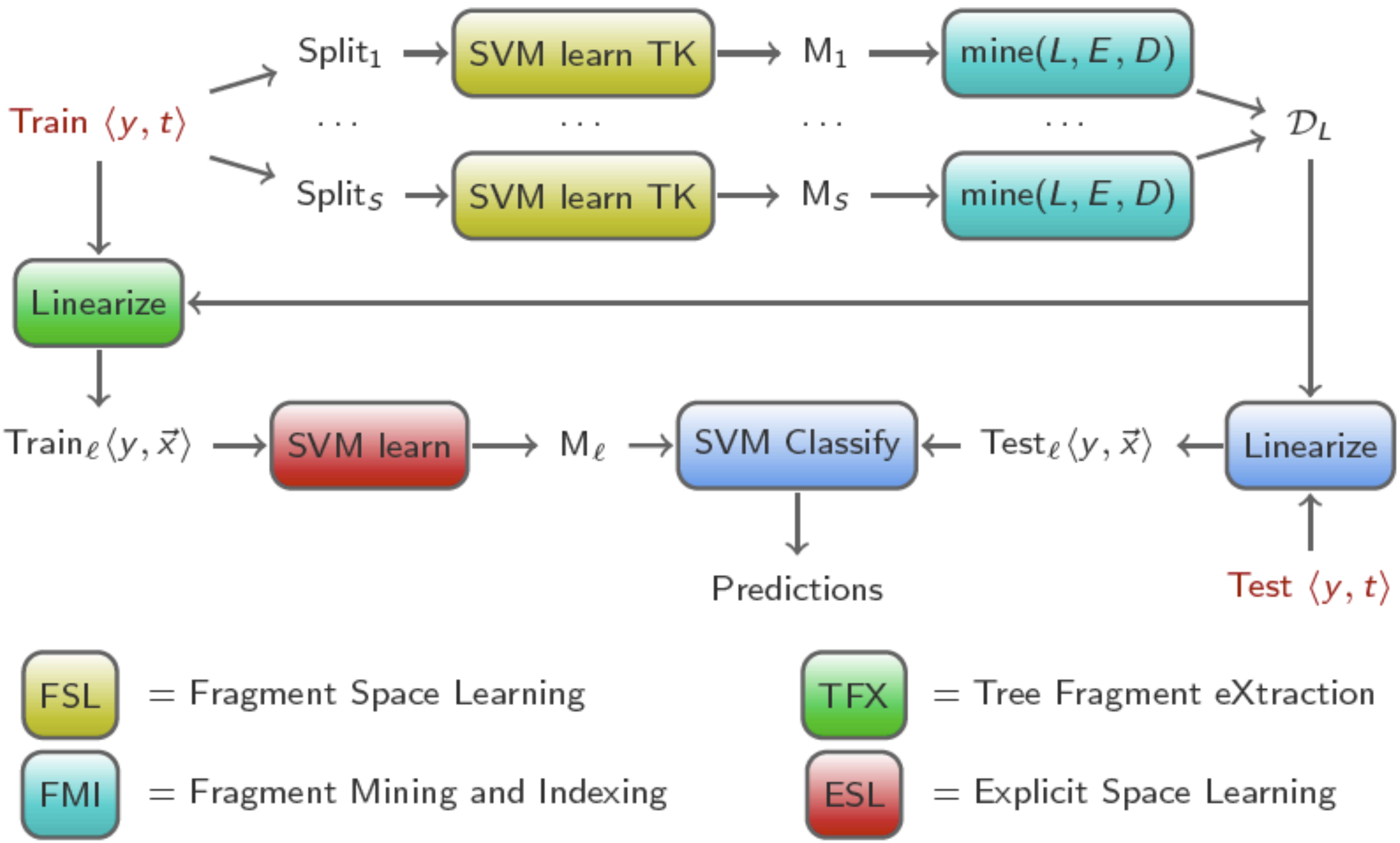
FMI = Fragment Mining and Indexing

ESL = Explicit Space Learning

Reverse Engineering Framework



Reverse Engineering Framework



Reverse Engineering of Kernel Models for Question Classification

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?

Results

- Tr+, Te+: number of positive/negative training instances
- SSTℓ: linearized tree kernel

Class	Data set Tr$^+$	Te$^+$	Accuracy SST	SST$_\ell$
ABBR	89	9	80.0	87.5
DESC	1,164	138	96.0	94.5
ENTY	1,269	94	63.9	63.5
HUM	1,231	65	88.1	87.2
LOC	834	81	77.6	77.9
NUM	896	113	80.4	80.8
Overall			86.2	86.6

Interpretation (Abbreviation Class)

(NN(abbreviation))

(NP(DT)(NN(abbreviation)))

(NP(DT(the))(NN(abbreviation)))

(IN(for))

(VB(stand))

(VBZ(does))

(PP(IN))

(VP(VB(stand))(PP))

(NP(NP(DT)(NN(abbreviation)))(PP))

(SQ(VBZ)(NP)(VP(VB(stand))(PP)))

(SBARQ(WHNP)(SQ(VBZ)(NP)(VP(VB(stand))(PP)))(.))

(SQ(VBZ(does))(NP)(VP(VB(stand))(PP)))

(VP(VBZ)(NP(NP(DT)(NN(abbreviation)))(PP)))

Interpretation (Numeric Class)

(WRB(How))
(WHADVP(WRB(When)))
(WRB(When))
(JJ(many))
(NN(year))
(WHADJP(WRB)(JJ))
(NP(NN(year)))
(WHADJP(WRB(How))(JJ))
(NN(date))
(SBARQ(WHADVP(WRB(When)))(SQ)(.(?)))
(SBARQ(WHADVP(WRB(When)))(SQ)(.))
(NN(day))

Interpretation (Description Class)

(WRB(Why))
(WHADVP(WRB(Why)))
(WHADVP(WRB(How)))
(WHADVP(WRB))
(VB(mean))
(VBZ(causes))
(VB(do))
(SBARQ(WHADVP(WRB(How)))(SQ))
(WRB(How))
(SBARQ(WHADVP(WRB(How)))(SQ)(.))
(SBARQ(WHADVP(WRB(How)))(SQ)(.(?)))

Conclusions

- Using semantic and structural representations is difficult:
  - How to engineer rules for exploiting syntactic/semantic information?
  - How to engineer features for learning algorithms?
- We can use powerful ML algorithms and kernel methods
  - Kernels can generate many features
  - SVMs are robust to noise and irrelevant features
- IDEA: using structural representations of data and similarity functions (Kernel Methods)
  - Structural syntactic/semantic similarity for text

Conclusions (cont'd)

- Kernel methods and SVMs are powerful tools for:
  - building complex classifiers, e.g., question classification or relation extraction; and
  - the design of learning to rank algorithms
- State of the art when reranking NLP/IR systems
  - Named Entity Recognizers
  - Predicate Argument Structures
  - Segmented and labeled Speech Transcriptions
  - Hierarchical text classifier output
  - Passages with relational representations

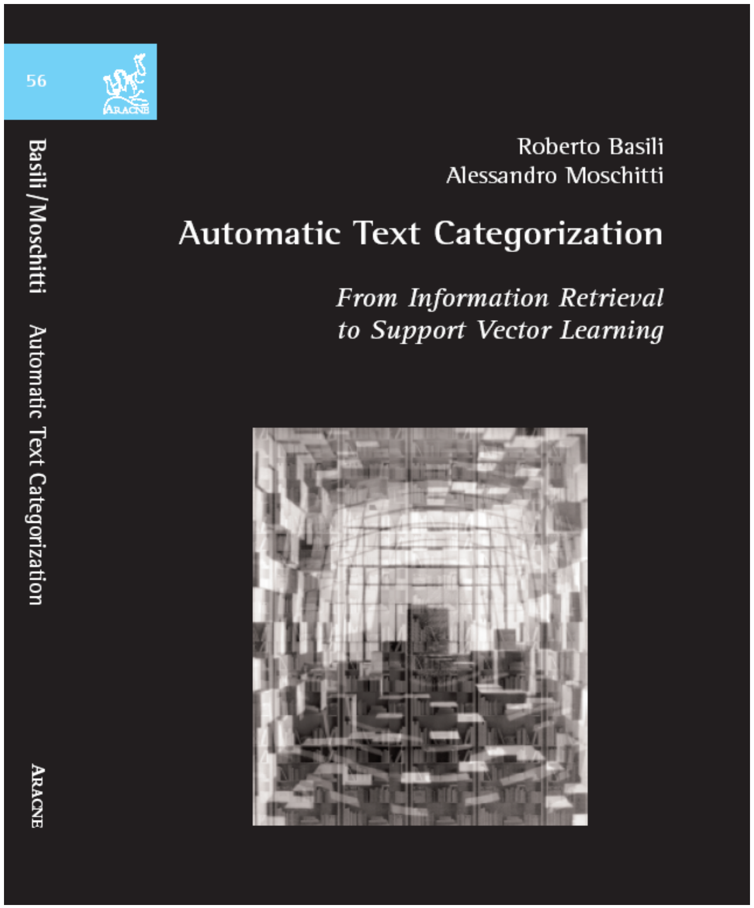
Future (on going work)

- Enriching text representation with semantic information, e.g., from Link Open Data or automatically generated by classifiers [Severyn, Nicosia, Moschitti, CIKM 2013]
- Deeper modeling of paragraphs: *shallow semantics and discourse structures* to design more compact and accurate representation of whole paragraphs
- Use of reverse kernel engineering to build efficient systems: [Pighin&Moschitti, CoNLL2009, EMNLP2009, CoNLL2010]
- Learning on large-scale data using combined uSVMs and linearized models [Severyn and Moschitti, IJCAI 2013]

Documentation

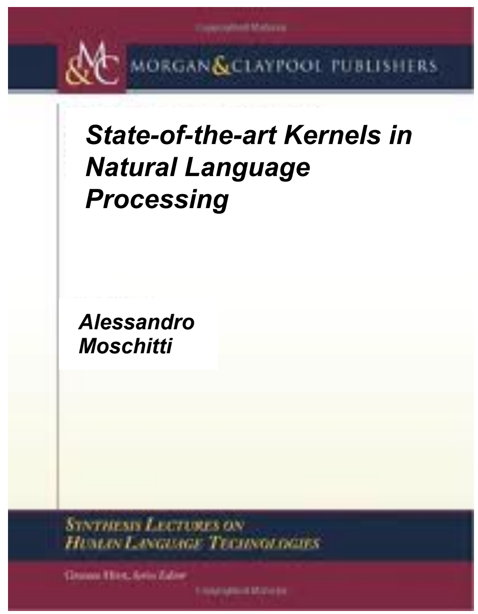
- Tutorial Webpage
  - http://disi.unitn.it/moschitti/SIGIR-tutorial.htm
  - Software
  - Data: Question Classification and Paragraph reranking
  - Updated slides
  - Papers
  - Books

An introductory book on SVMs, Kernel Methods and Text Categorization



Forthcoming 2014/15

- *State-of-the-art Kernels in Natural Language Processing*
  Author: Alessandro Moschitti
  Synthesis Lectures on Human Language Technologies
  Editor: Morgan & Claypool Publishers



Acknowledgments

This research has been partially supported by the European Project #288024 (FP7-ICT-2011-7, STREP):

LiMoSINe - Linguistically Motivated Semantic aggregation engines

Many thanks to the IBM Watson team, my co-authors and to Thorsten Joachims, Daniele Pighin, Aliaksei Severyn for lending me some of their slides

Thank you

References

- Qi Ju, Alessandro Moschitti: *Incremental Reranking for Hierarchical Text Classification*. ECIR 2013: 726-729
- Qi Ju, Alessandro Moschitti, Richard Johansson: *Learning to Rank from Structures in Hierarchical Text Classification*. ECIR 2013: 183-194
- Aliaksei Severyn and Massimo Nicosia and Alessandro Moschitti. *iKernels-Core: Tree Kernel Learning for Textual Similarity*. In *SEM 2013.
- Aliaksei Severyn and Alessandro Moschitti. *Fast Linearization of Tree Kernels over Large-Scale Data*. In IJCAI 2013.
- Aliaksei Severyn and Massimo Nicosia and Alessandro Moschitti. *Building Structures from Classifiers for Passage Reranking*. To appear in CIKM 2013.
- Barbara Plank and Alessandro Moschitti. *Embedding Semantic Similarity in Tree Kernels for Domain Adaptation of Relation Extraction*. To appear in ACL 2013.

References

- Aliaksei Severyn and Massimo Nicosia and Alessandro Moschitti. *Learning Adaptable Patterns for Passage Reranking*. In CoNLL 2013.
- Aliaksei Severyn and Massimo Nicosia and Alessandro Moschitti. *Learning Semantic Textual Similarity with Structural Representations.* In ACL 2013
- Aliaksei Severyn, Alessandro Moschitti: *Structural relationships for large-scale learning of answer re-ranking*. SIGIR 2012: 741-750
- Alessandra Giordani, Alessandro Moschitti, *Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked*. Proceedings of the 24rd International Conference on Computational Linguistics (Coling), Bombai, Mumbai, India, 2012.
- Alessandro Moschitti, Qi Ju, Richard Johansson: *Modeling Topic Dependencies in Hierarchical Text Categorization*. ACL 2012: 759-767

References

- Aliaksei Severyn, Alessandro Moschitti: *Fast Support Vector Machines for Convolution Tree Kernels*. Data Mining Knowledge Discovery 25: 325-357, 2012.
- Siddharth Patwardhan, Branimir Boguraev, Apoorv Agarwal, Alessandro Moschitti, Jennifer Chu-Carroll, *Labeling by Landscaping: Classifying Tokens in Context by Pruning and Decorating Trees*. In Proceedings of the Conference on Information and Knowledge Management, Maui Hawaii, CIKM 2012.
- Danilo Croce, Alessandro Moschitti, Roberto Basili, Martha Palmer: *Verb Classification using Distributional Similarity in Syntactic and Semantic Structures*. ACL 2012: 263-272
- Vien Nguyen and Alessandro Moschitti, *Structural Reranking Models for Named Entity Recognition*, Special issue on Natural Language Processing in the Web Era, Intelligenza Artificiale, IOS Press, Volume 6.2, 2012.

References

- M. Dinarelli, A. Moschitti, and G. Riccardi. *Discriminative Reranking for Spoken Language Understanding*. IEEE Transaction on Audio, Speech and Language Processing, 2012.
- Alessandro Moschitti and Silvia Quarteroni, *Linguistic Kernels for Answer Re-ranking in Question Answering Systems,* Information and Processing Management: an International journal, ELSEVIER, 2011
- Danilo Croce, Alessandro Moschitti, and Roberto Basili. *Structured lexical similarity via convolution kernels on dependency trees*. In Proceedings of EMNLP, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.
- Aliaksei Severyn and Alessandro Moschitti. *Fast support vector machines for structural kernels*. In ECML-PKDD, 2011, Greece, 2011. Best Machine Learning Student Paper Award

References

- Truc Vien T. Nguyen and Alessandro Moschitti. *Joint distant and direct supervision for relation extraction*. In Proceedings of the The 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, November 2011, Association for Computational Linguistics.
- Alessandro Moschitti, Jennifer Chu-carroll, Siddharth Patwardhan, James Fan, and Giuseppe Riccardi. *Using syntactic and semantic structural kernels for classifying definition questions in Jeopardy!* In Proceedings of EMNLP, pages 712–724, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.
- Truc Vien T. Nguyen and Alessandro Moschitti. *End-to-end relation extraction using distant supervision from external semantic repositories*. In Proceedings of HLT-ACL, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
- *Large-Scale Support Vector Learning with Structural Kernels*, In Proceedings of the 21th European Conference on Machine Learning (ECML-PKDD2010), Barcelona, Spain, 2010.

References

- Alessandro Moschitti' handouts http://disi.unitn.eu/moschitti/teaching.html
- 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.
- 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.
- 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.
- 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.

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.