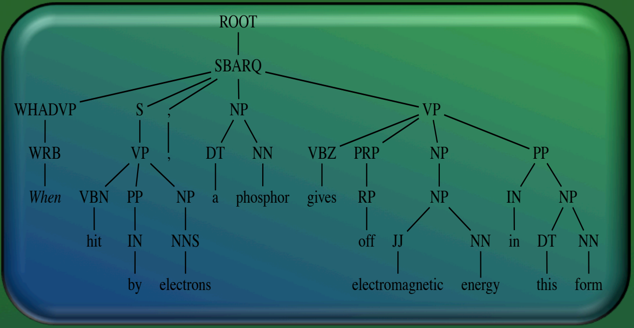
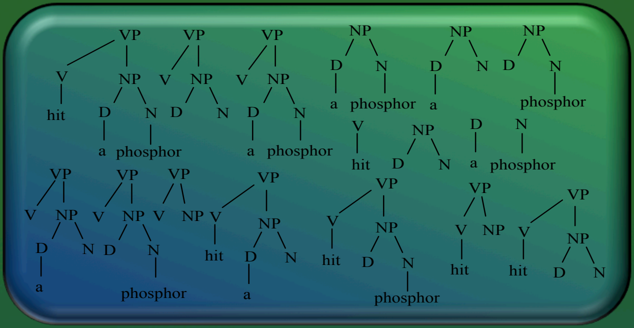
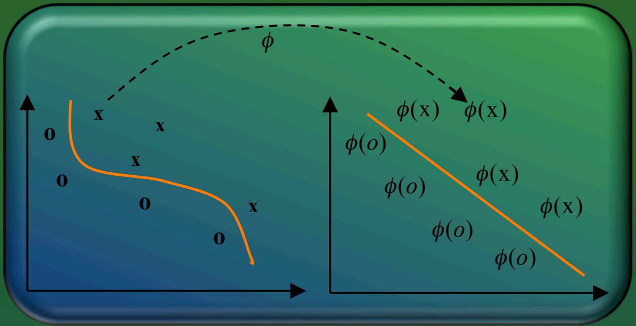






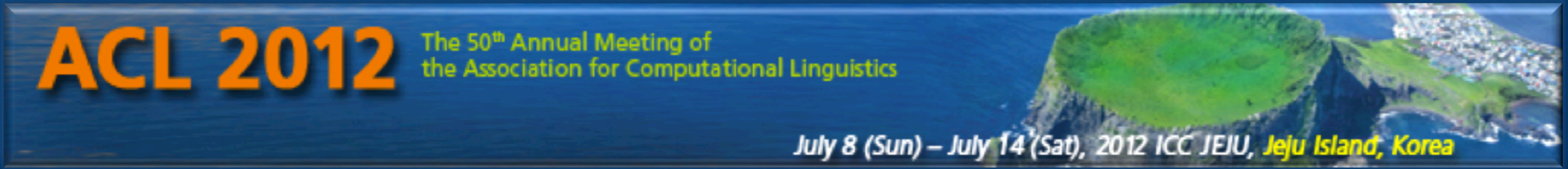
State-of-the-Art Kernels in Natural Language Processing

Alessandro Moschitti

Dept. of Computer Science and Engineering

University of Trento

moschitti@disi.unitn.it



Tutorial Schedule

- 9:00 – 10:30 First Part
- 10:30 – 11:00 Break
- 11:00 – 12:30 Second Part

Outline: Part I – Kernel Machines

- Outline and Motivation (10 min)
- Kernel Machines (25 min)
  - 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 (30 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
- Simple Kernel Applications (25 min)
  - Question Classification in TREC
  - Cue Classification in Jeopardy!
  - Question and Answer Classification
- Break (30 min)

Outline: Part II – Applications with Simple Kernels

- Practical Exercise with SVM-Light-TK (15 min)
  - Question Classification with dependency and constituency trees
- NLP applications with simple kernels (15 min)
  - Semantic Role Labeling (SRL): FrameNet and PropBank
  - Relation Extraction: ACE
  - Coreference Resolution

Outline: Part II – Joint Kernel Models

- Reranking for (15 min)
  - Preference kernel framework
  - Concept Segmentation and Classification of speech
  - Named Entity Recognition
  - Predicate Argument Structure Extraction
- Relational Kernels (15 min)
  - Recognizing Textual Entailment
  - Answer Reranking

Outline: Part II – Advanced Topics

- Fast learning and classification approaches (10 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 Research (5 min)

Motivation (1)

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

Motivation (2)

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

Motivation (3)

- 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. Relation Extraction
- Fast prototyping and adaptation for new domains and applications
- The major contribution of kernels is to make easier system modeling.

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
  - automatic feature engineering
  - explicit representation: are more meaningful when inspected

Why and when using kernels?

- Using them is very simple: much simpler than feature vector
- They are like any other machine learning approach simply better than feature vector
- 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

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 acquisition stocks sell market

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

buy company stocks sell

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

Linear Classifier

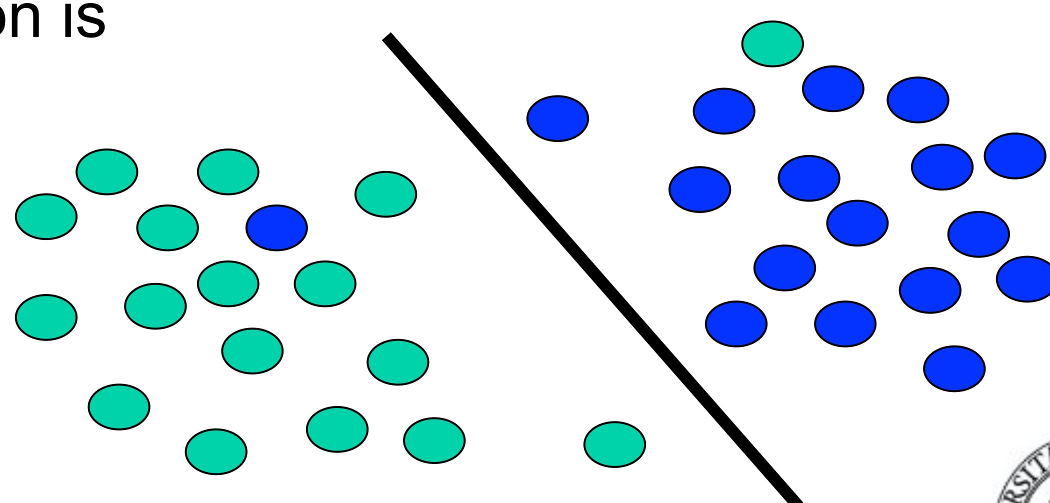
- The equation of a hyperplane is

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

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

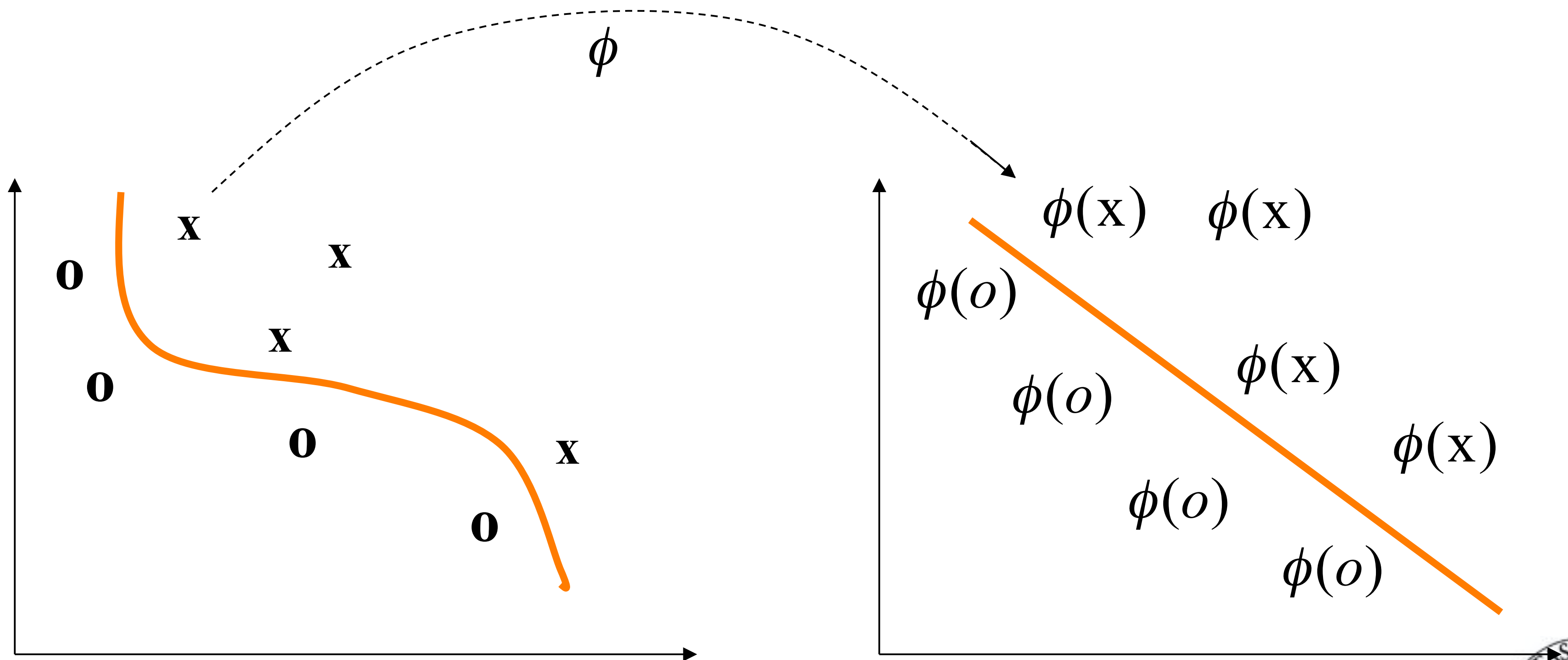
$$h(x) = \text{sign}(f(x))$$

Basically $\vec{d} \cdot \vec{C} > th$



The main idea of Kernel Functions

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



A kernel-based Machine: Perceptron training

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

do

for i = 1 to ℓ

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

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

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

k = k + 1

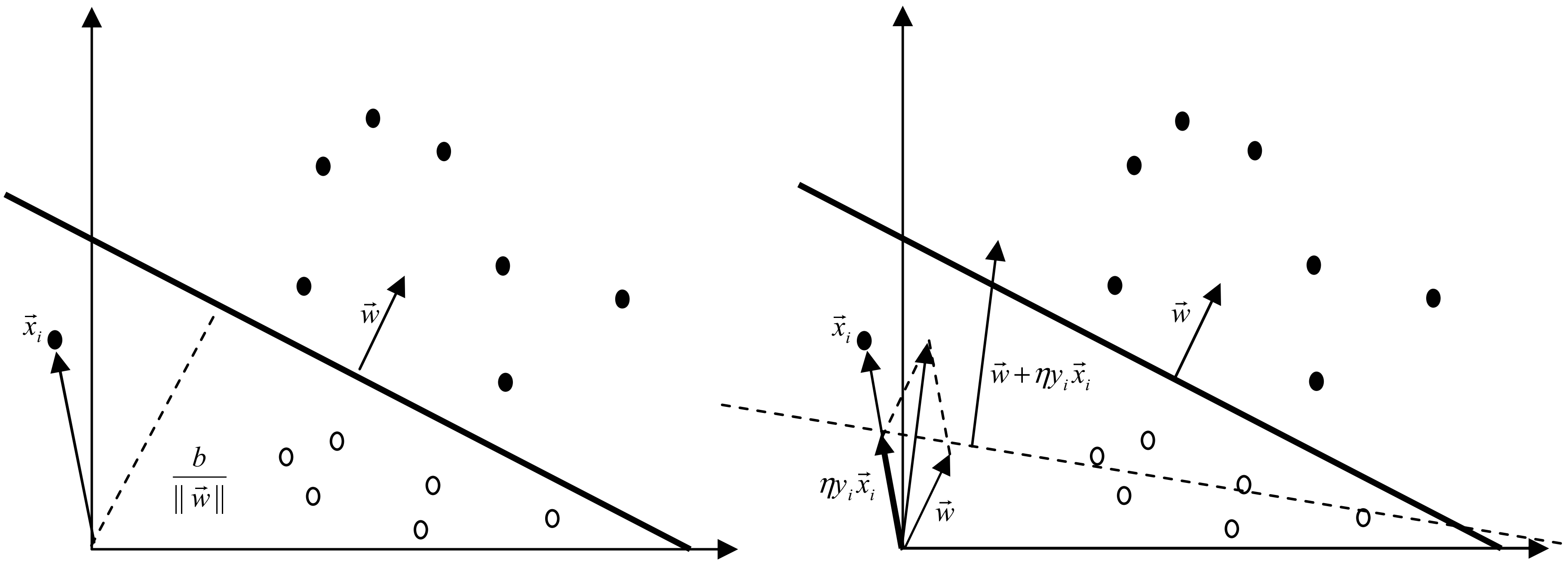
endif

endfor

while an error is found

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

Graphic interpretation of the Perceptron



Dual Representation for Classification

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

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

- 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

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

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

Dual Perceptron algorithm and Kernel functions

- We can rewrite the classification function as

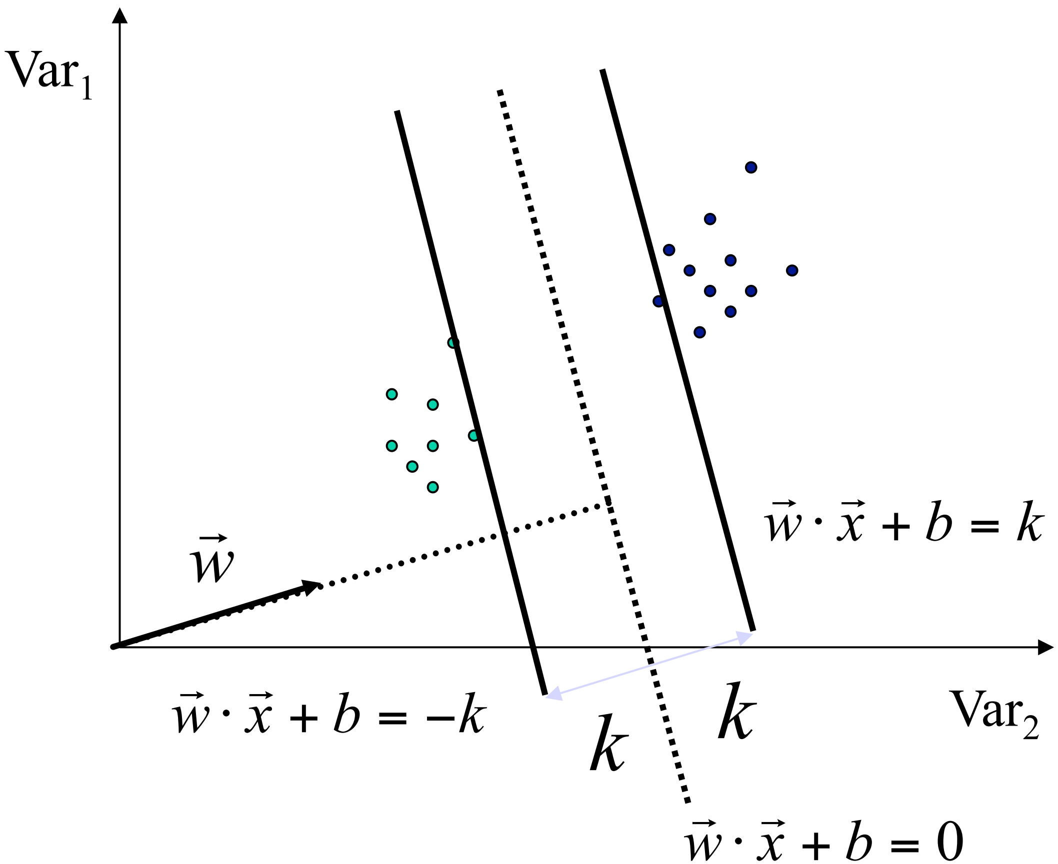
$$h(x) = \text{sgn}(\vec{w}_\phi \cdot \phi(\vec{x}) + b_\phi) = \text{sgn}(\sum_{j=1..\ell} \alpha_j y_j \phi(\vec{x}_j) \cdot \phi(\vec{x}) + b_\phi) =$$

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

- As well as the updating function

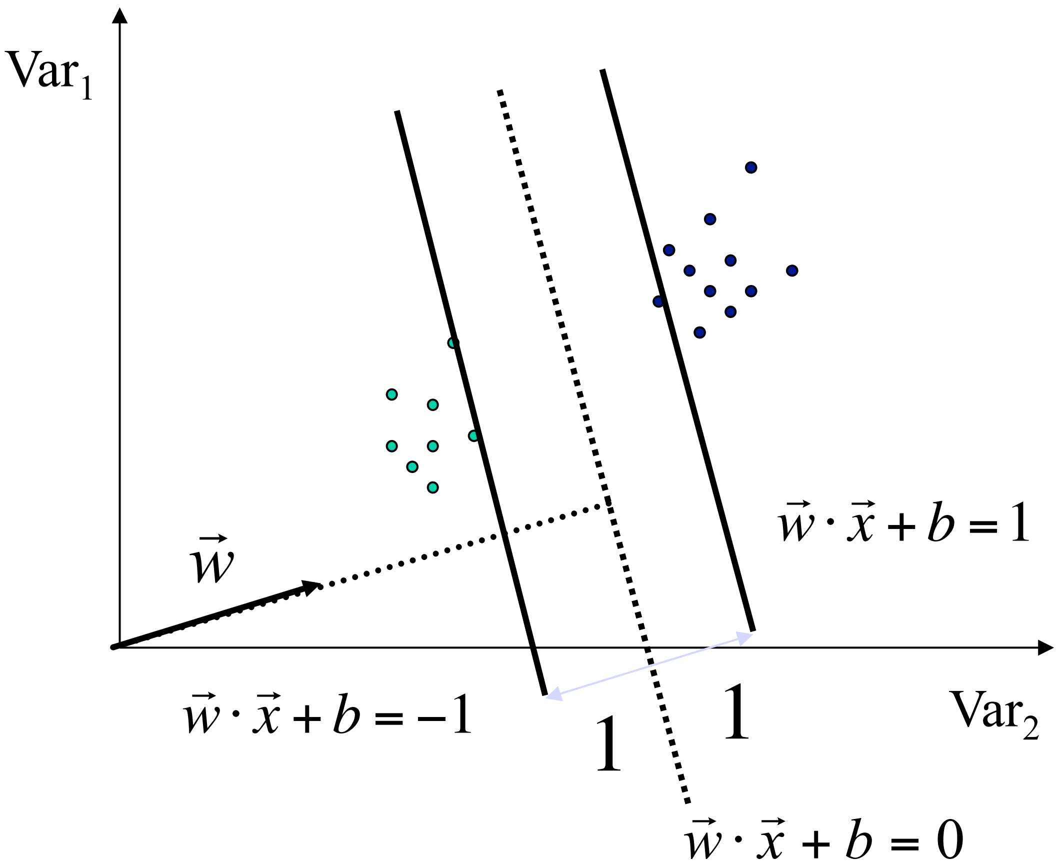
$$\text{if } y_i \left(\sum_{j=1..\ell} \alpha_j y_j k(\vec{x}_j, \vec{x}_i) + b_\phi \right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

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

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

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

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

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

$$\mathit{maximize} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x_i} \cdot \vec{x_j}$$

$$\mathit{subject\ \ to} \quad \alpha_i \geq 0, \quad i = 1, .., m$$

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

Soft Margin optimization problem

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

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

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

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:

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

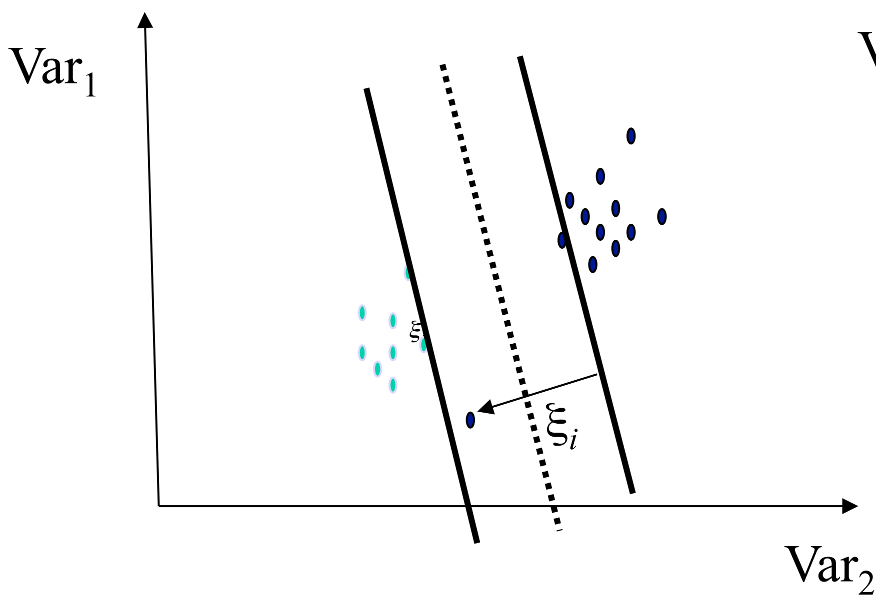
Soft Margin Support Vector Machines

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

$$\xi_i \geq 0$$

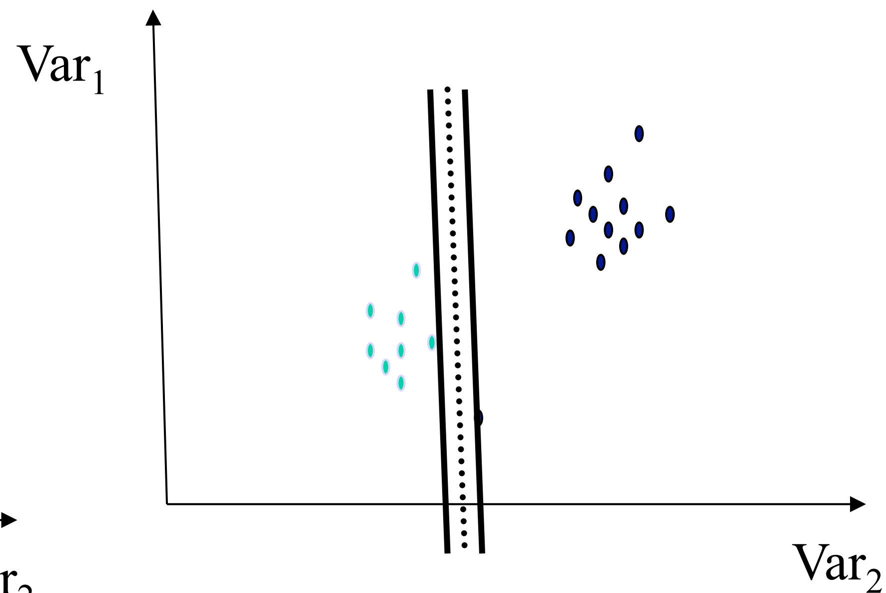
- The algorithm tries to keep ξ_i low and maximize the margin
- NB: the number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized
- If $C \to \infty$, the solution tends to the one of the *hard-margin* algorithm
  - If C increases the number of error 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

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

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

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

Valid Kernels

Def. B.11 *Eigen Values*
Given a matrix $\boldsymbol{A} \in \mathbb{R}^m \times \mathbb{R}^n$, *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 $\Rightarrow \exists\, \mathrm{V}$: $\mathrm{K} = \mathrm{V}\Lambda\mathrm{V}'$ for Takagi factorization of a complex-symmetric matrix, where:
  - Λ is the diagonal matrix of the eigenvalues λ_t of K
  - $\vec{\mathrm{v}}_t = \left(v_{ti}\right)_{i=1}^{n}$ are the eigenvectors, i.e. the columns of V
- Let us assume lambda values non-negative

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

Mercer's Theorem (sufficient conditions)

- Therefore

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

- which implies that K is a kernel function

Mercer's Theorem (necessary conditions)

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

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

- has the following norm:

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

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

this contradicts the geometry of the space.

Is it a valid kernel?

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

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

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

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
(Feature Extraction Functions)

Basic Kernels for unstructured data

- Linear Kernel
- Polynomial Kernel
- Lexical Kernel
- String Kernel
- Tree Kernels: Subtree, Syntactic, Partial Tree Kernels (PTK), and 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 acquisition stocks sell market

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

buy company stocks sell

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

Feature Conjunction (polynomial Kernel)

- The initial vectors are mapped in a higher space

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

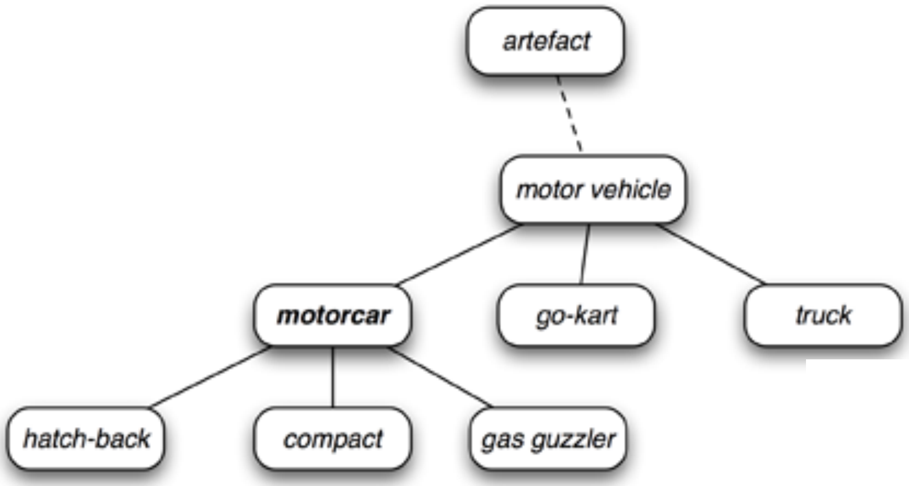
- More expressive, as (x_1x_2) encodes

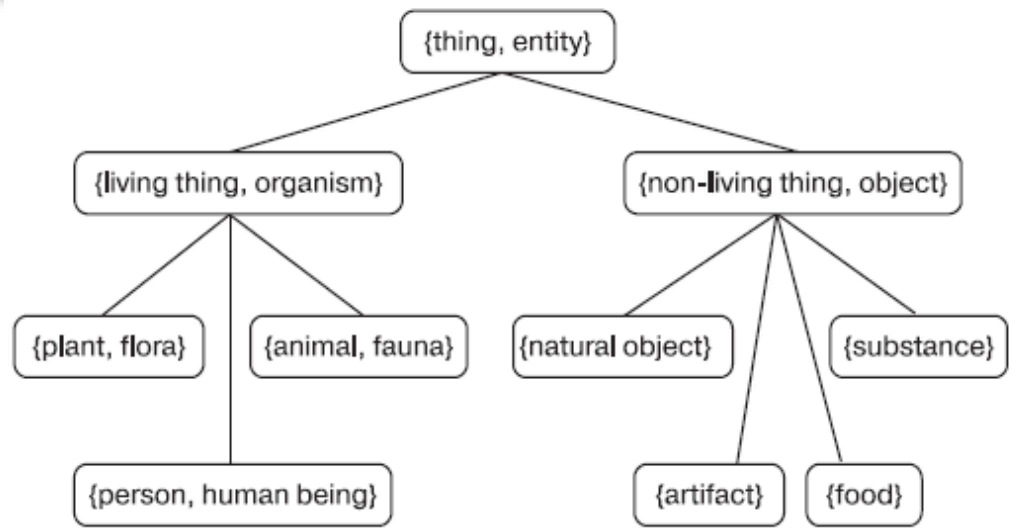
  `Stock+Market` vs. **`Downtown+Market`** features

- We can smartly compute the scalar product as

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

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:

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

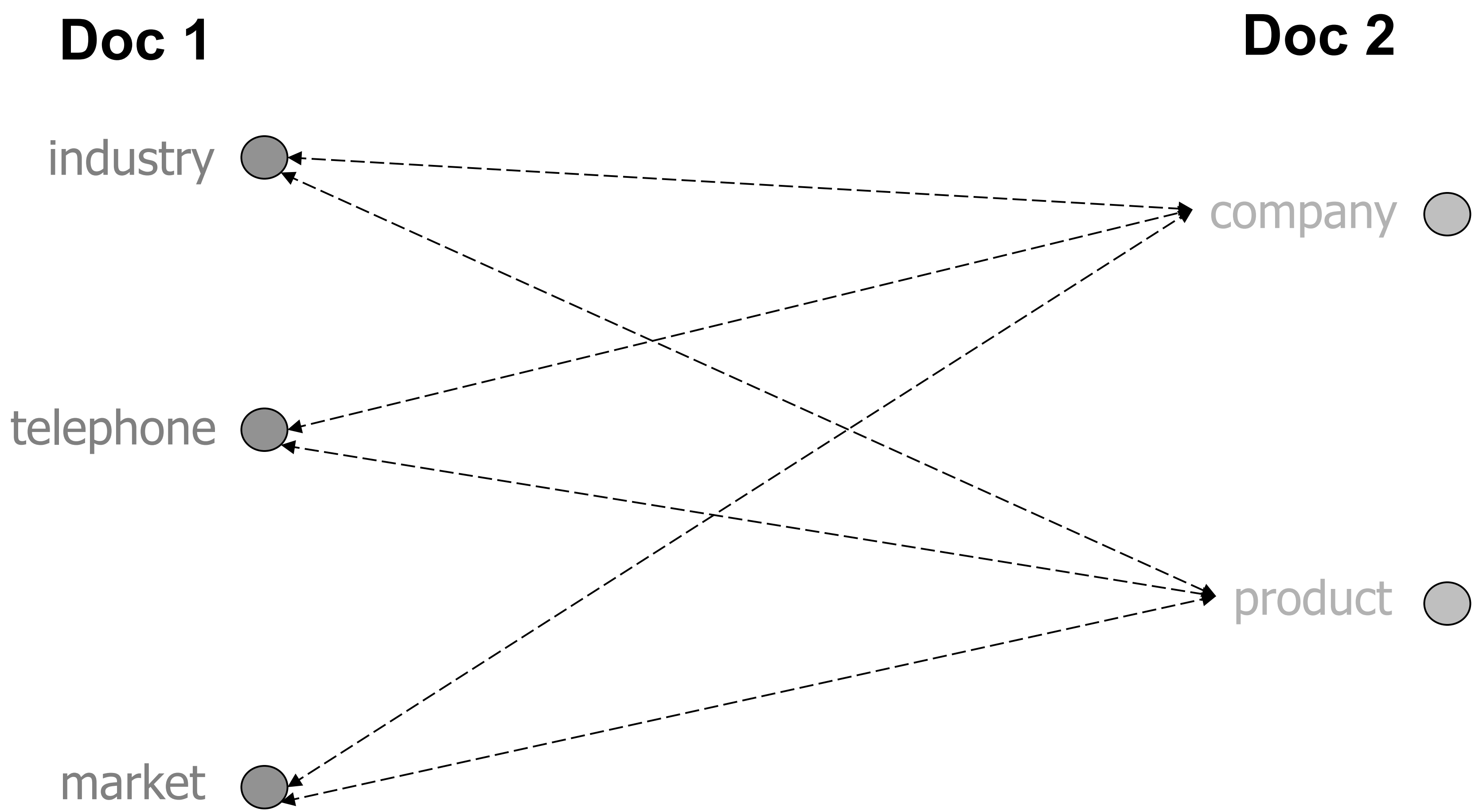
Resnik:

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

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 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

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

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

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

Efficient Evaluation: Intuition

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

Efficient Evaluation

Given two sequences s_1a and s_2b, we define:

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

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

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

D_p satisfies the recursive relation:

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

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

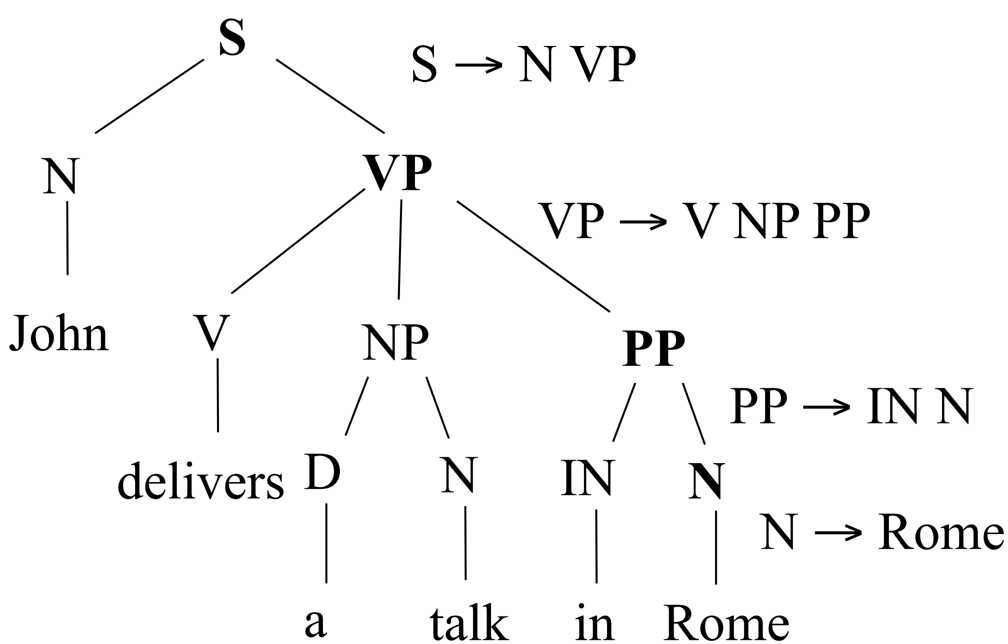
$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r])$$

Tree kernels

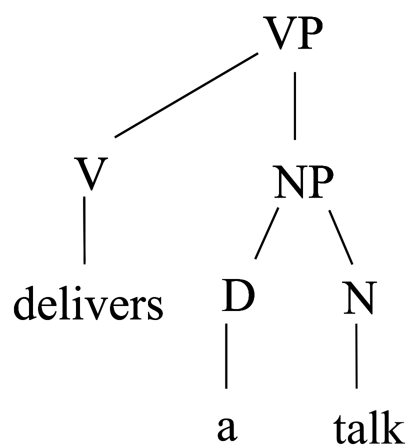
- Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK
- Efficient computation

Example of a parse tree

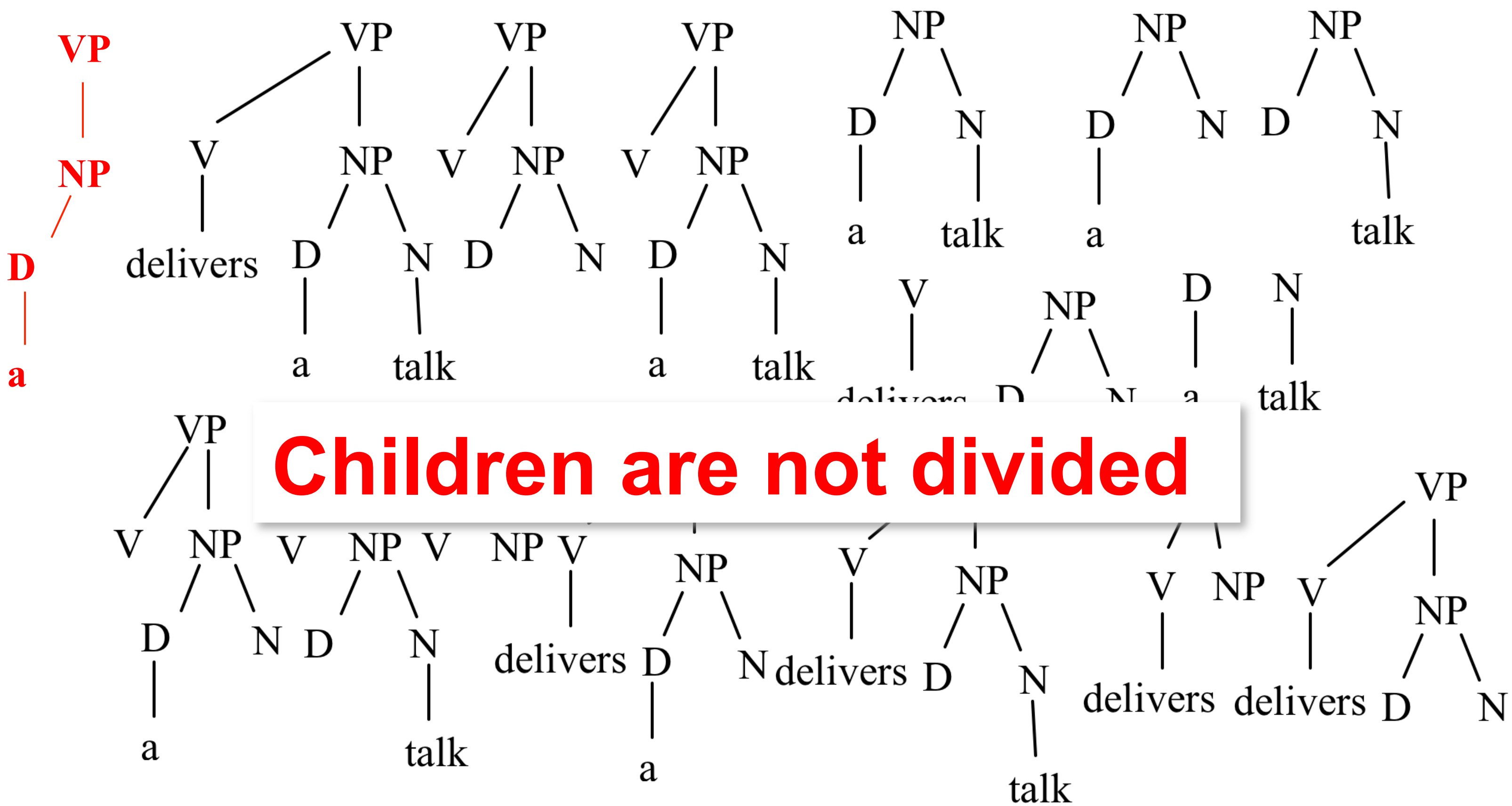
- "John delivers a talk in Rome"



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

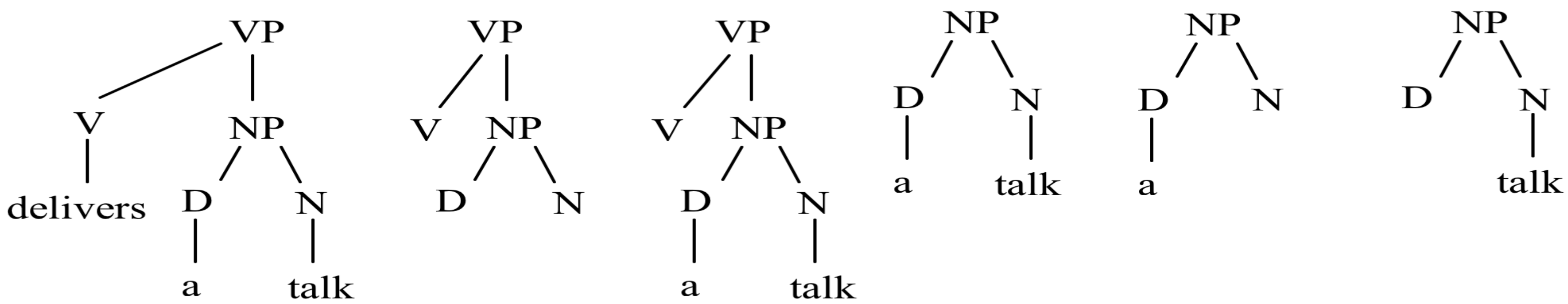


The overall fragment set

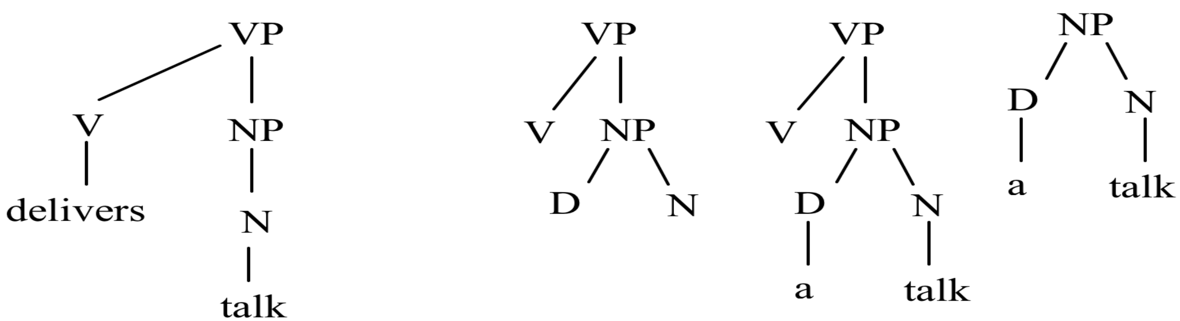


Explicit kernel space

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



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



- $\vec{x} \cdot \vec{z}$ 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(n²):

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

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

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

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

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

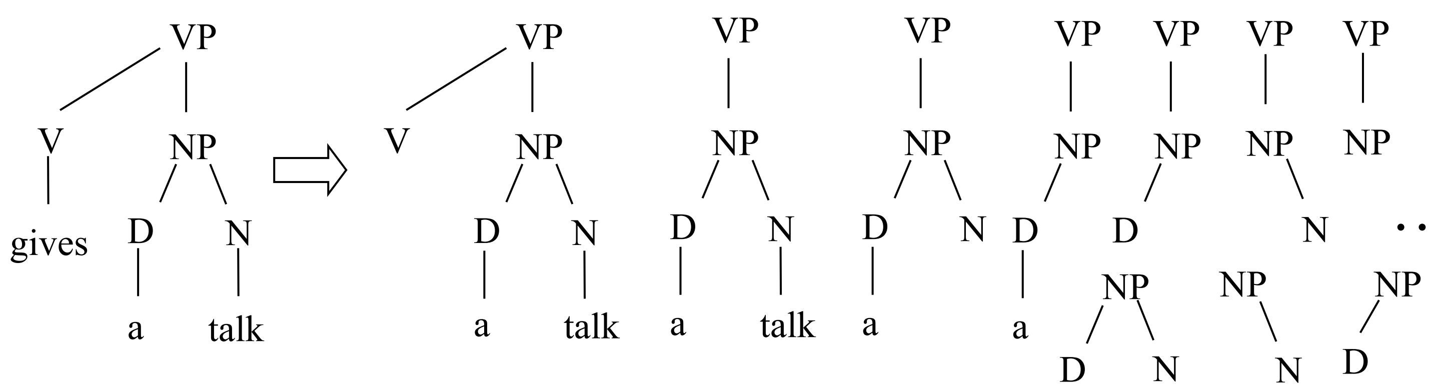
Observations

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

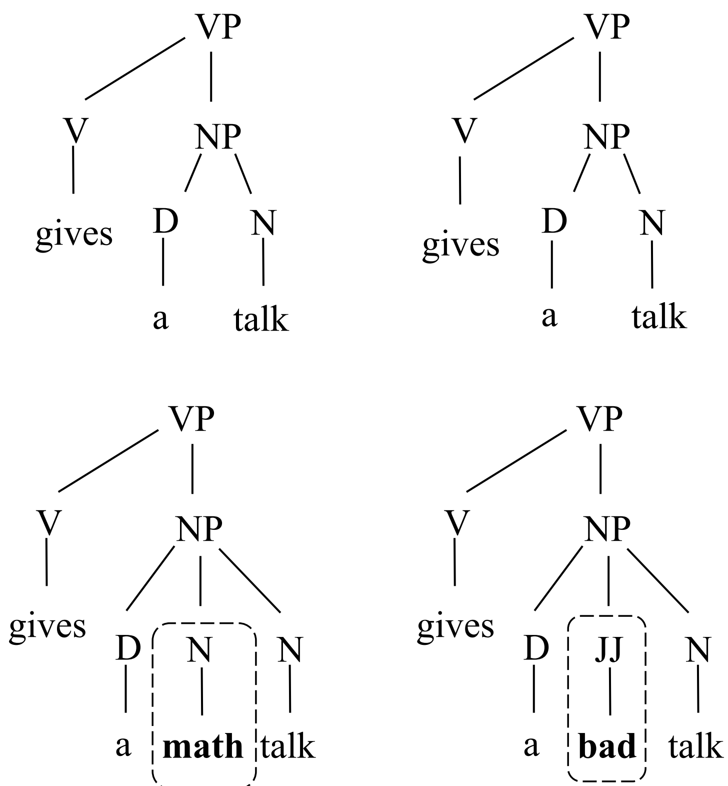
  $|T_x| + |T_z|$ *running time* [Moschitti, EACL 2006]
- If T_x and T_z are generated by only one production rule $\Rightarrow$ $O(|T_x| \times |T_z|)$… ***Very Unlikely!!!!***

Labeled Ordered Tree Kernel

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



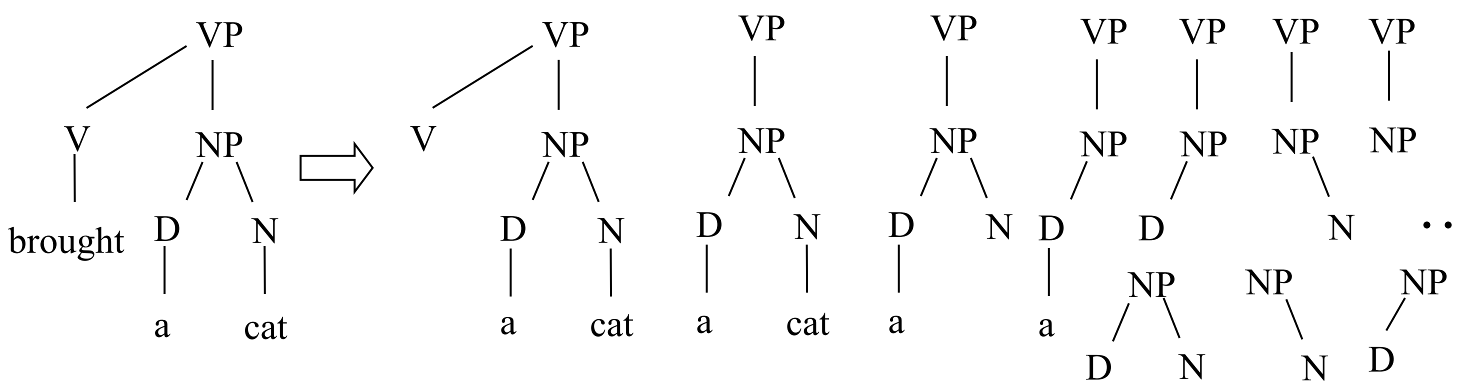
Weighting Problems



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

Partial 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

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

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

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

Efficient Evaluation (2)

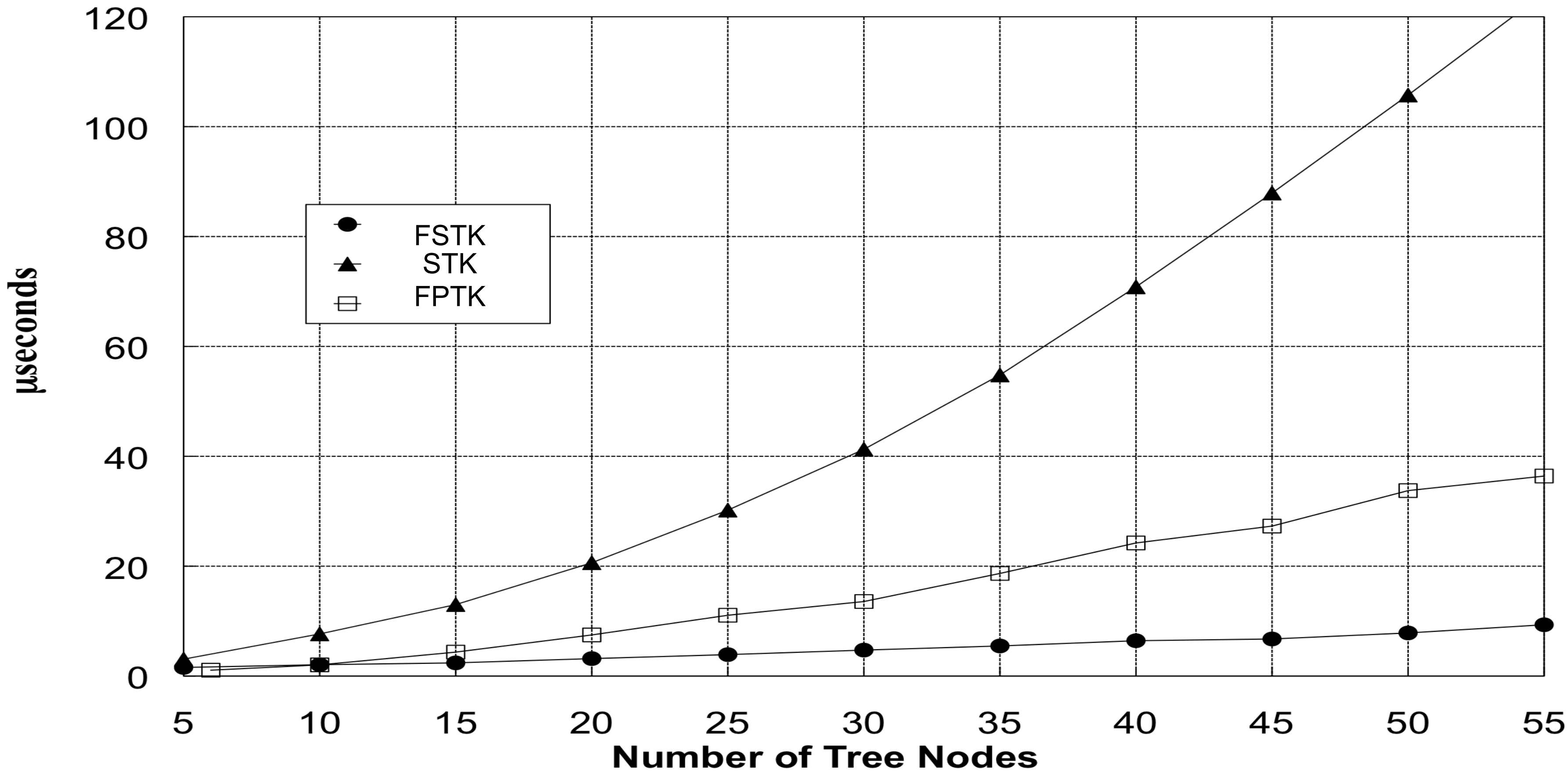
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

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

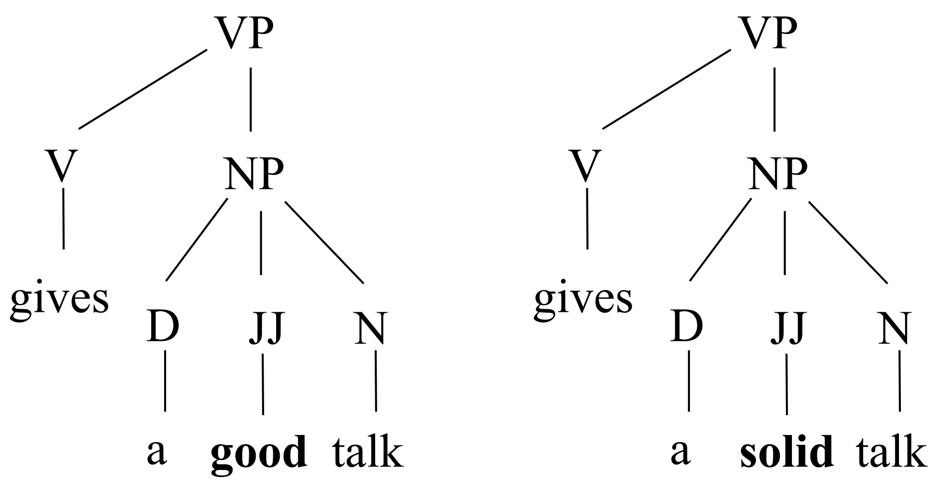
- The complexity of finding the subsequences is $O(p|s_1||s_2|)$
- Therefore the overall complexity is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ where ρ is the maximum branching factor ($p = \rho$)

Running Time of Tree Kernel Functions



- STK vs. Fast STK (FSTK) and Fast PTK (FPTK)

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



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

Equations of SSTK

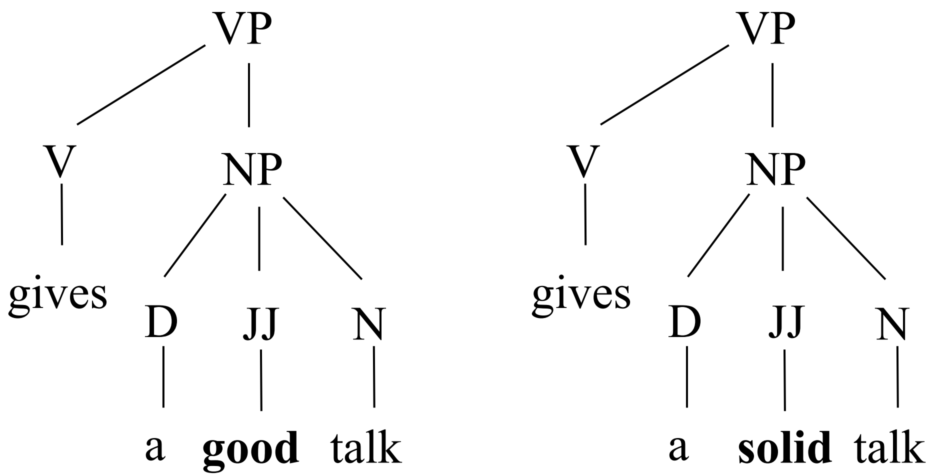
Definition 4 (Tree Fragment Similarity Kernel). *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[6]*:*

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

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

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

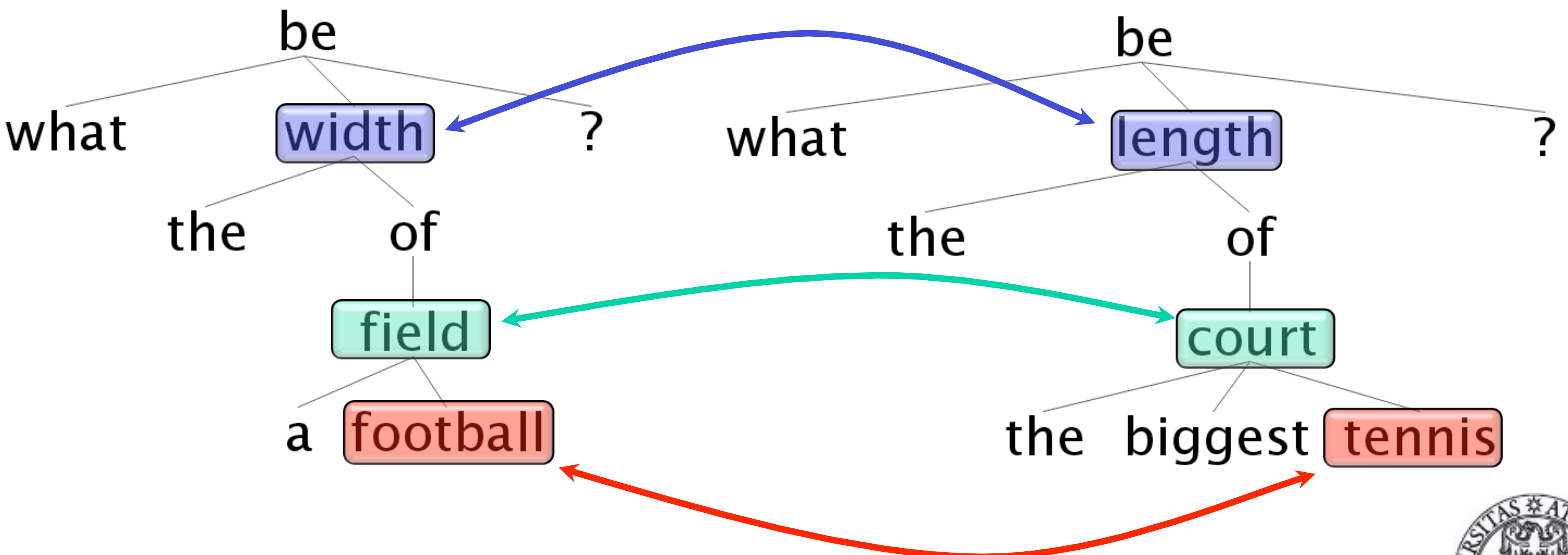
2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

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 $\Delta_\sigma(n_1, n_2) = \boxed{\mu\lambda\sigma(n_1, n_2)}$

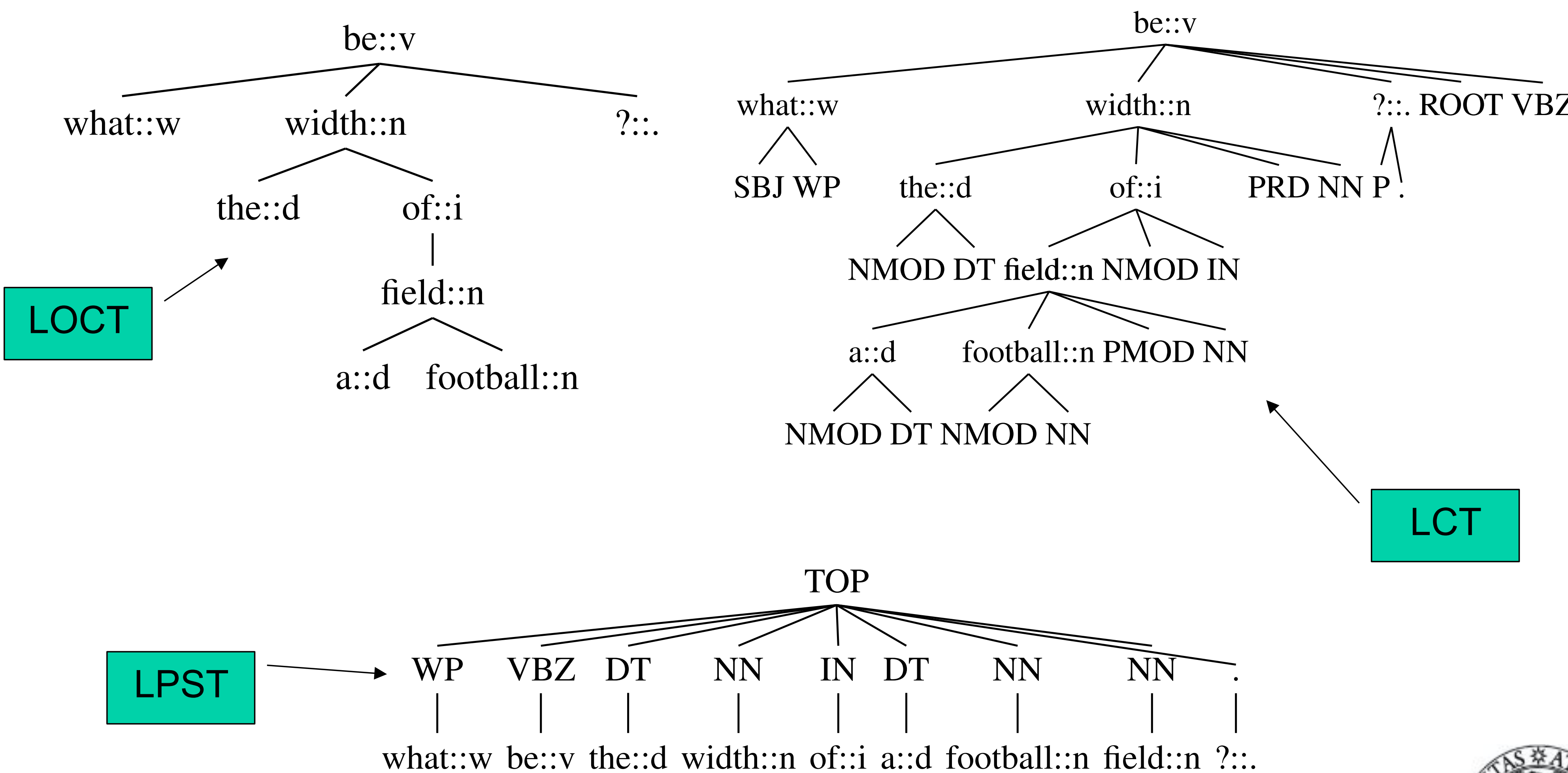
else

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

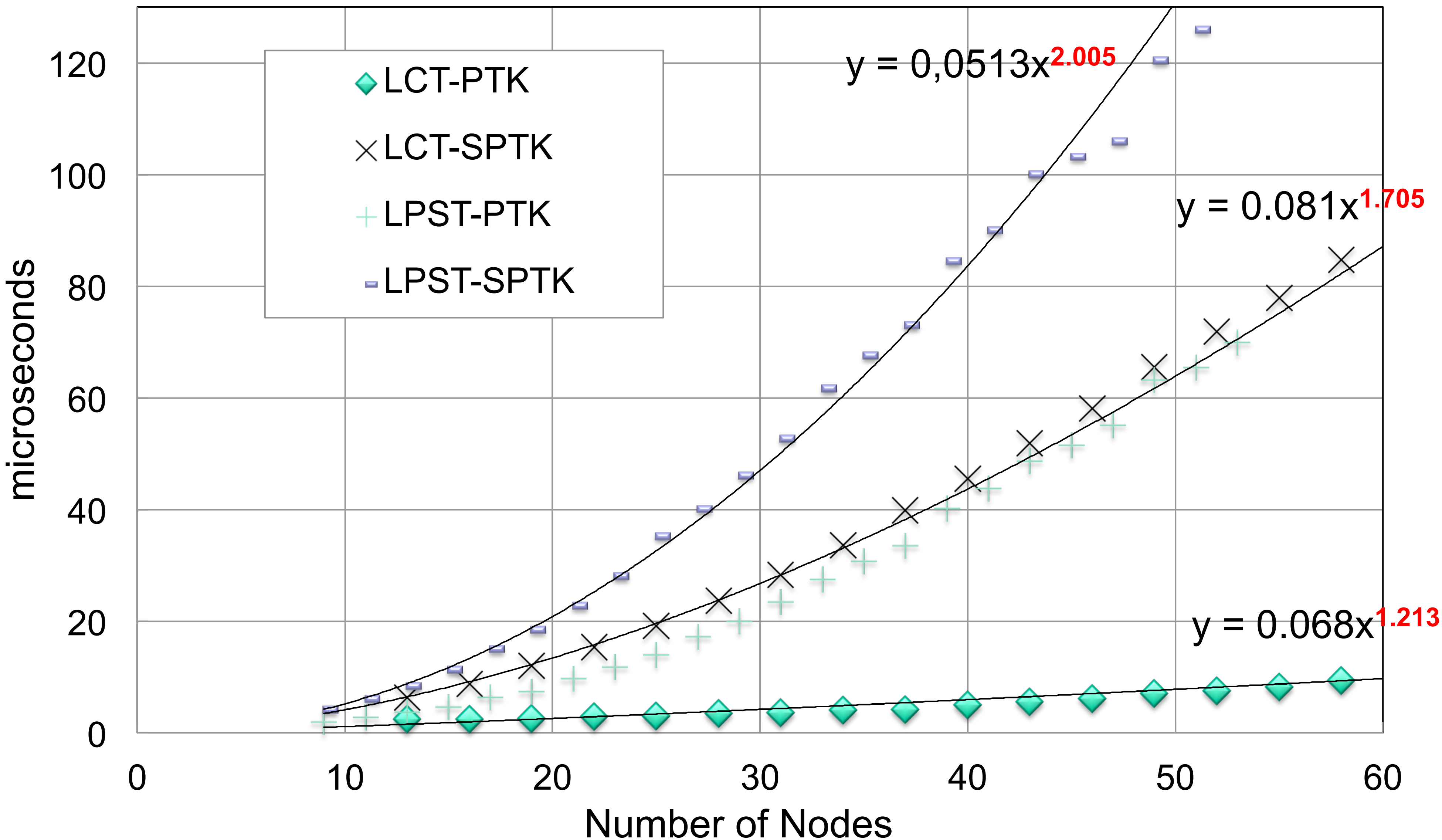
Lexical Similarity

PTK

Different versions of Computational Dependency Trees for PTK/SPTK

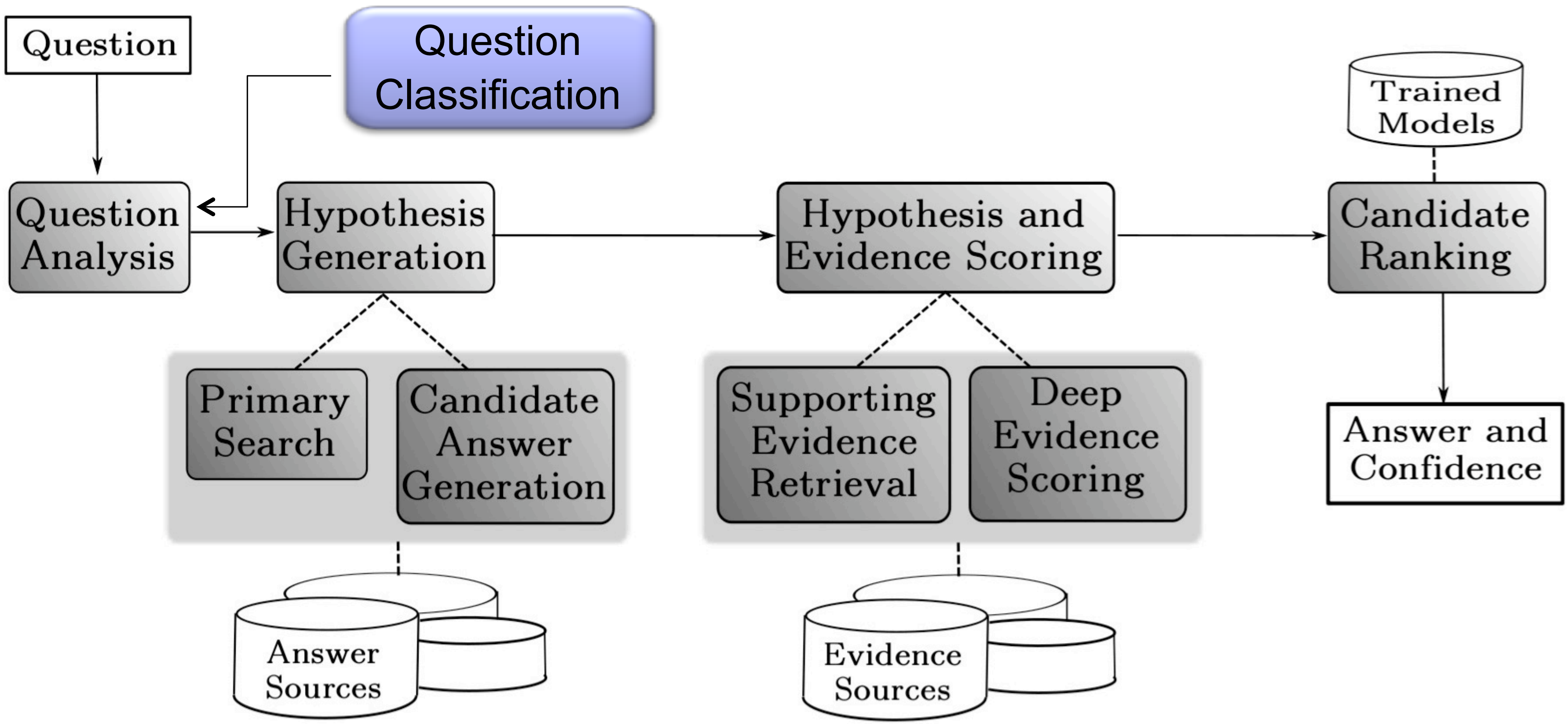


Tree Kernel Efficiency



Simple Kernel Applications

A QA Pipeline: Watson Overview



Question Classification

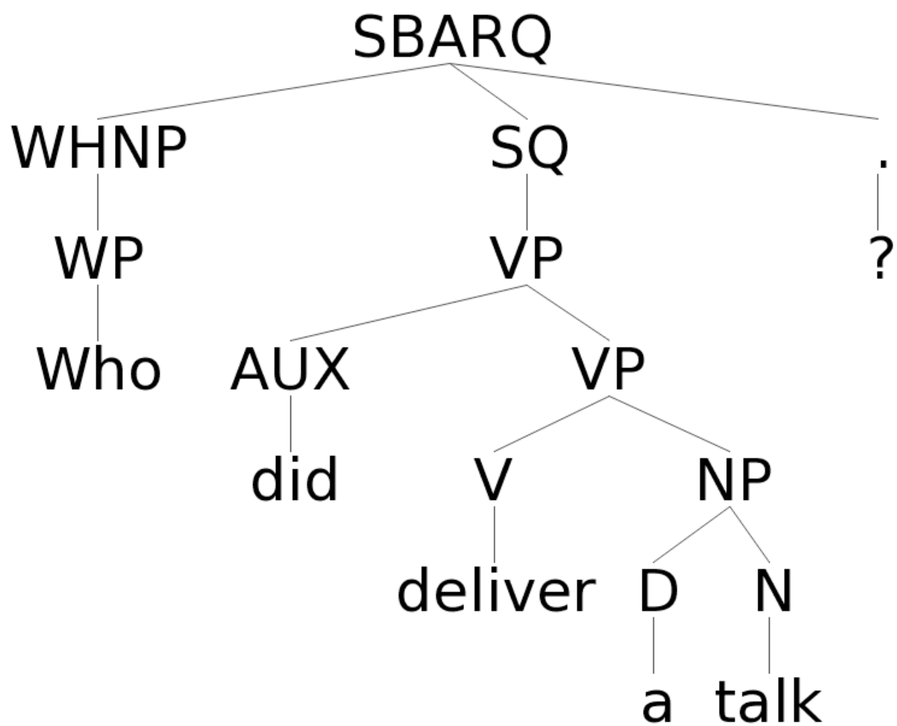
- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: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

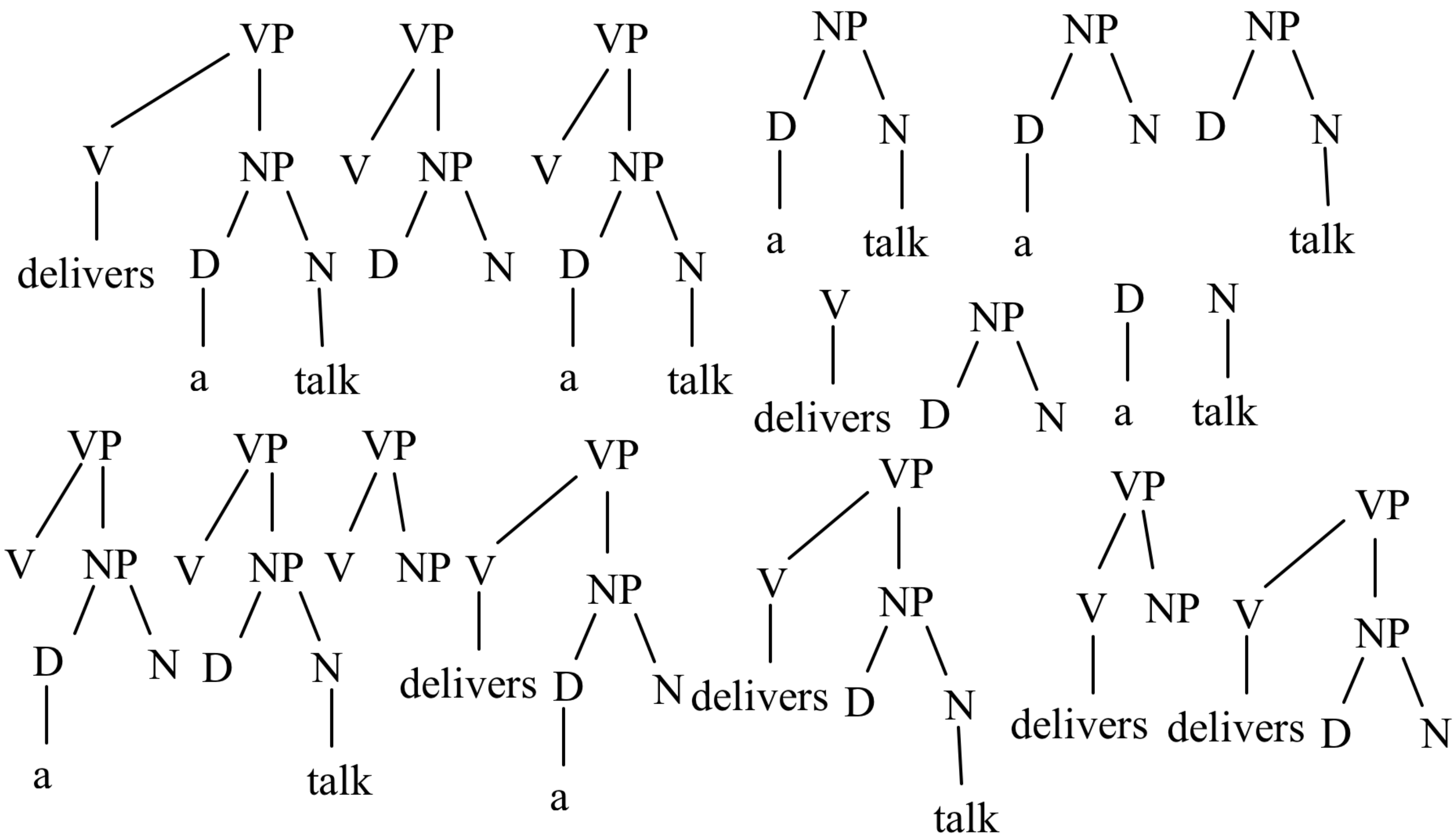
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Using the whole question parse trees
  - Constituent parsing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

“What is an offer of direct stock purchase plan ?”

Syntactic Parse Trees (PT)

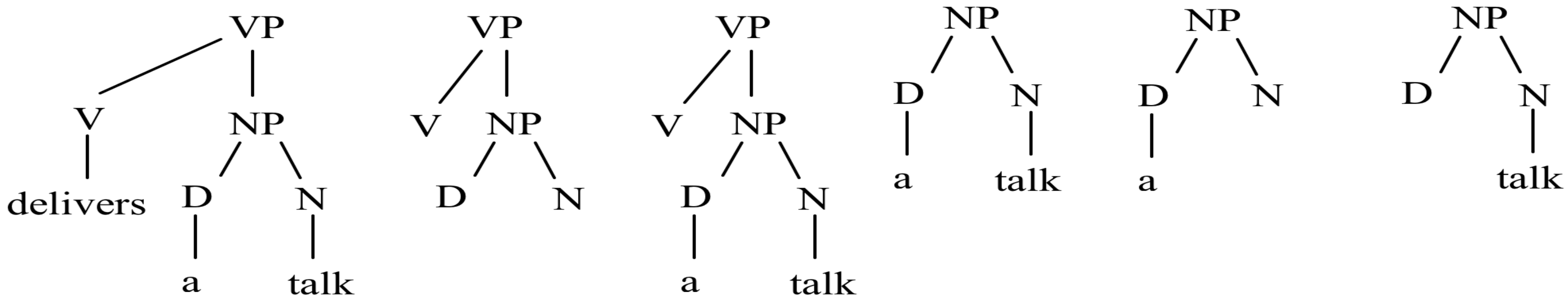


Some fragments

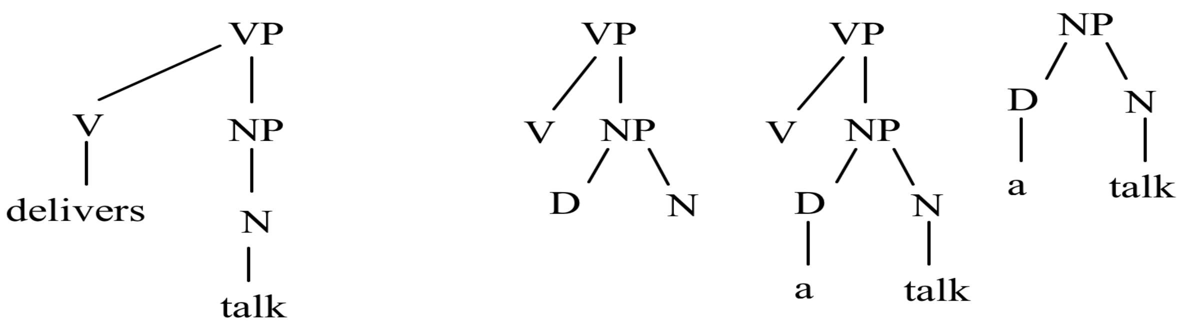


Explicit kernel space

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



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

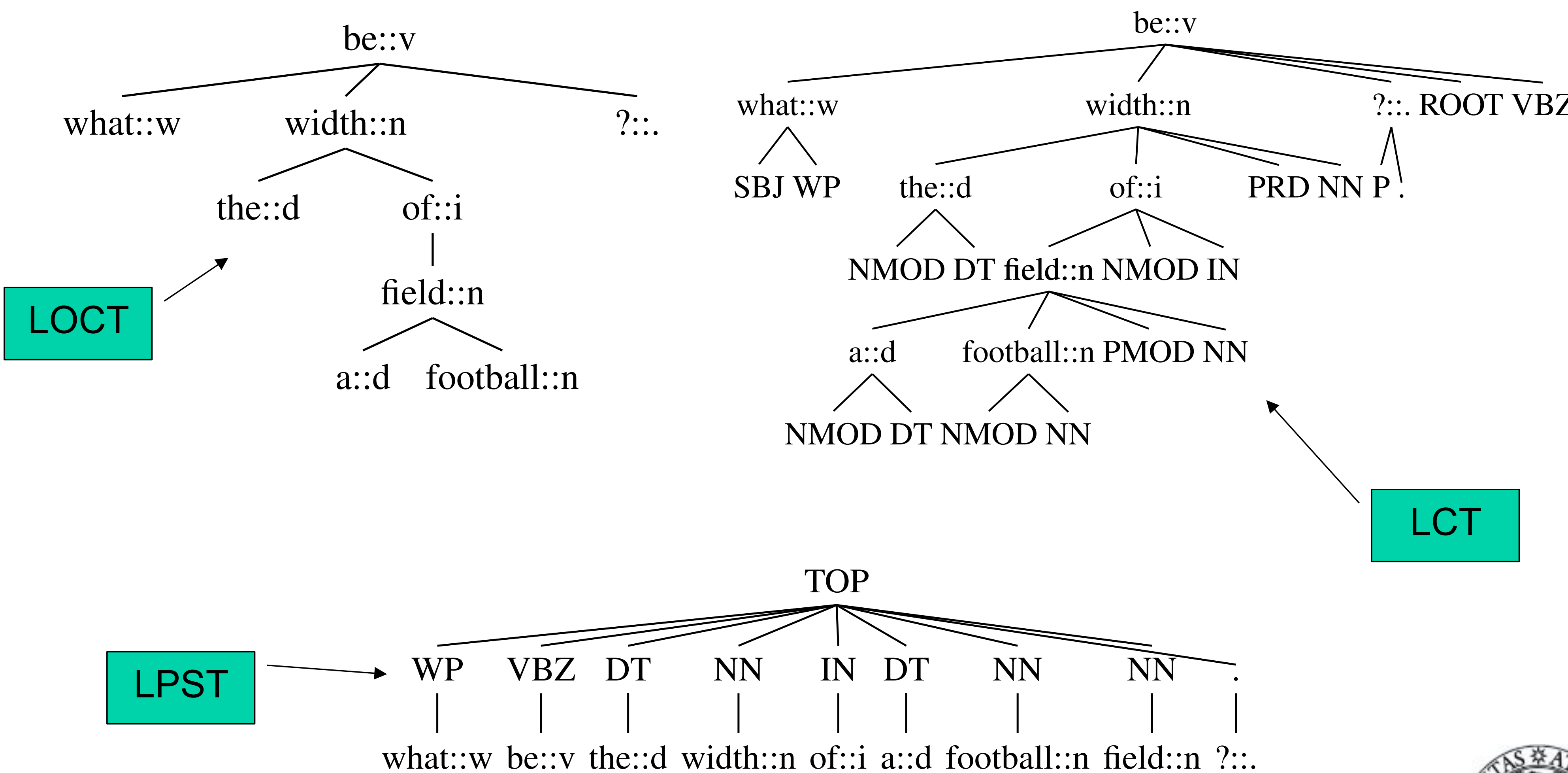


- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

Question Classification with SSTK
[Blohedorn&Moschitti, CIKM2007]

	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

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%	


JEOPARDY!

Classification in Definition vs not Definition in Jeopardy!

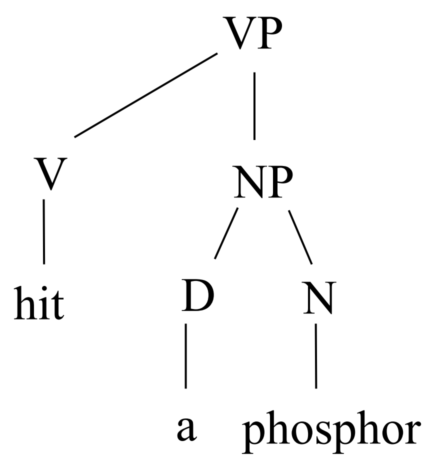
- **Definition:** *Usually, to do this is to lose a game without playing it*

  (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

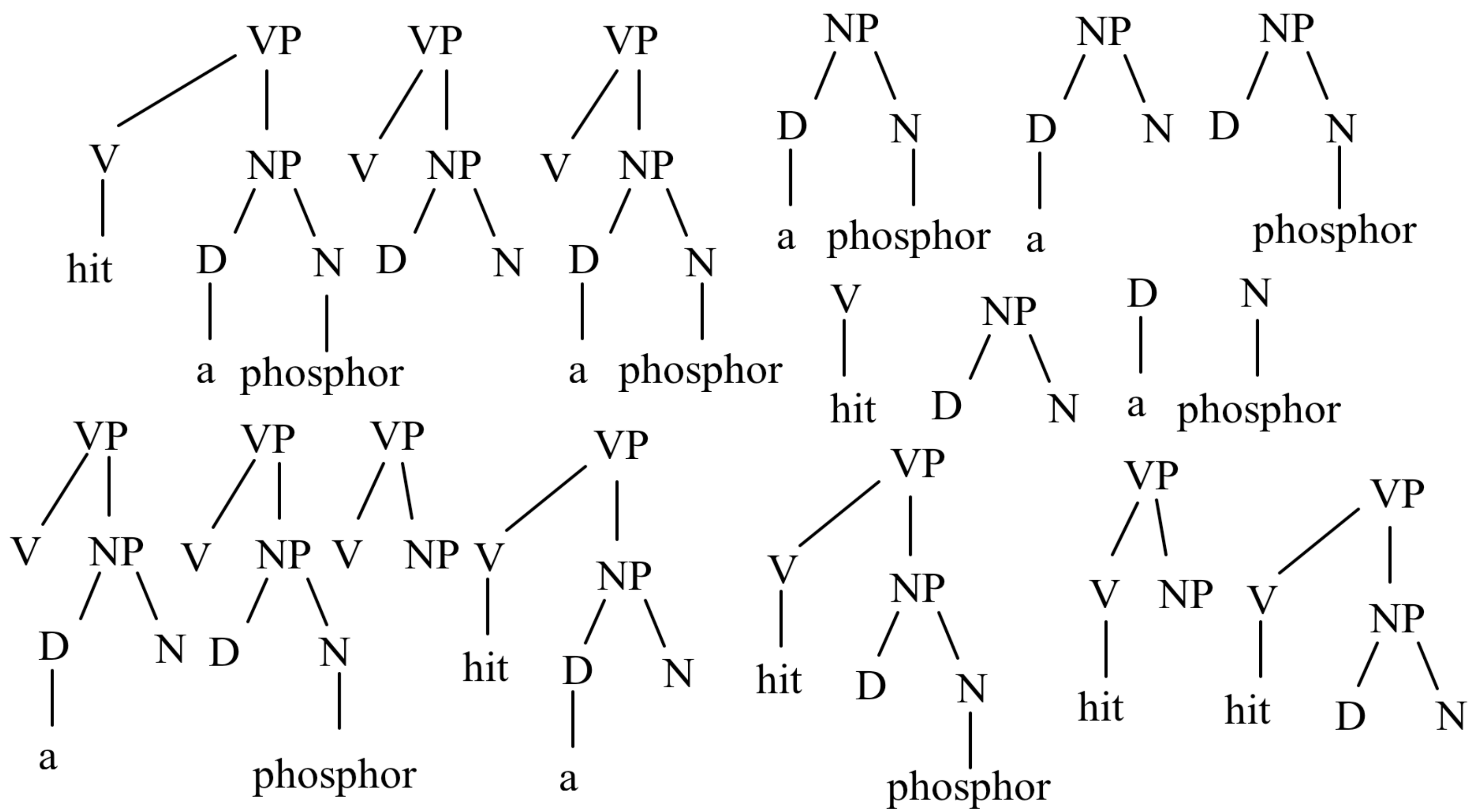
Automatic Learning of a Question Classifier

- 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**:
  - $K_1(\langle q_1,q_2 \rangle)+K_2(\langle q_1,q_2 \rangle)+\ldots+K_n(\langle q_1,q_2 \rangle)$
- Tree kernels measure similarity between trees

Syntactic Tree Kernel (STK) (Collins and Duffy 2002)

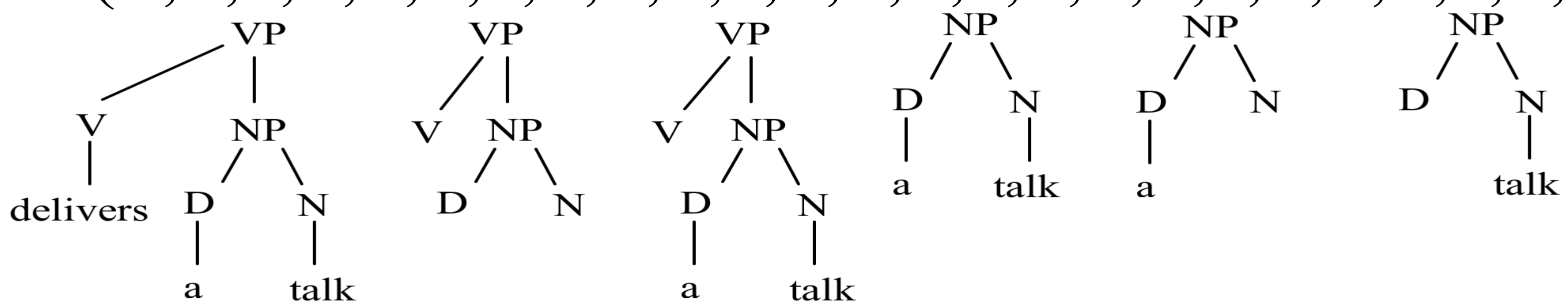


Syntactic Tree Kernel (STK) (Collins and Duffy 2002)

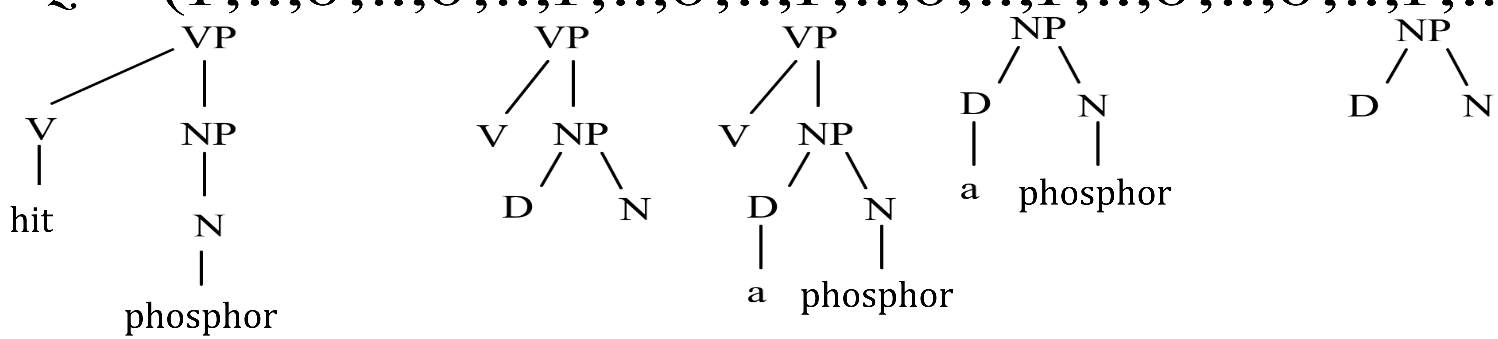


The resulting explicit kernel space

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



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

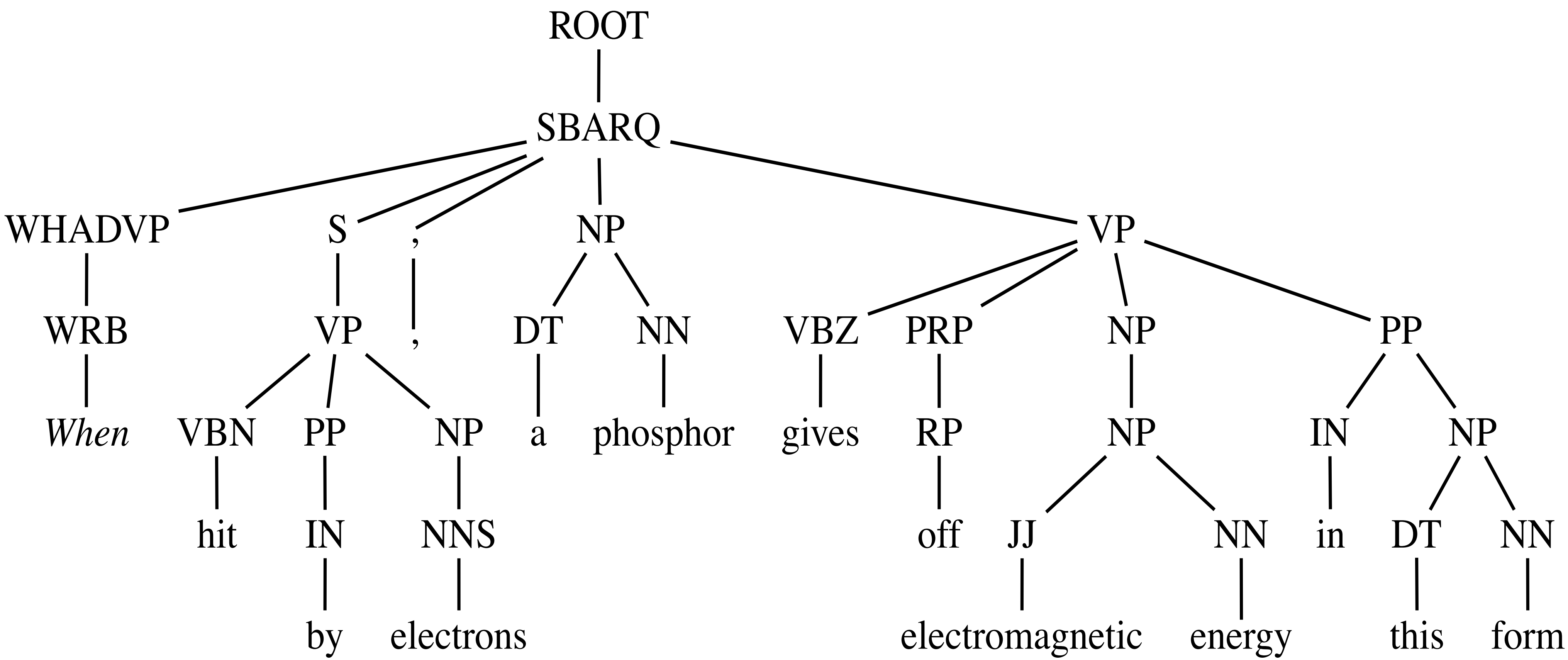


- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

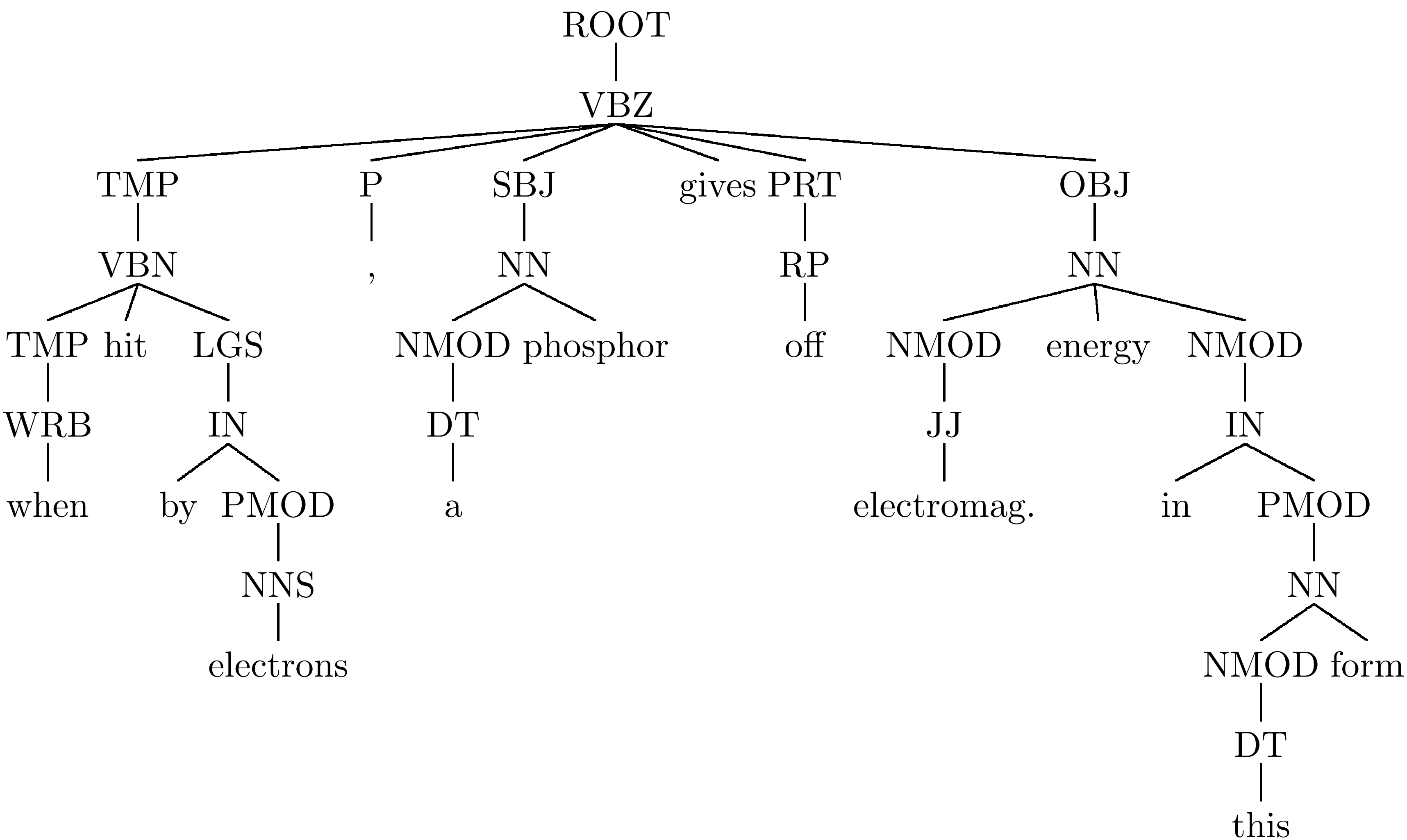
Experimental setup

- 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

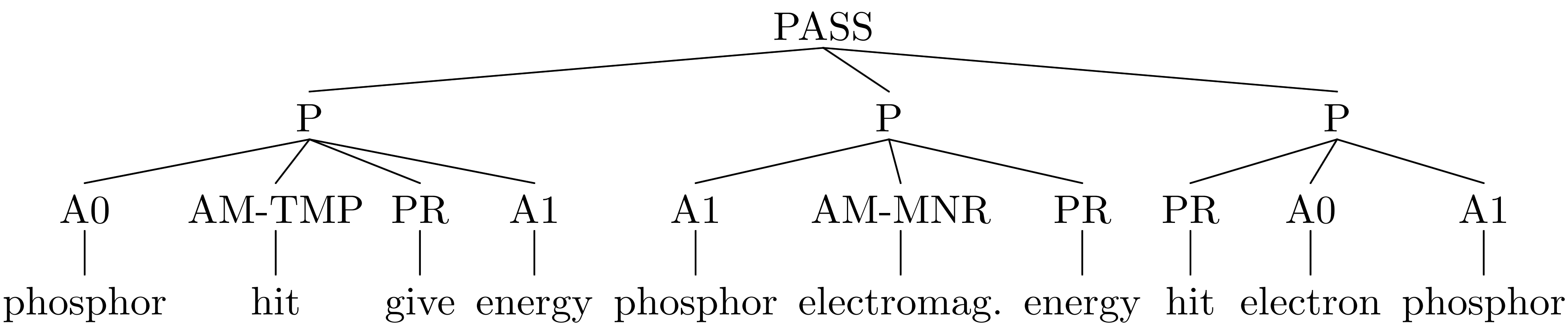
Constituency Tree (CT)



Dependency Tree (DT)



Predicate Argument Structure Set (PASS)



Sequence Kernels

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)

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**

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

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
- All questions evaluation
  - Selected 66 unseen Jeopardy! games
  - 3,546 questions

Watson's Accuracy, Precision and Earnings

- Comparison between use or not QC
- Different set of questions

	NoDef	StatDef	NoDef	RuleDef
# Questions	1606	1606	1875	1875
Accuracy	63.76%	65.57%	56.64%	57.51%
P@70	82.22%	84.53%	72.73%	74.87%

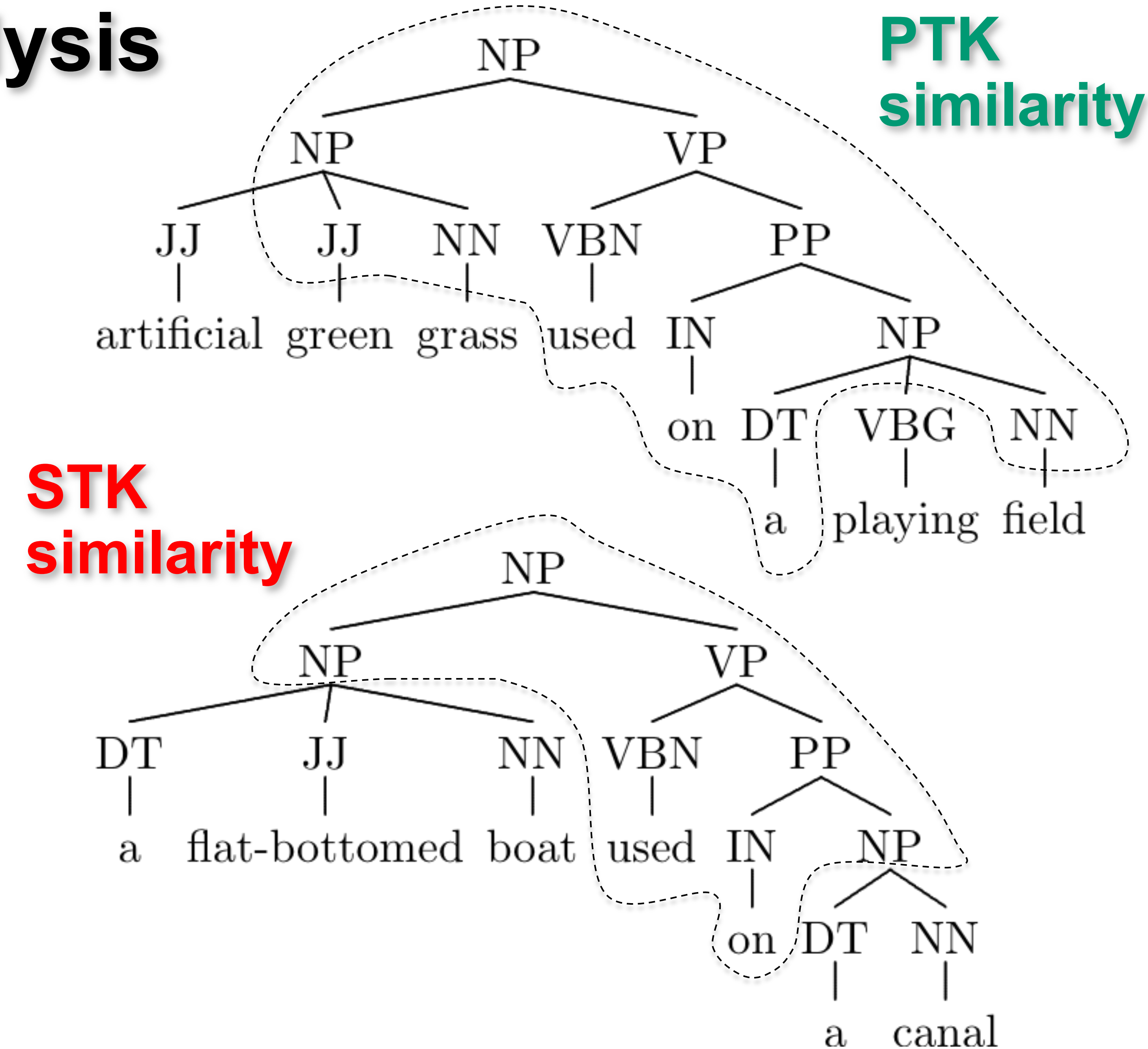
	# Def Q's	Accuracy	P@70	Earnings
NoDef	0	69.71%	86.79%	$24,818
RuleDef	480	69.23%	86.31%	$24,397
StatDef	131	69.85%	87.19%	$25,109

Error Analysis

Test Example

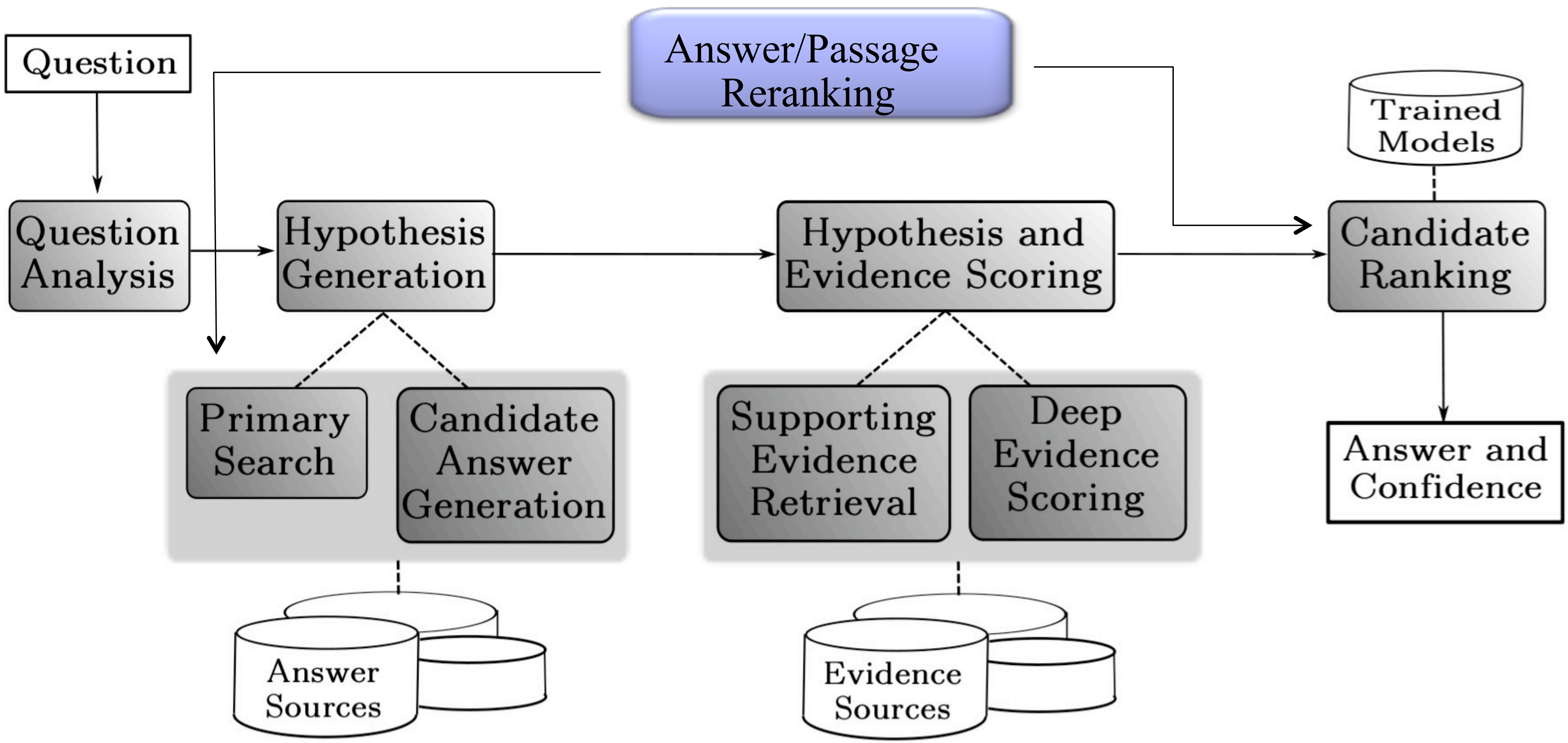
- PTK ok
- STK not ok

Training Example



Question and Answer Classification

Answer/Passage Reranking

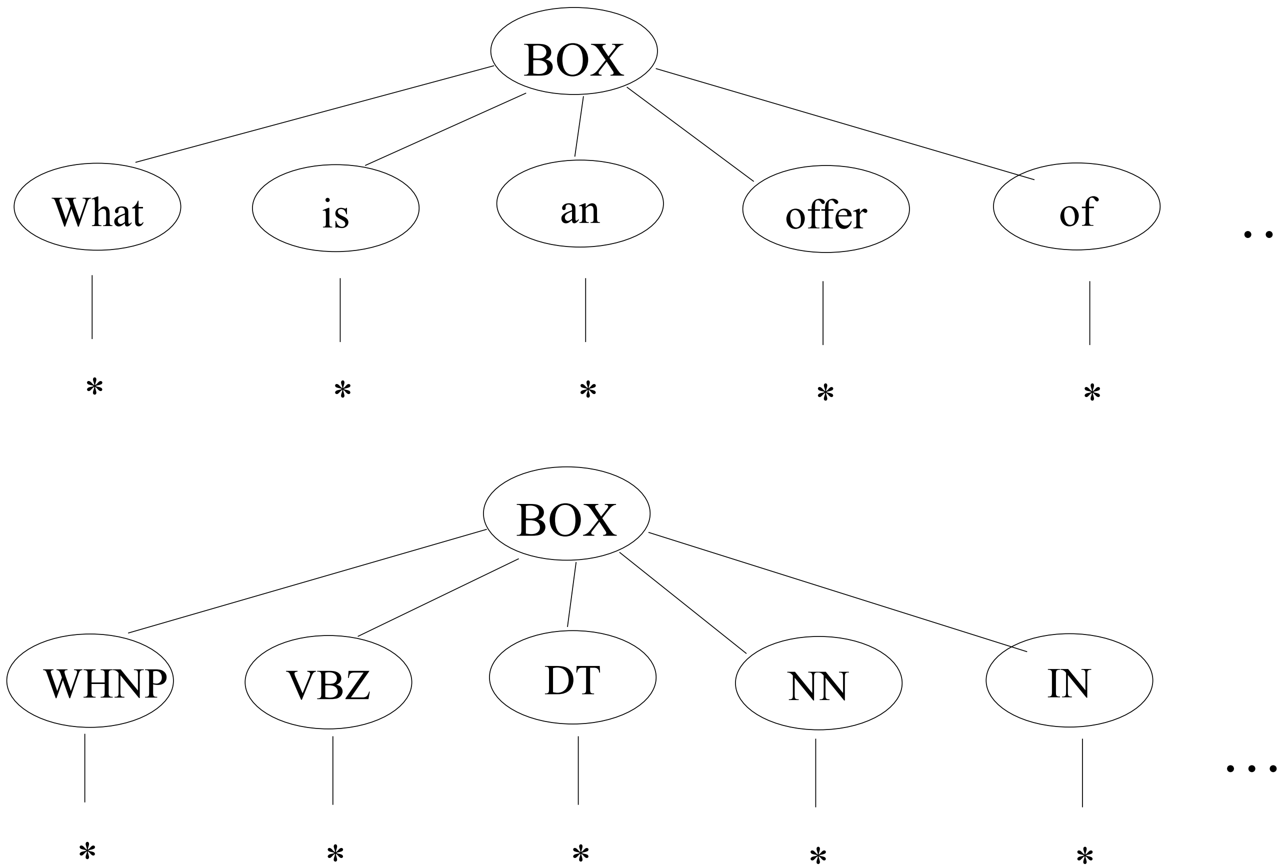


TASK: Question/Answer Classification
[Moschitti, CIKM 2008]

- 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:



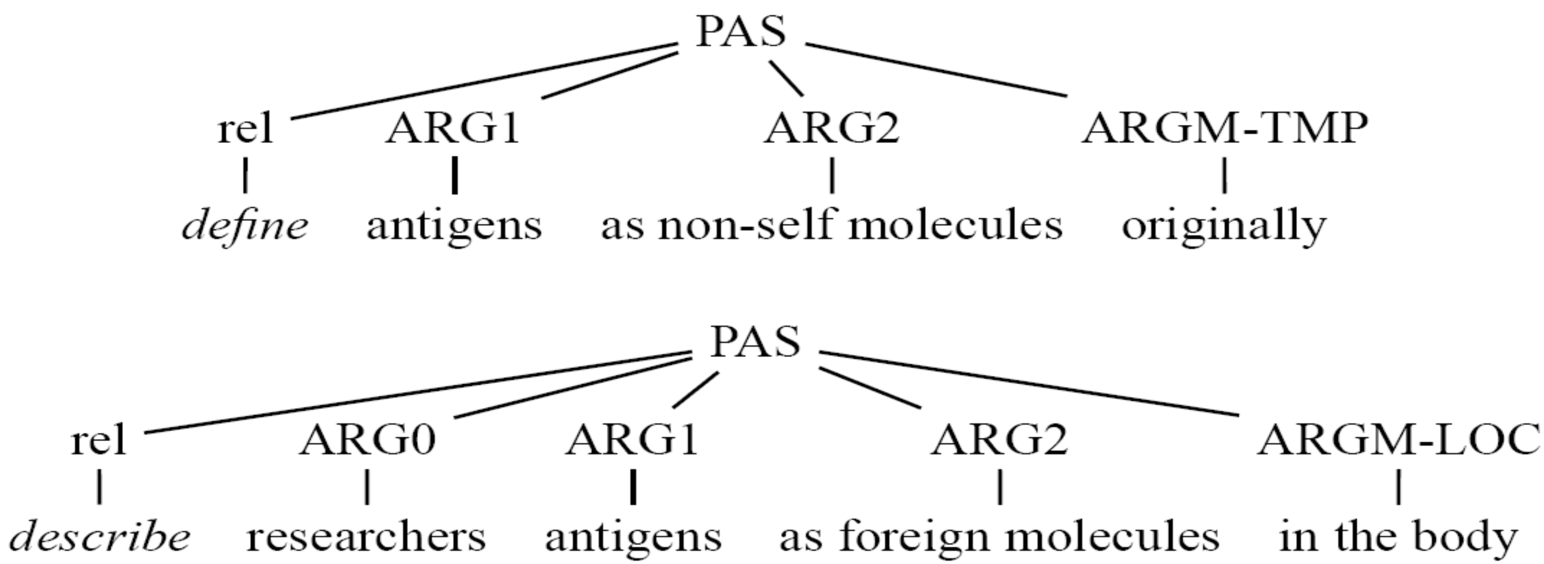
Word and POS Sequences

- 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

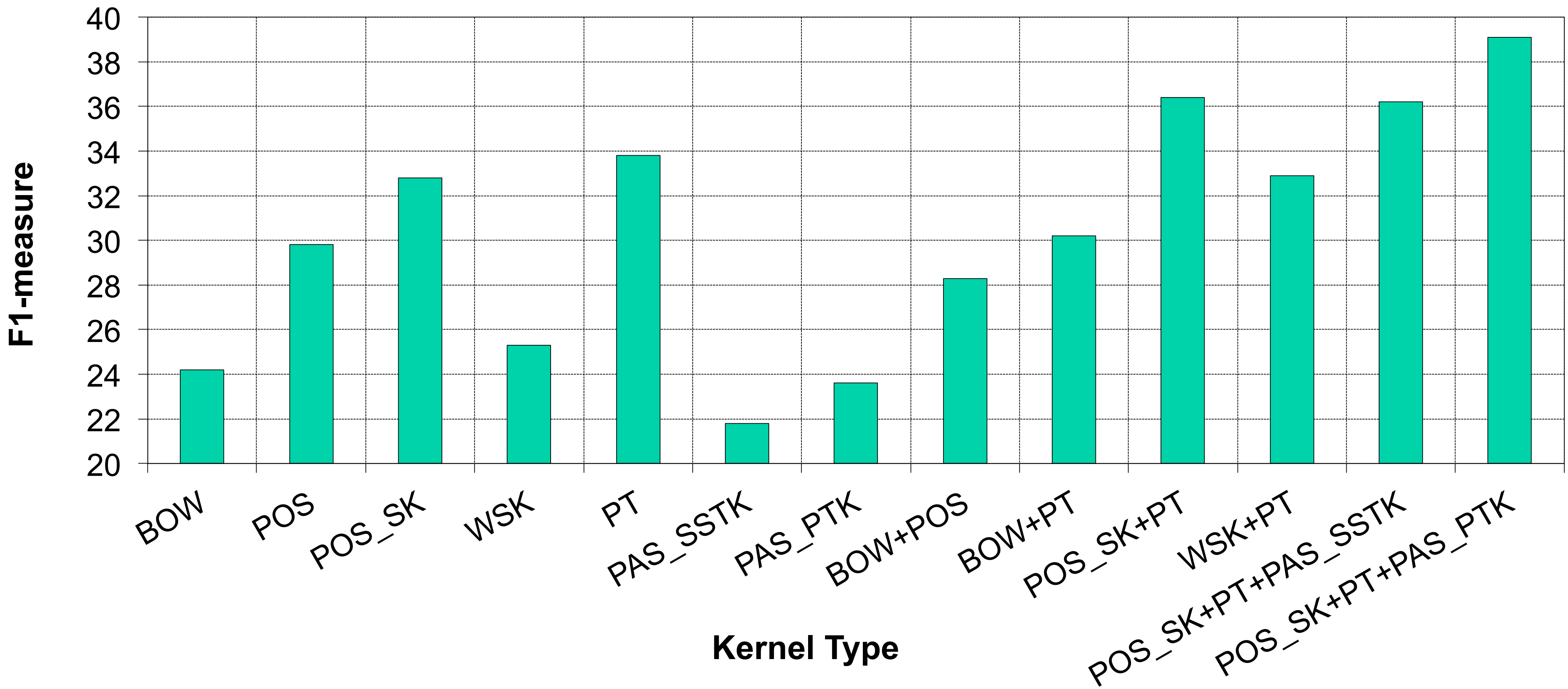
- 138 TREC 2001 test questions labeled as “description”
- 2,256 sentences, extracted from the best ranked paragraphs (using a basic QA system based on Lucene search engine on TREC dataset)
- 216 of which labeled as correct by one annotator

Kernels and Combinations

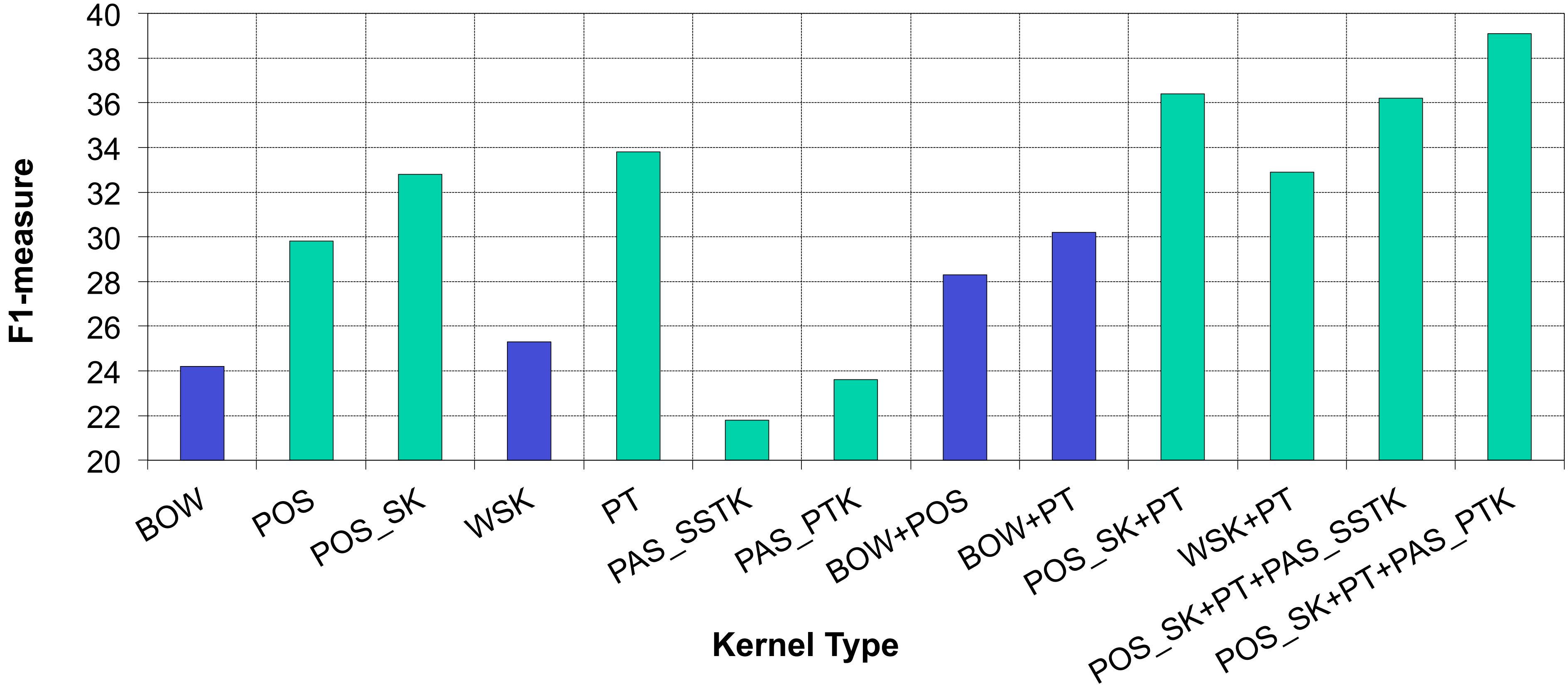
- Exploiting the property: $k(x,z) = k_1(x,z)+k_2(x,z)$
- Given: BOW, POS, WSK, POSSK, PT, PAS_{PTK}

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

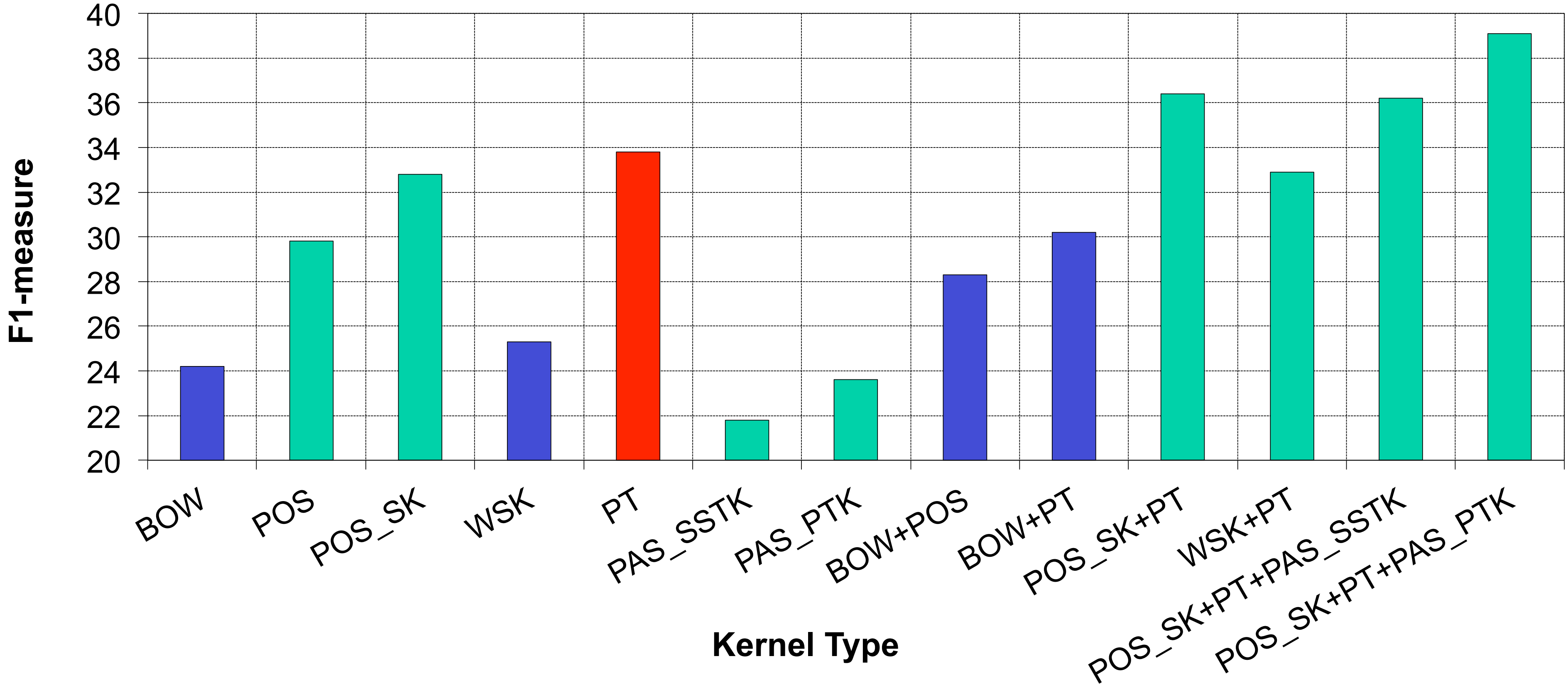
Results on TREC Data
(5 folds cross validation)



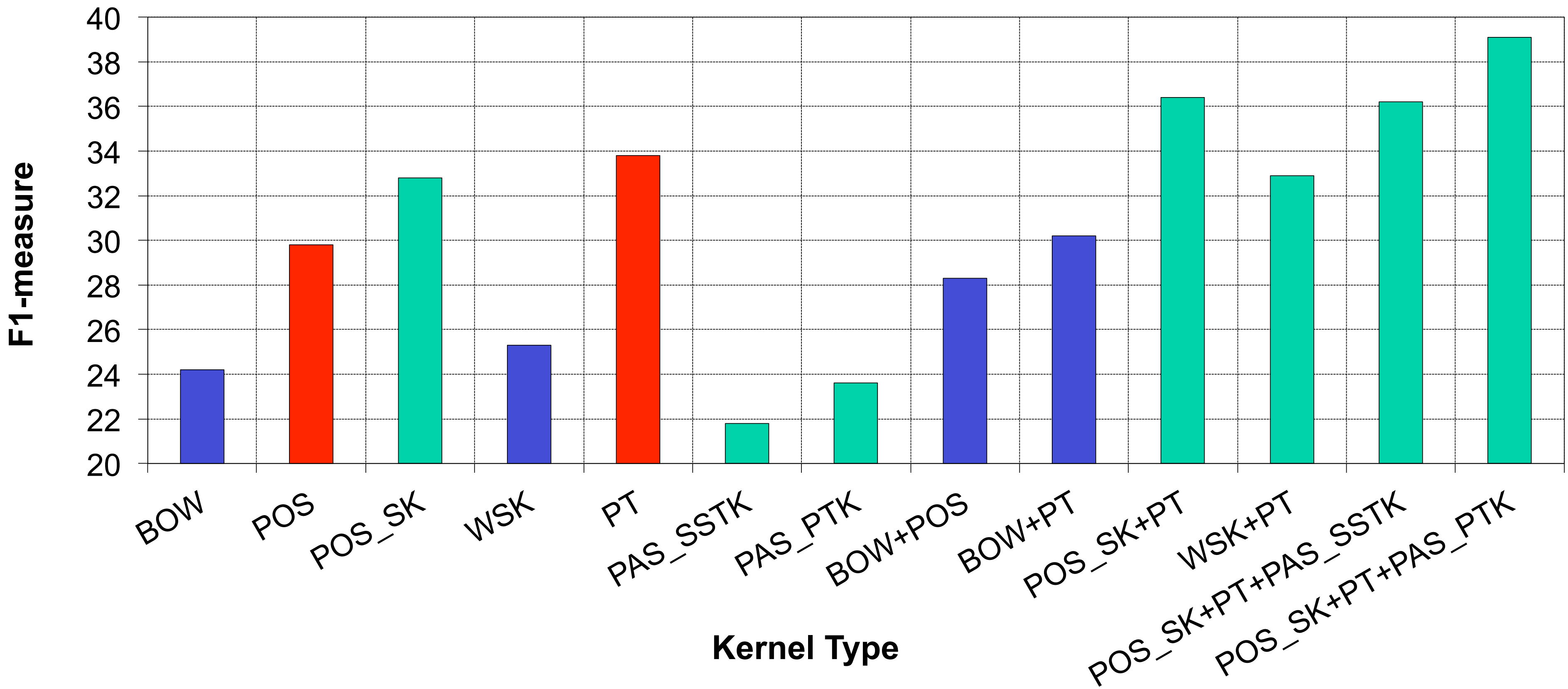
Results on TREC Data
(5 folds cross validation)



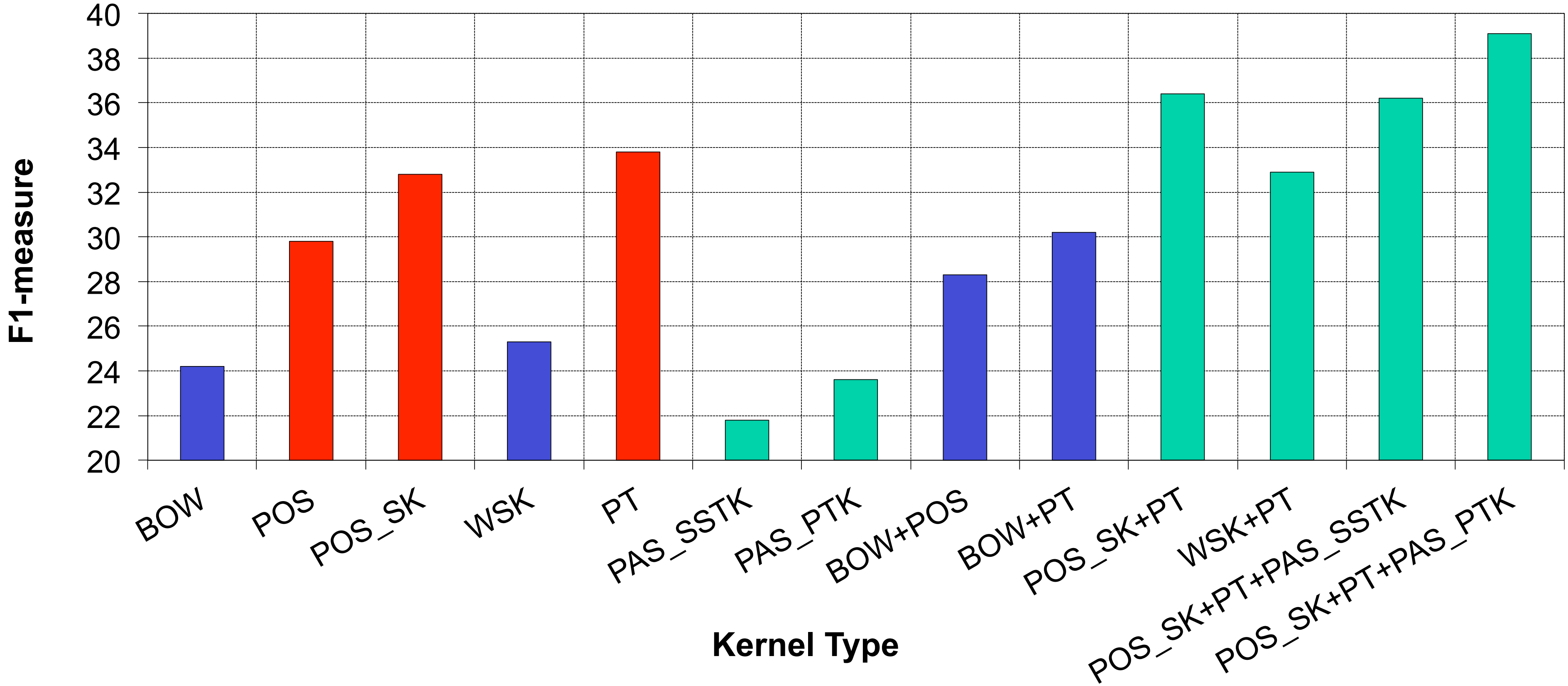
Results on TREC Data
(5 folds cross validation)



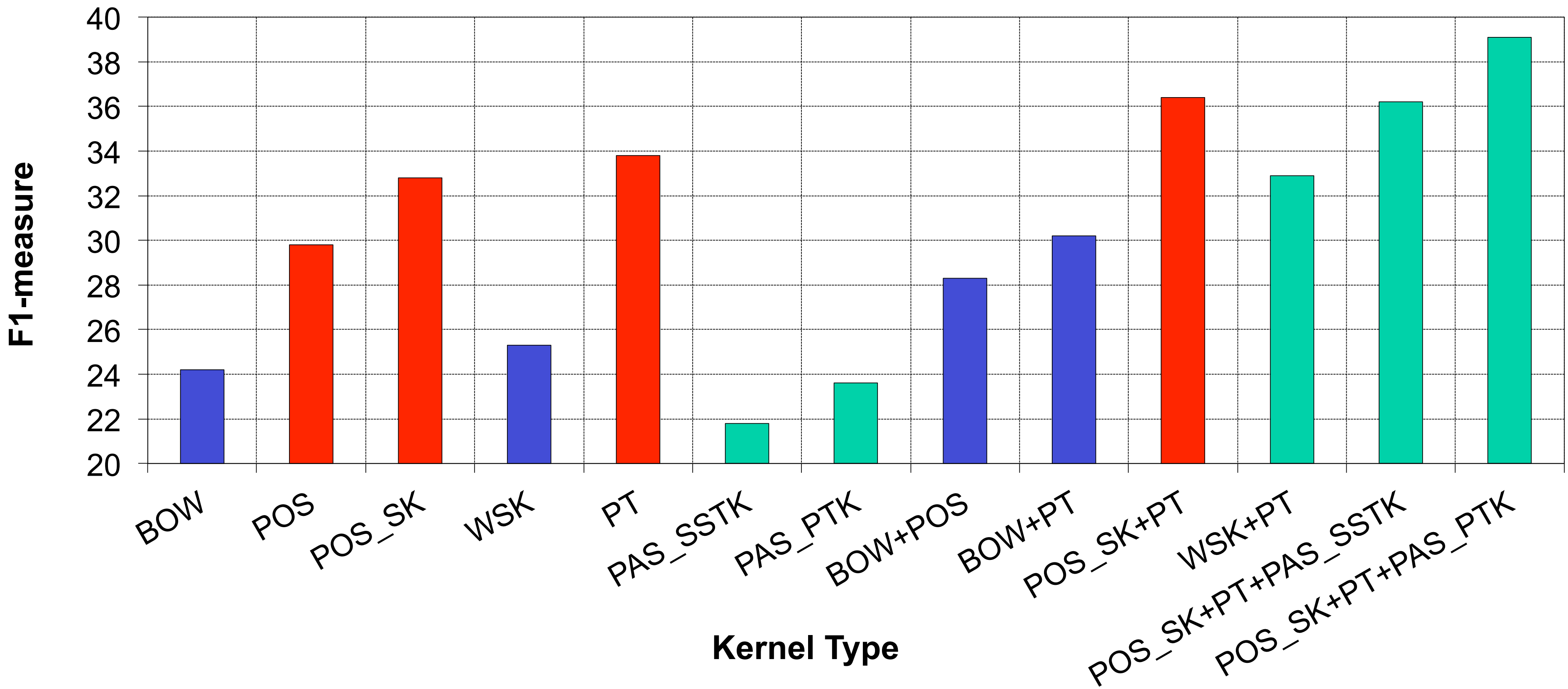
Results on TREC Data
(5 folds cross validation)



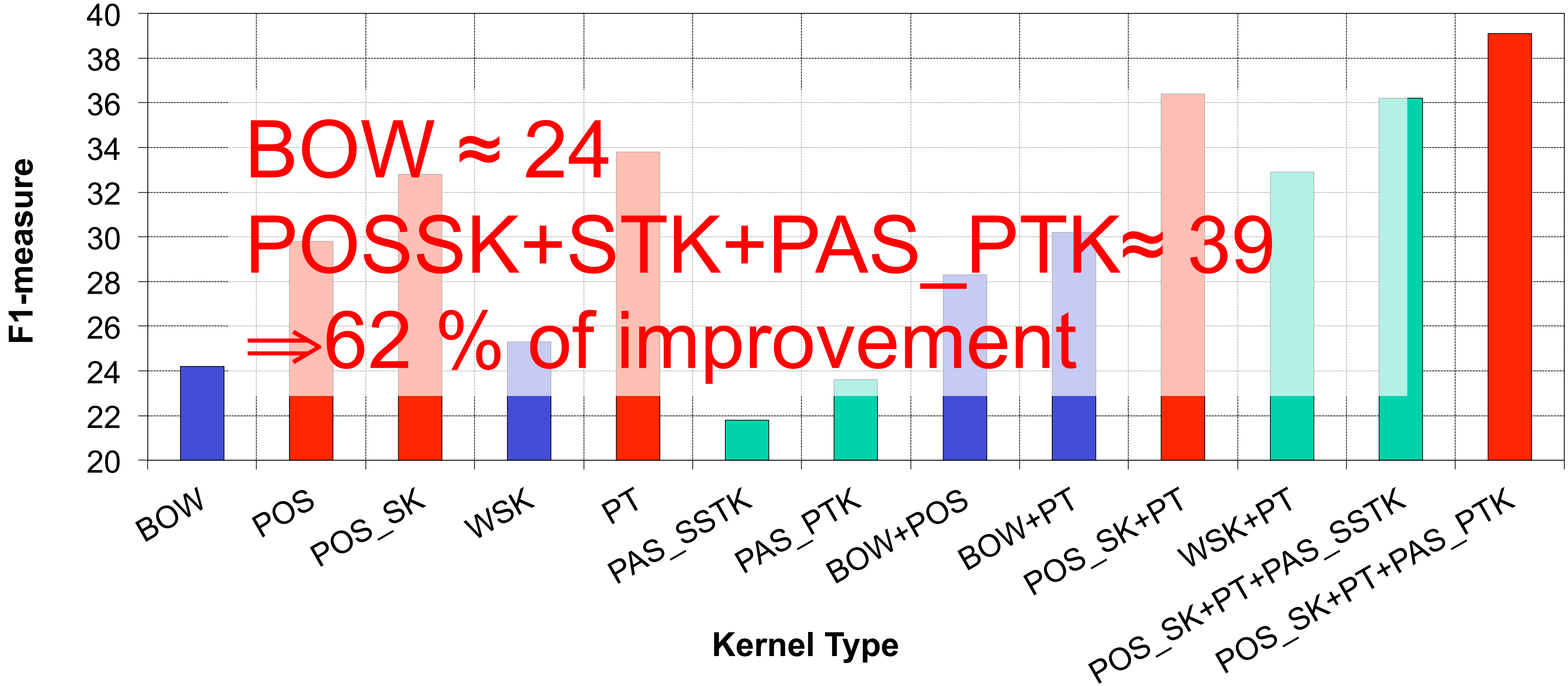
Results on TREC Data
(5 folds cross validation)



Results on TREC Data
(5 folds cross validation)



Results on TREC Data
(5 folds cross validation)



BOW ≈ 24

POSSK+STK+PAS_PTK≈ 39

⇒62 % of improvement

Break

Practical Exercise

SVM-light-TK Software

- Encodes STK, PTK and combination kernels in SVM-light [Joachims, 1999]
- Available at http://disi.unitn.it/moschitti
- Tree forests, vector sets
- You can download the version I am using from:
- http://disi.unitn.it/moschitti/material/ACL2012-Tutorial.Moschitti.zip

Data Format

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

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

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

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

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

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

Kernel Combinations an example

K_p^3 polynomial kernel of flat features

K_{Tree} Tree kernel

- Kernel Combinations:

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

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

Basic Commands

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

More on kernel applications

Semantic Role Labeling

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

  Paul gives a talk in Rome

Example on Predicate Argument Classification

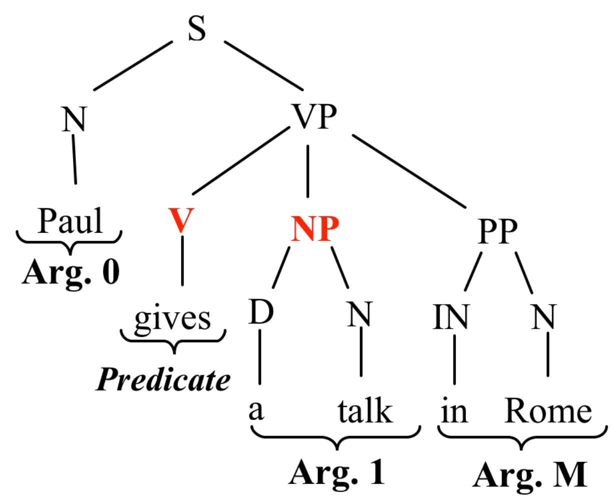
- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

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

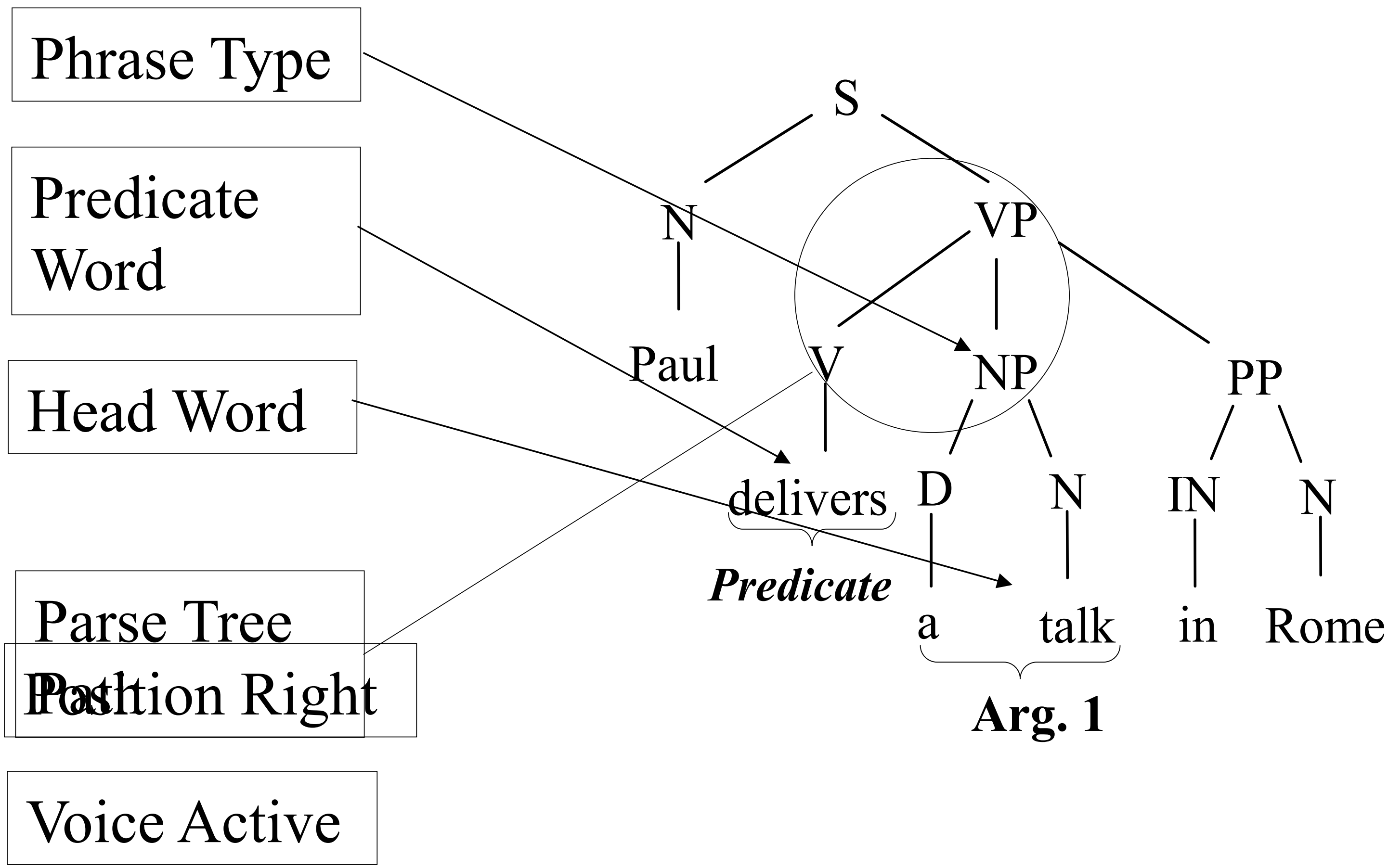
Predicate-Argument Feature Representation

Given a sentence, a predicate *p*:

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



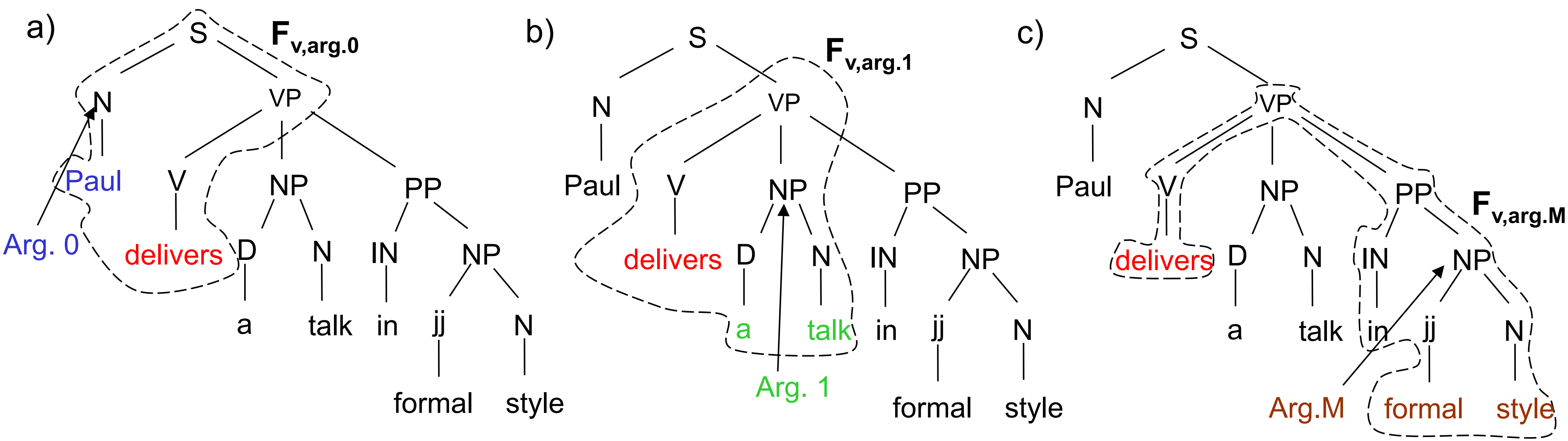
Vector Representation for the linear kernel



PAT Kernel [Moschitti, ACL 2004]

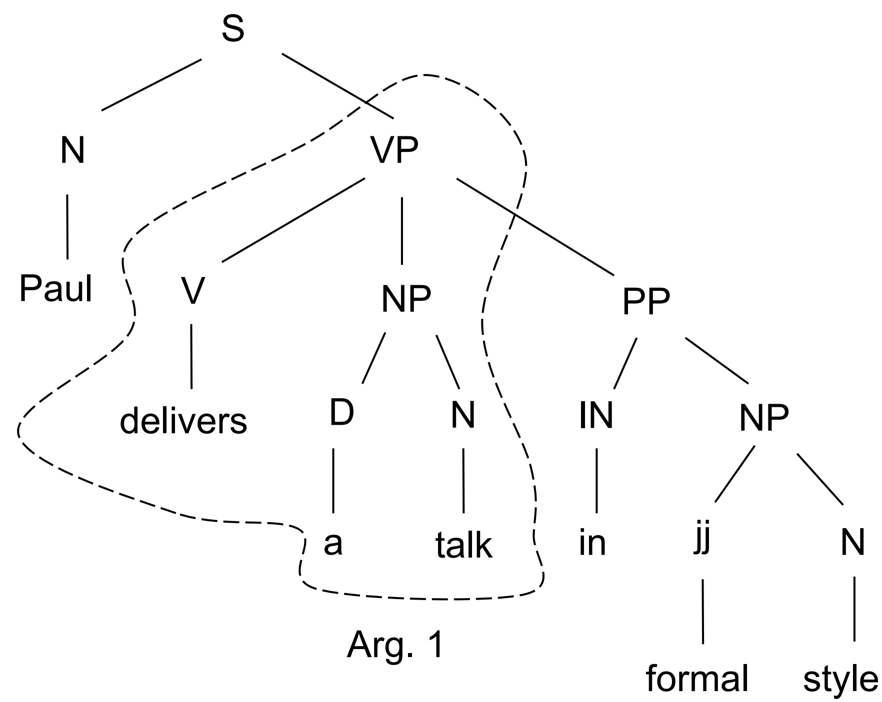
- Given the sentence:

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

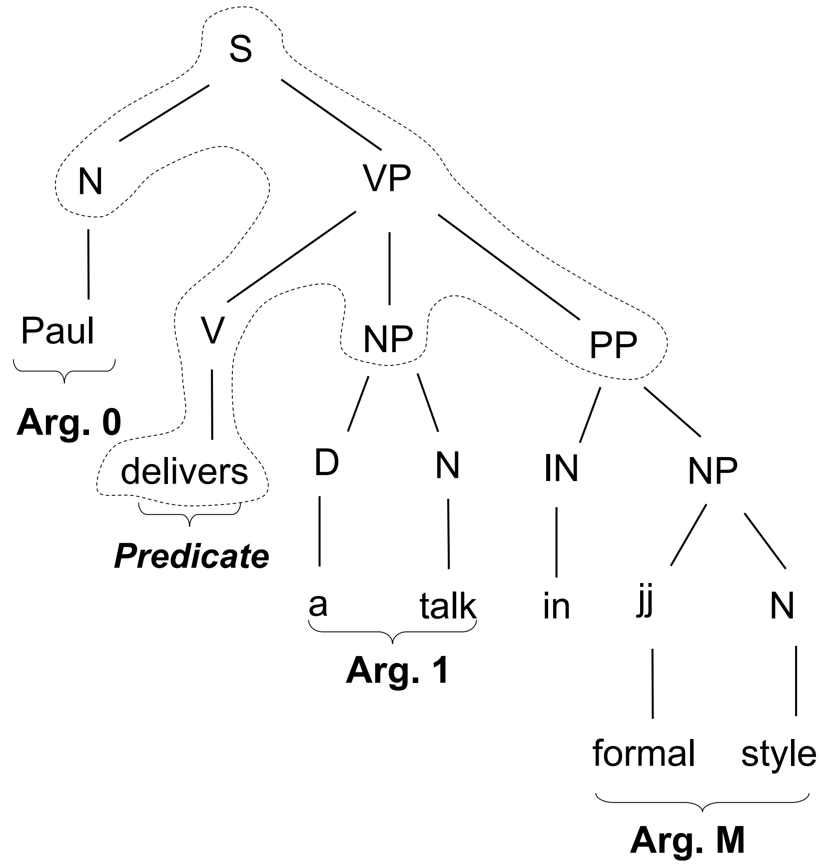


- These are Semantic Structures

In other words we consider…



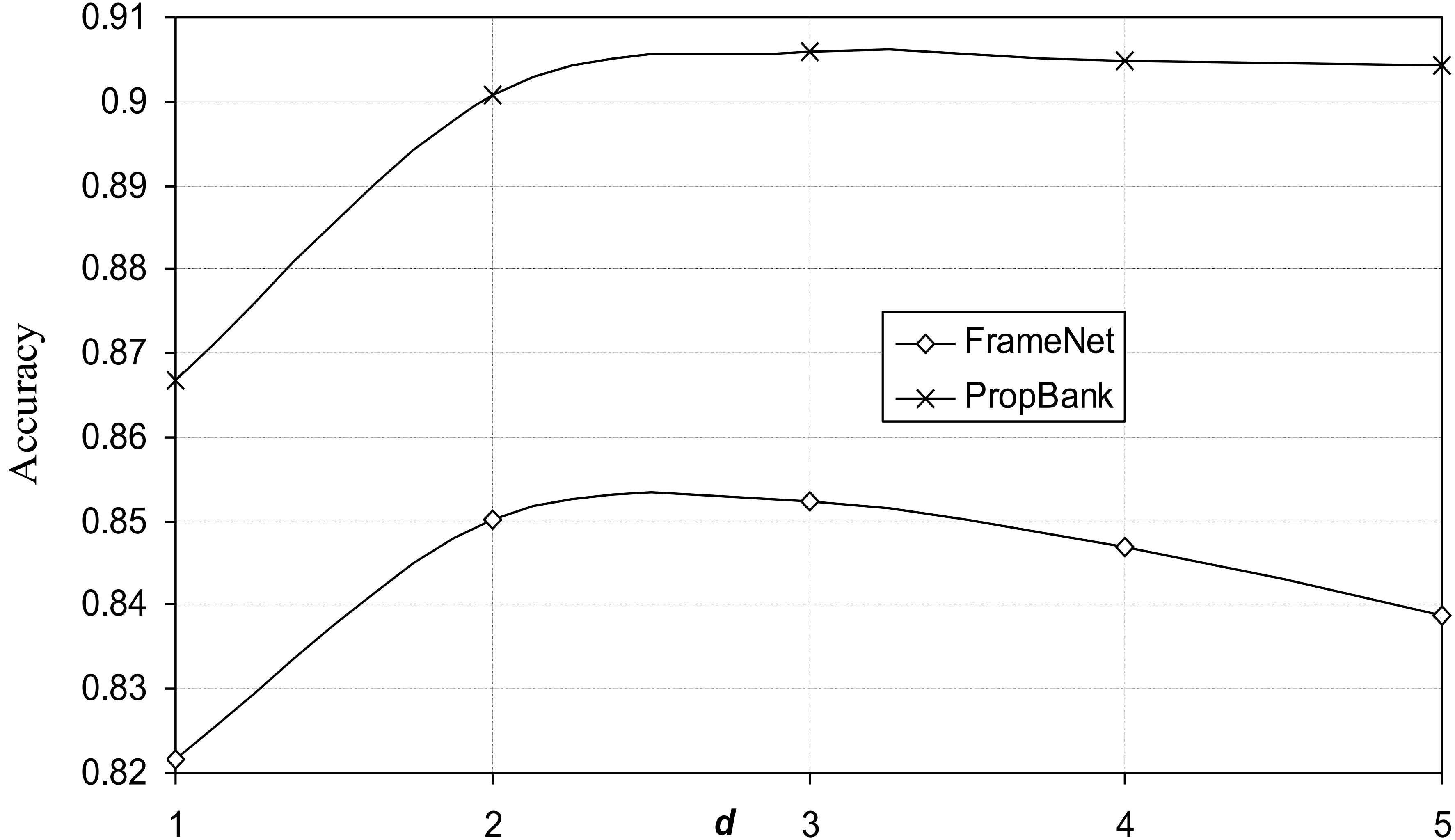
Sub-Categorization Kernel (SCF) [Moschitti, ACL 2004]



Experiments on Gold Standard Trees

- PropBank and PennTree bank
  - about 53,700 sentences
  - Sections from 2 to 21 train., 23 test., 1 and 22 dev.
  - Arguments from Arg0 to Arg5, ArgA and ArgM for a total of 122,774 and 7,359
- FrameNet and Collins' automatic trees
  - 24,558 sentences from the 40 frames of Senseval 3
  - 18 roles (same names are mapped together)
  - Only verbs
  - 70% for training and 30% for testing

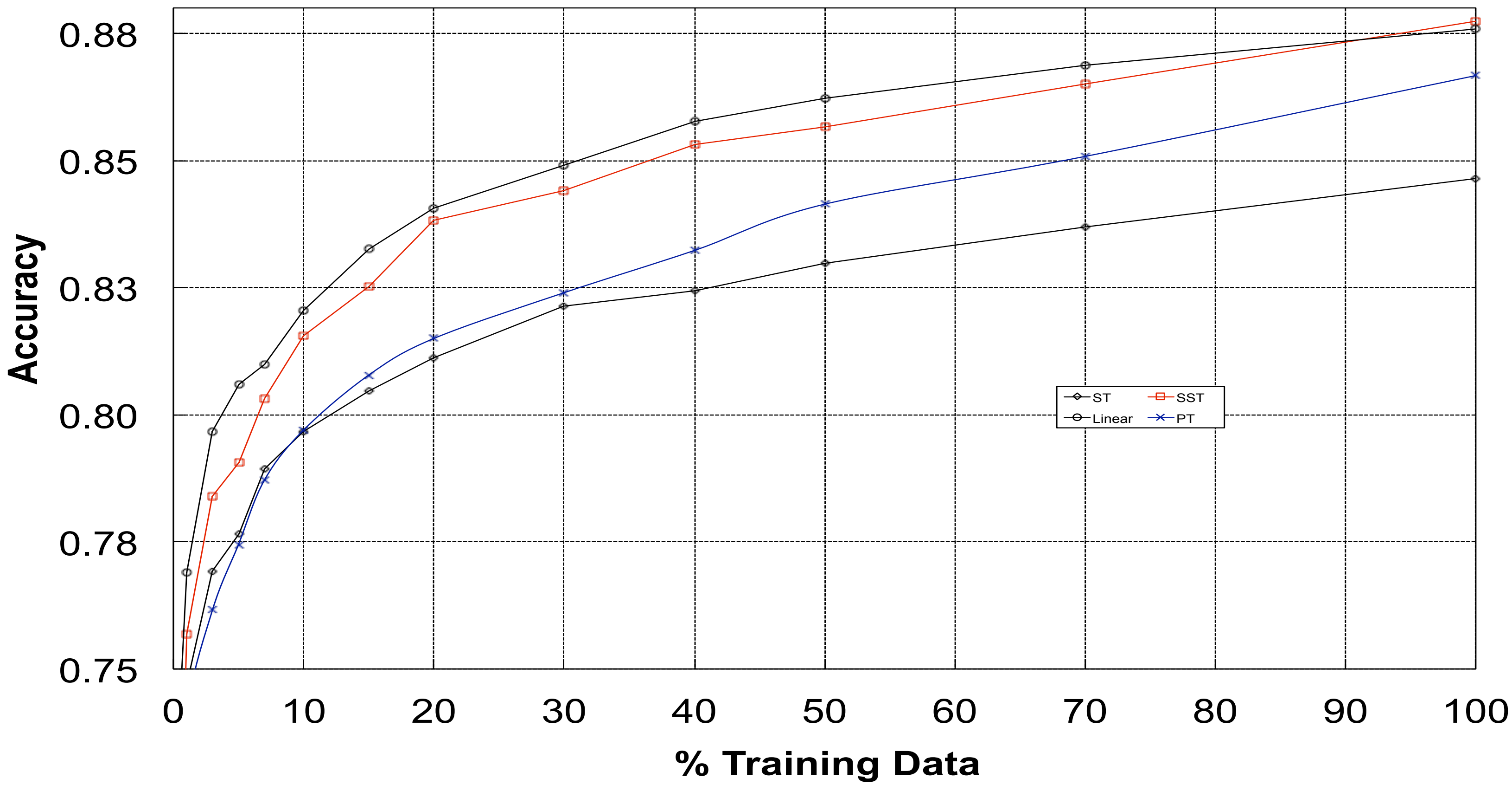
Argument Classification with Poly Kernel



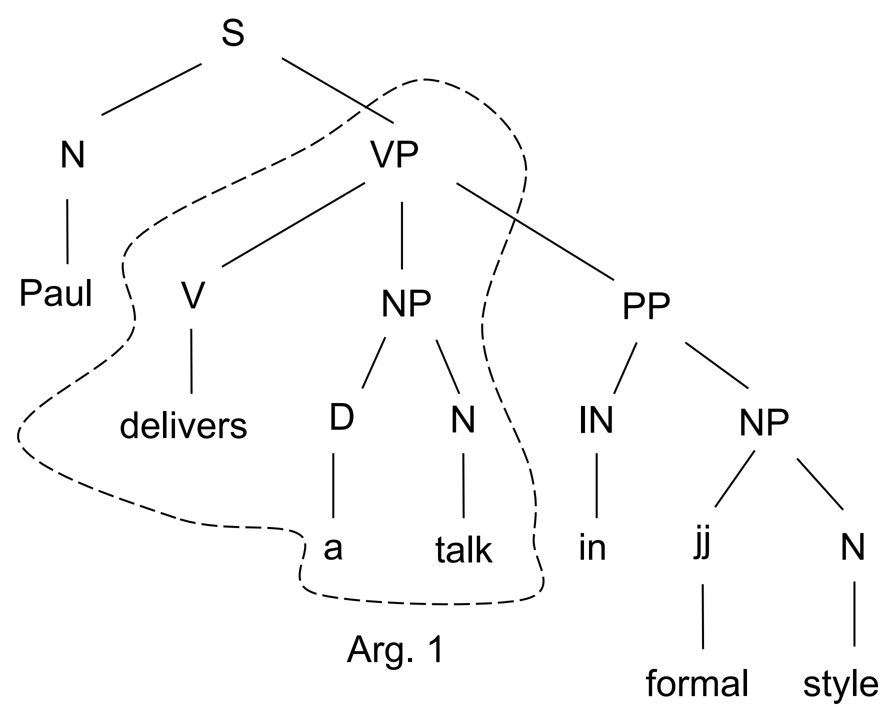
PropBank Results

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

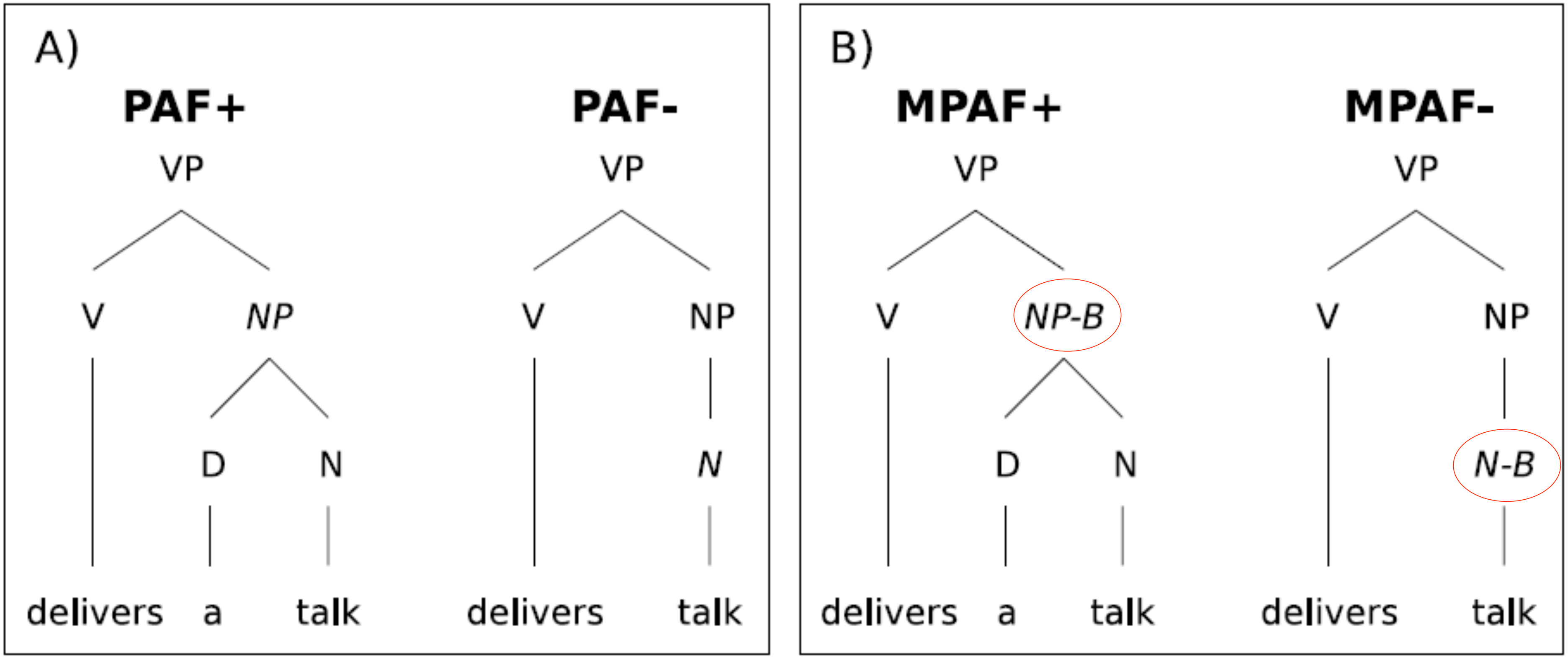
Argument Classification on PAT using different Tree Fragment Extractor



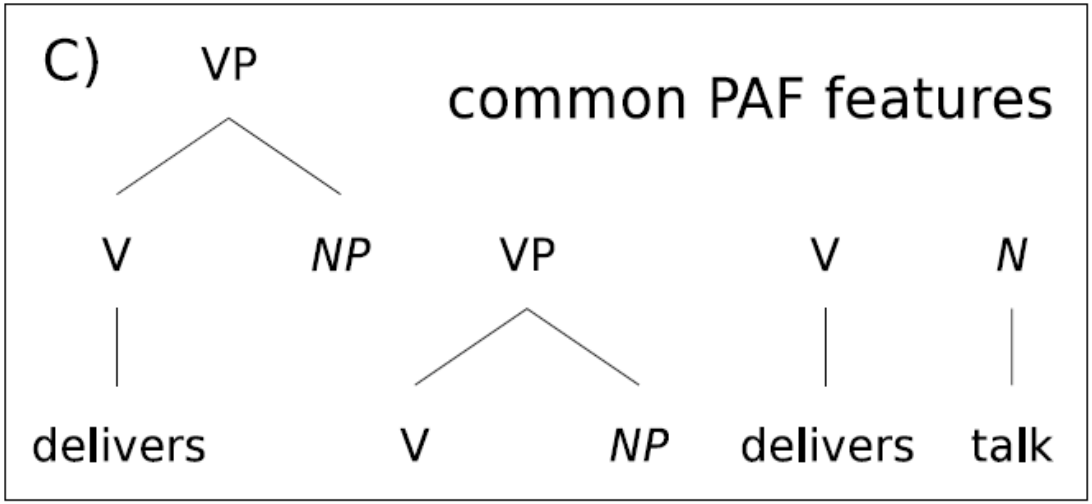
Boundary Detection



Improvement by Marking Boundary nodes



Node Marking Effect



Experiments

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

Number of examples/nodes of Section 2

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

Predicate Argument Feature (PAF) vs. Marked PAF (MPAF) [Moschitti et al, CLJ 2008]

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

Results on FrameNet SRL
[Coppola and Moschitti, LREC 2010]

- 135,293 annotated and parsed sentences.
- 782 different frames (including split per pos-tag)
- 90% of training data for BD and BC 121,798 sentences
- 10% of testing data (1,345 sentences)

Enhanced PK+TK			
Eval Setting	P	R	F_1
BD (nodes)	1.0	.732	.847
BD (words)	.963	.702	.813
BD+RC (nodes)	.784	.571	.661
BD+RC (words)	.747	.545	.630

Experiments on Luna Corpus
[Coppola at al, SLT 2008]

- BD and RC over 50 Human-Human dialogs
  - 1,677 target words spanning 162 different frames
  - manually-corrected syntactic trees
  - Training 90% data and testing on remaining 10%

Evaluation Stage	Precision	Recall	F1
Boundary Detection	**0.905**	**0.873**	**0.889**
Boundary Detection + Role Classification	**0.774**	**0.747**	**0.760**

- Automatic SRL viable for Spoken Dialog Data

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?

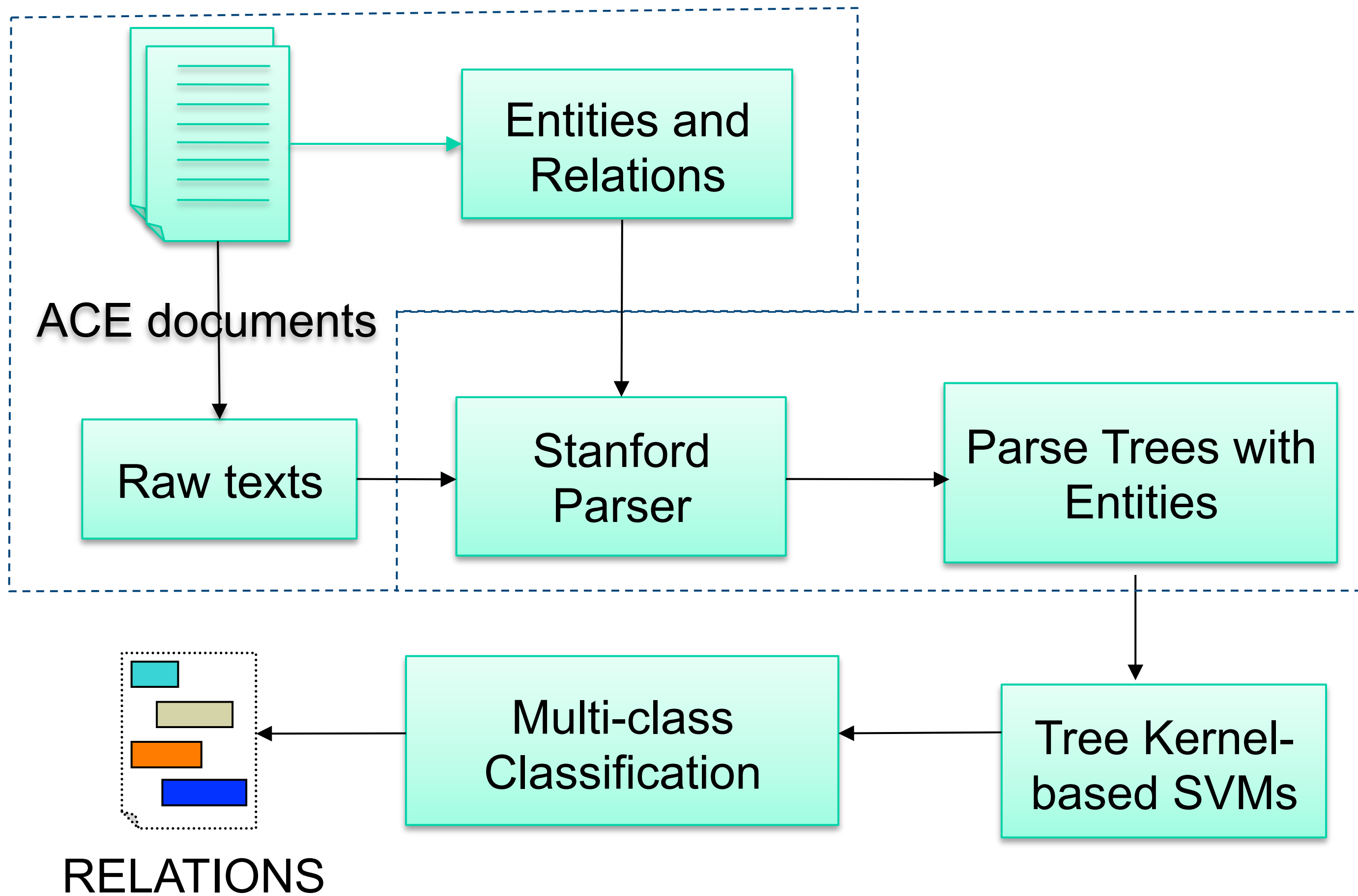
Relation Extraction defined in ACE

- Major relation types (from ACE 2004)

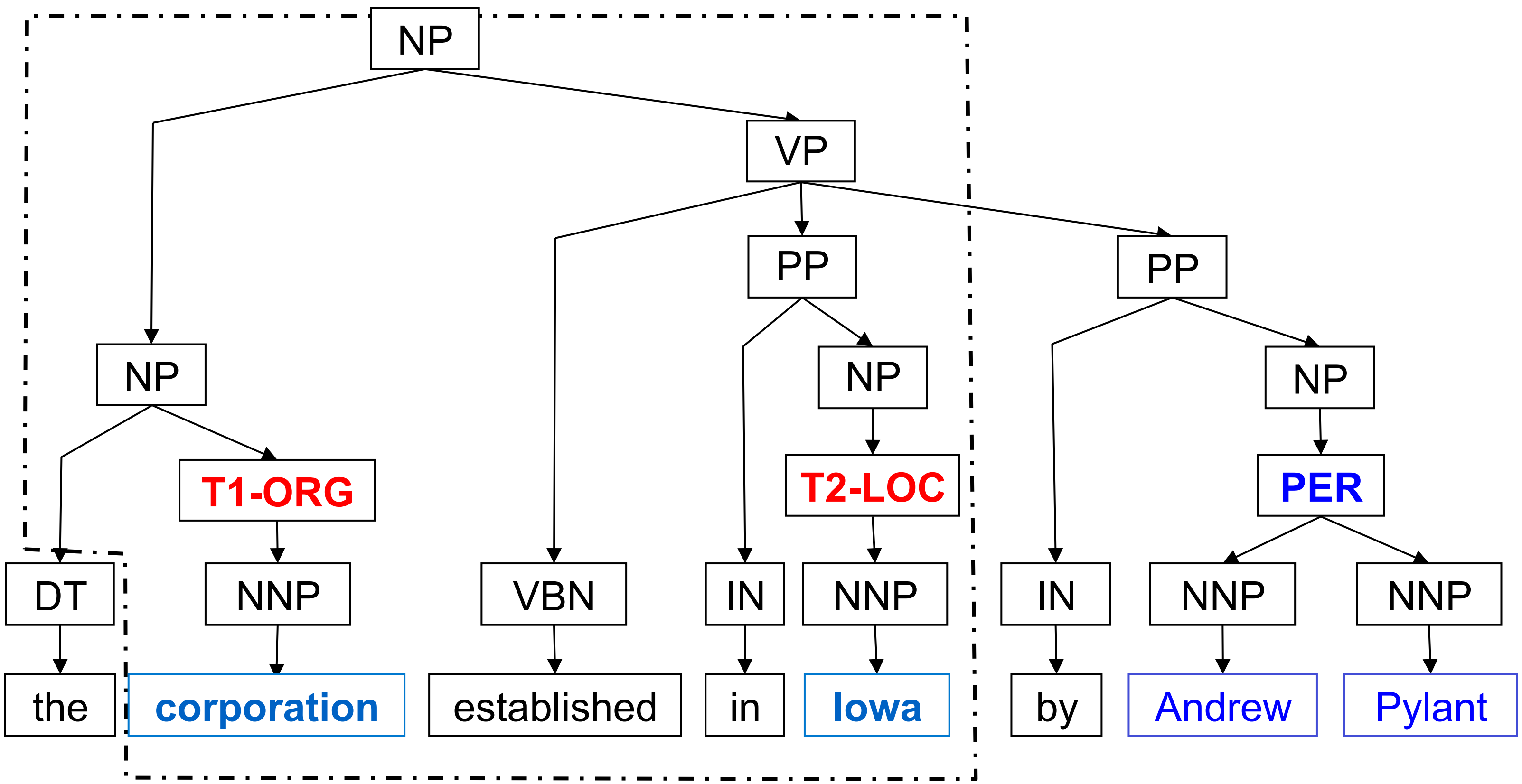
Type	Definition	Example
EMP-ORG	Employment	*US president*
PHYS	Located, near, part-whole	*a military base in Germany*
GPE-AFF	Affiliation	*U.S. businessman*
PER-SOC	Social	*a spokesman for the senator*
DISC	Discourse	*each of whom*
ART	User, owner, inventor …	*US helicopters*
OTHER-AFF	Ethnic, ideology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

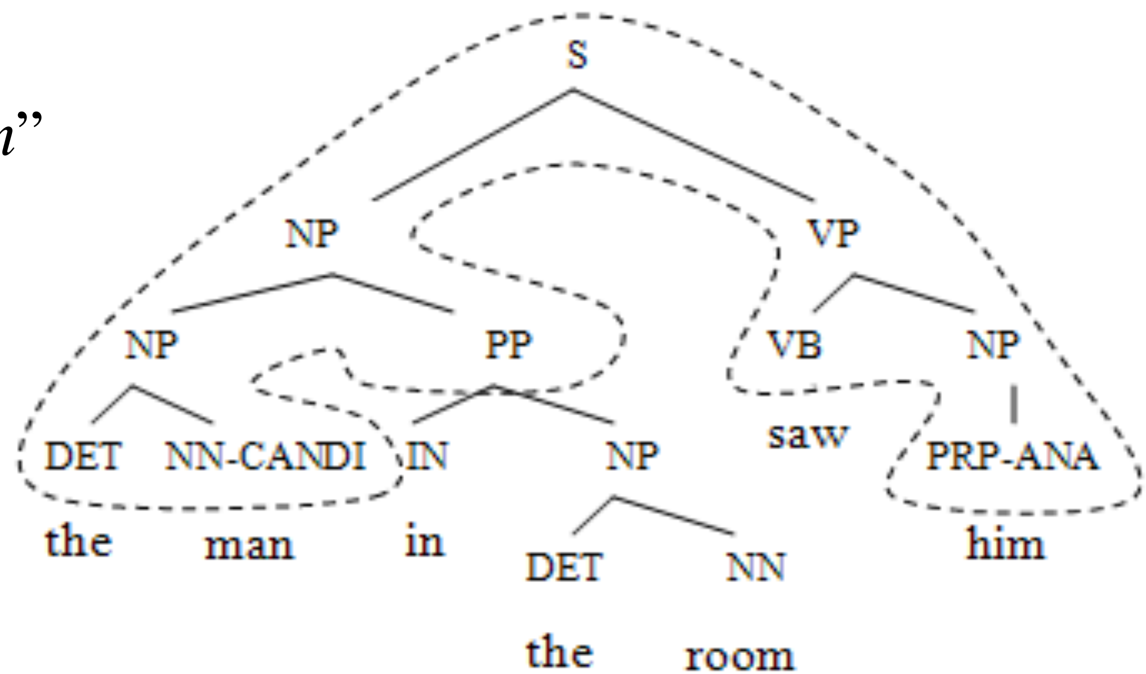
- Subtree that covers both anaphor and antecedent candidate

⇒ 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]

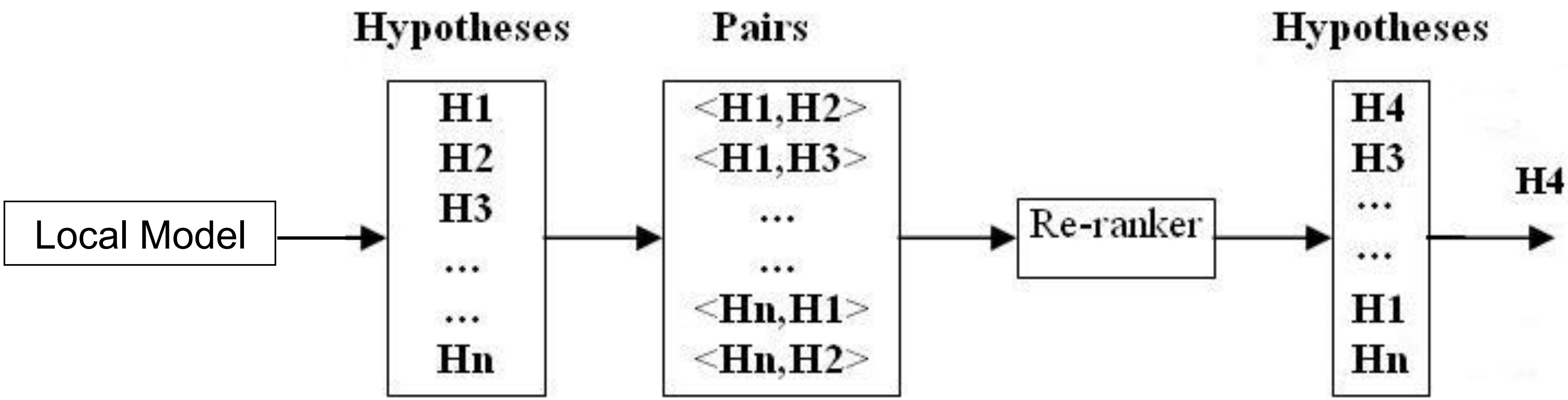
	MUC-6			ACE-02-BNews		
	R	P	F	R	P	F
All attribute value features	64.3	63.1	63.7	58.9	68.1	63.1
+ Syntactic Tree + Word Sequence	65.2	80.1	**71.9**	65.6	69.7	**67.6**

Results on the overall Coreference Resolution 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 + SyntaxTree + 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

Kernels for Reranking

Reranking framework



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 Kernel

$$P_K(x, y) = \left\langle \phi(x_1) - \phi(x_2), \phi(y_1) - \phi(y_2) \right\rangle =$$

$$P_K(\langle x_1, x_2 \rangle, \langle y_1, y_2 \rangle) = K(x_1, y_1) +$$
$$K(x_2, y_2) - K(x_1, y_2) - K(x_2, y_1),$$

where K is a kernels on the text, e.g., in case of question and answer:

$$K(x_1, y_1) = \mathrm{PTK}(q_{x_1}, q_{y_1}) + \mathrm{PTK}(a_{x_1}, a_{y_1})$$

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

Concept Segmentation and Classification of speech

- Given a transcription, i.e., a sequence of words, chunk and label subsequences with concepts
- Air Travel Information System (ATIS)
  - Dialog systems answering user questions
  - Conceptually annotated dataset
  - Frames

An example of concept annotation in ATIS

- User request: ***list TWA flights from Boston to Philadelphia***

$$\underbrace{list}_{null}\ \underbrace{TWA}_{airline\_code}\ \underbrace{flights}_{null}\ \underbrace{from}_{null}\ \underbrace{Boston}_{fromloc.city}\ \underbrace{to}_{null}\ \underbrace{Philadelphia}_{toloc.city}$$

- The concepts are used to build rules for the dialog manager (e.g. actions for using the DB)
  - from location
  - to location
  - airline code

list flights from boston to Philadelphia	
FRAME:	FLIGHT
	FROMLOC.CITY = boston
	TOLOC.CITY = Philadelphia

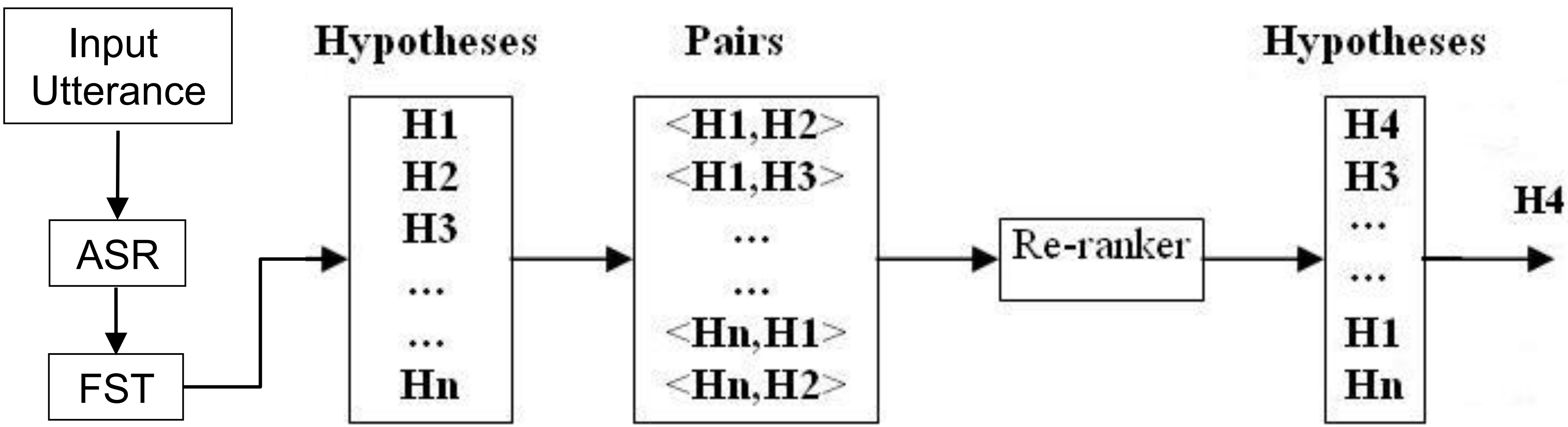
Our Approach
[Dinarelli et al., TASL 2012]

- Use of Finite State Transducer (or CRF) to generate word sequences and concepts
- Probability of each annotation

$\Rightarrow$ *m* best hypothesis can be generated

- Idea: use a discriminative model to choose the best one
  - Re-ranking and selecting the top one

Reranking for SLU



Reranking concept labeling

- *I have a problem with my monitor*

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

H ^{j}: *I* ***NULL*** *have* ***NULL*** *a* ***NULL*** *problem* ***HW-B*** *with* ***NULL*** *my* ***NULL*** *monitor*

Luna Corpus

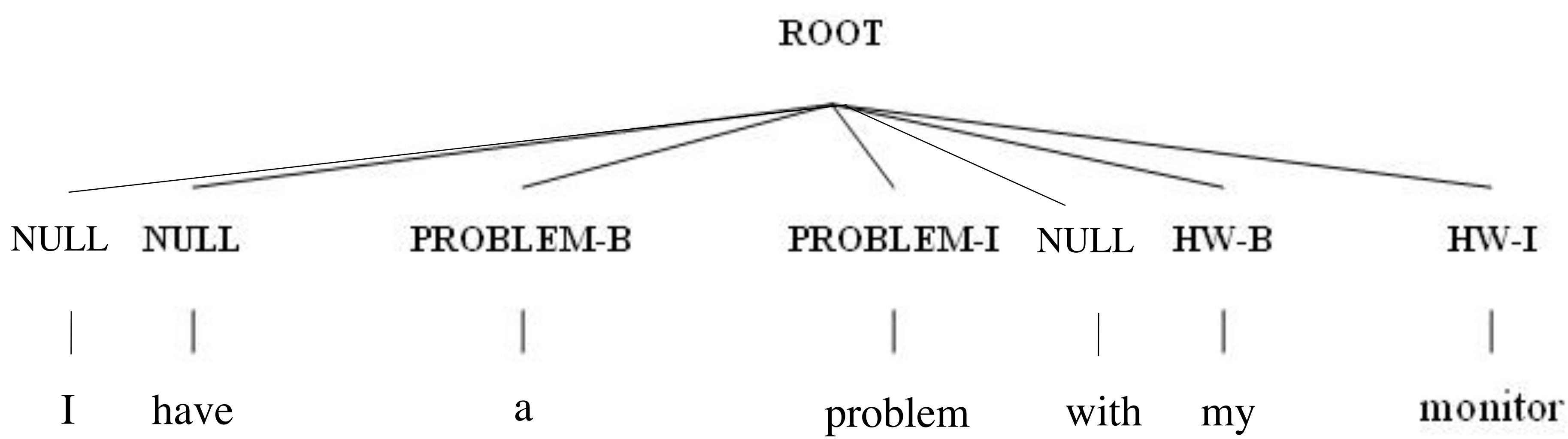
- Wizard of OZ, helpdesk scenario

Corpus LUNA	Training set		Test set	
	words	concepts	words	concepts
Dialogs	183		67	
Turns	1,019		373	
Tokens	8,512	2,887	2,888	984
Vocabulary	1,172	34	-	-
OOV rate	-	-	3.2%	0.1%

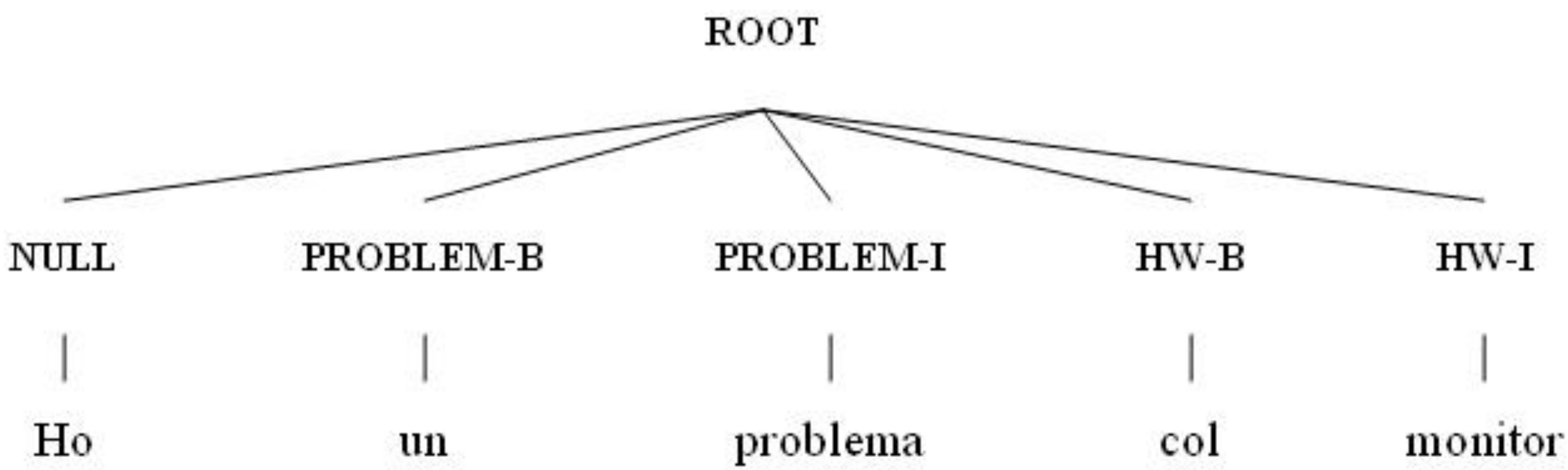
Media Corpus

	training		development		test	
# sentences	12,908		1,259		3,005	
	words	concepts	words	concepts	words	concepts
# tokens	94,466	43,078	10,849	4,705	25,606	11,383
# vocabulary	2,210	99	838	66	1,276	78
# OOV rate [%]	–	–	1.33	0.02	1.39	0.04

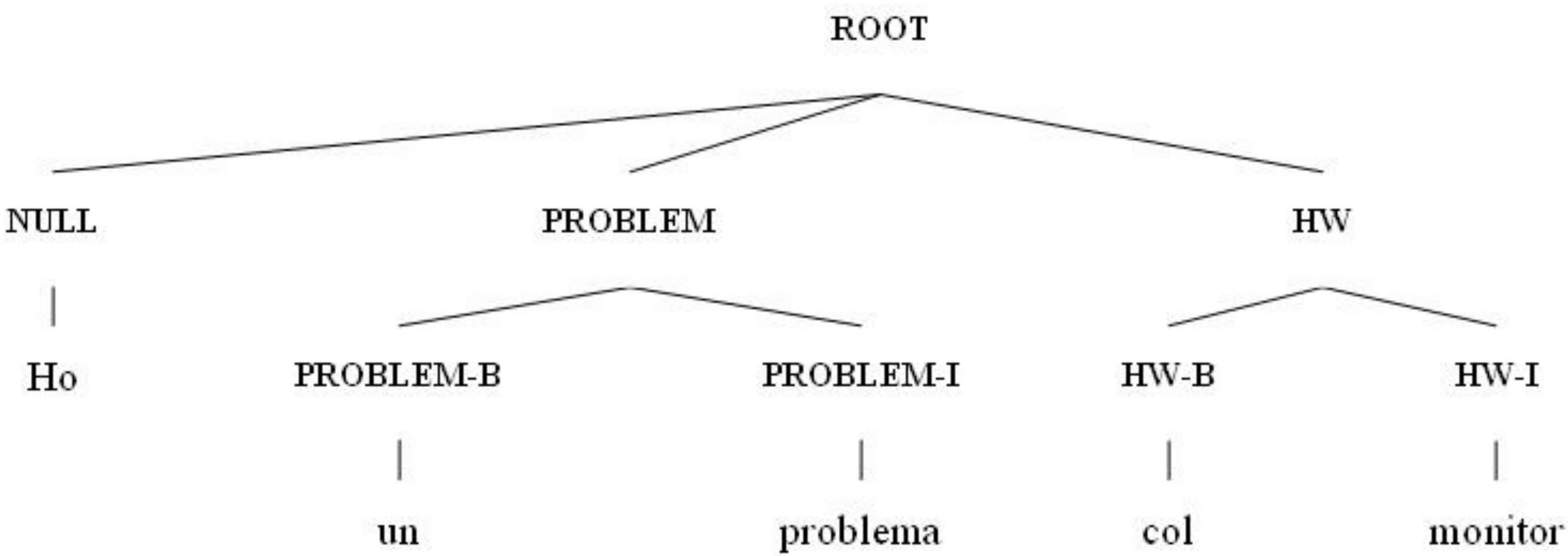
Flat tree representation



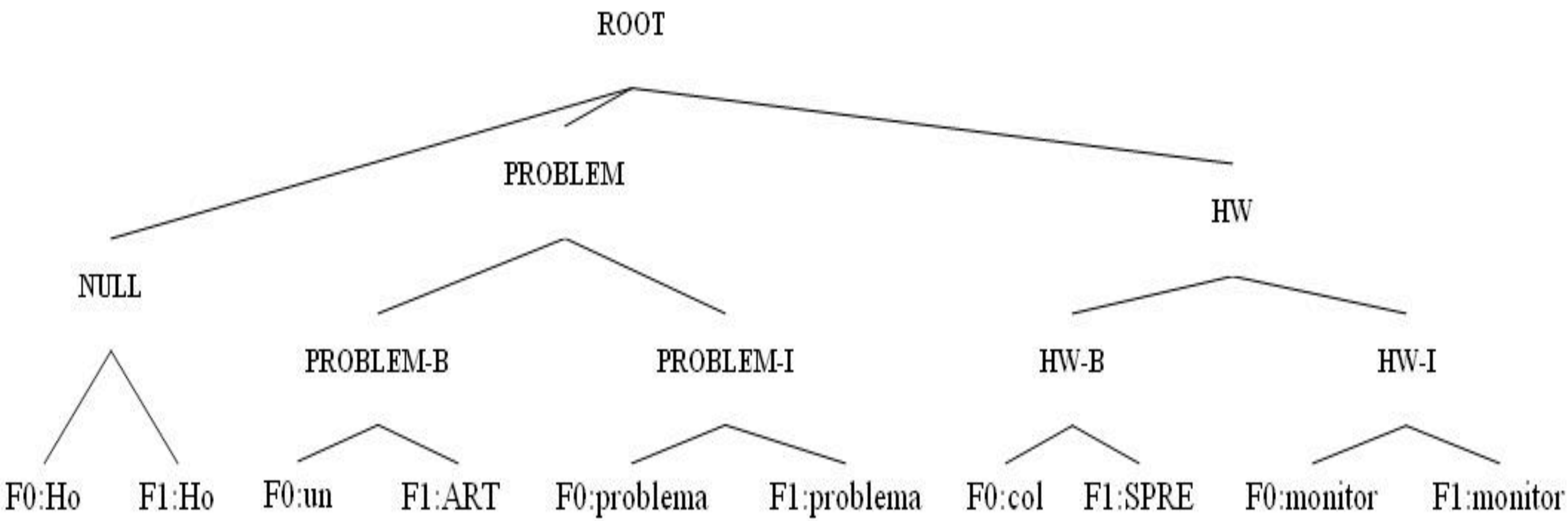
Cross-language approach: Italian version



Multilevel Tree



Enriched Multilevel Tree



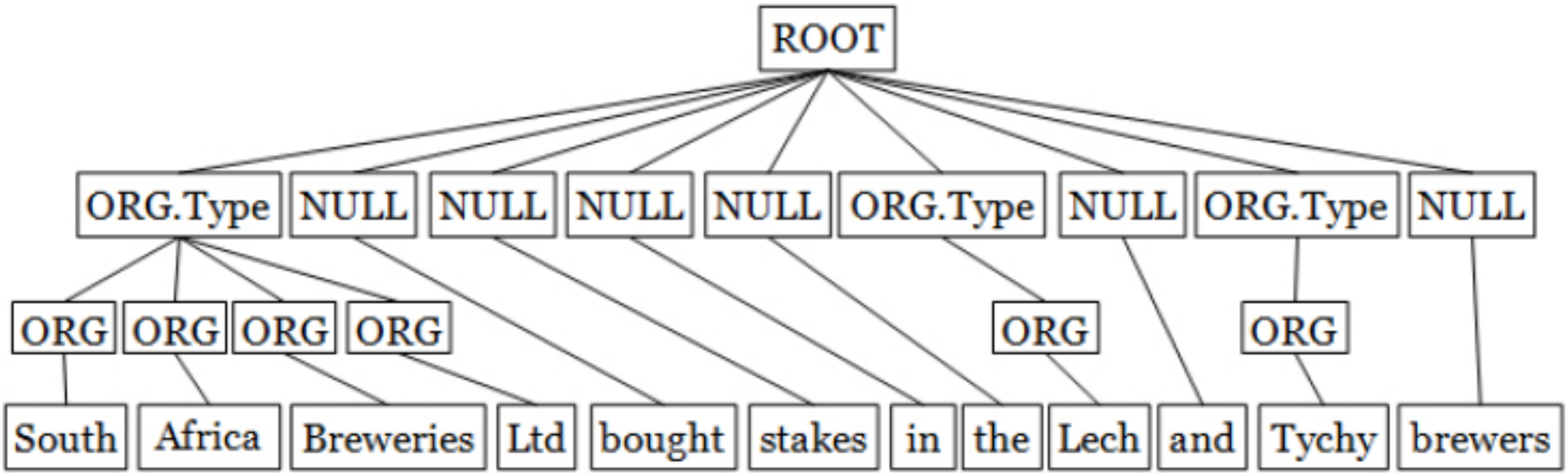
Results on LUNA

	Text Input (CER)		Speech Input (CER)	
Model	**Attr.**	**Attr.-Val.**	**Attr.**	**Attr.-Val.**
FST	24.4%	27.4%	36.4%	39.9%
SVM	25.3%	27.1%	34.0%	36.7%
CRF	21.3%	23.5%	31.0%	34.2%
FST-RR	20.7%	22.8%	32.7%	36.2%
CRF-RR	19.9%	21.9%	29.0%	32.2%
$FST + RR_S$	19.2%	21.5%	30.4%	33.8%
$CRF + RR_S$	19.0%	21.1%	28.3%	31.4%

Results on Media

	Text Input (CER)		Speech Input (CER)	
Model	**Attr.**	**Attr.-Val.**	**Attr.**	**Attr.-Val.**
FST	14.2%	17.0%	28.9%	33.6%
SVM	13.4%	15.9%	25.8%	29.7%
CRF	11.7%	14.2%	24.3%	28.2%
FST-RR	11.9%	14.6%	25.4%	29.9%
CRF-RR	11.5%	14.1%	23.6%	27.2%
$FST + RR_S$	11.3%	13.8%	24.5%	28.2%
$CRF + RR_S$	11.1%	13.1%	22.7%	26.3%

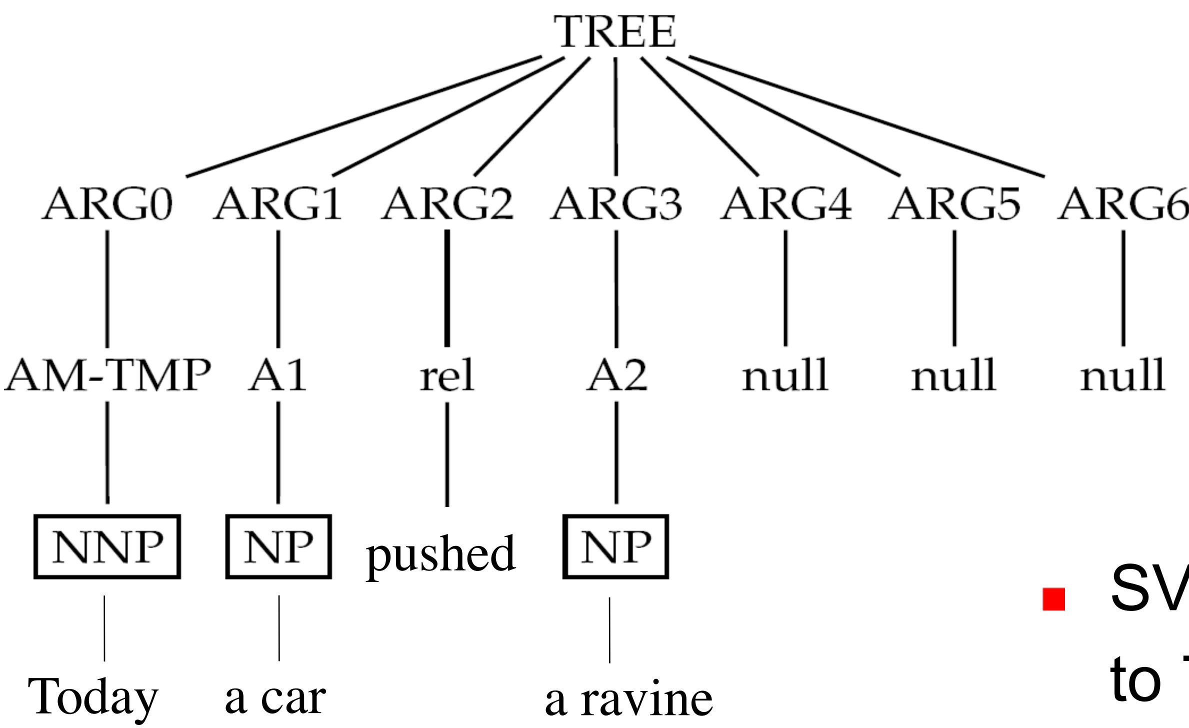
Reranking for Named-Entity Recognition
[Vien et al, 2010]



- CRF F1 from 84.86 to 88.16
- Best Italian system F1 82, improved to 84.33

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

Relational Kernels

Recognizing Textual Entailment

<u>learning</u> **textual entailment recognition rules** from annotated examples

… the textual entailment recognition task:

determine whether or not a text T implies a hypothesis H

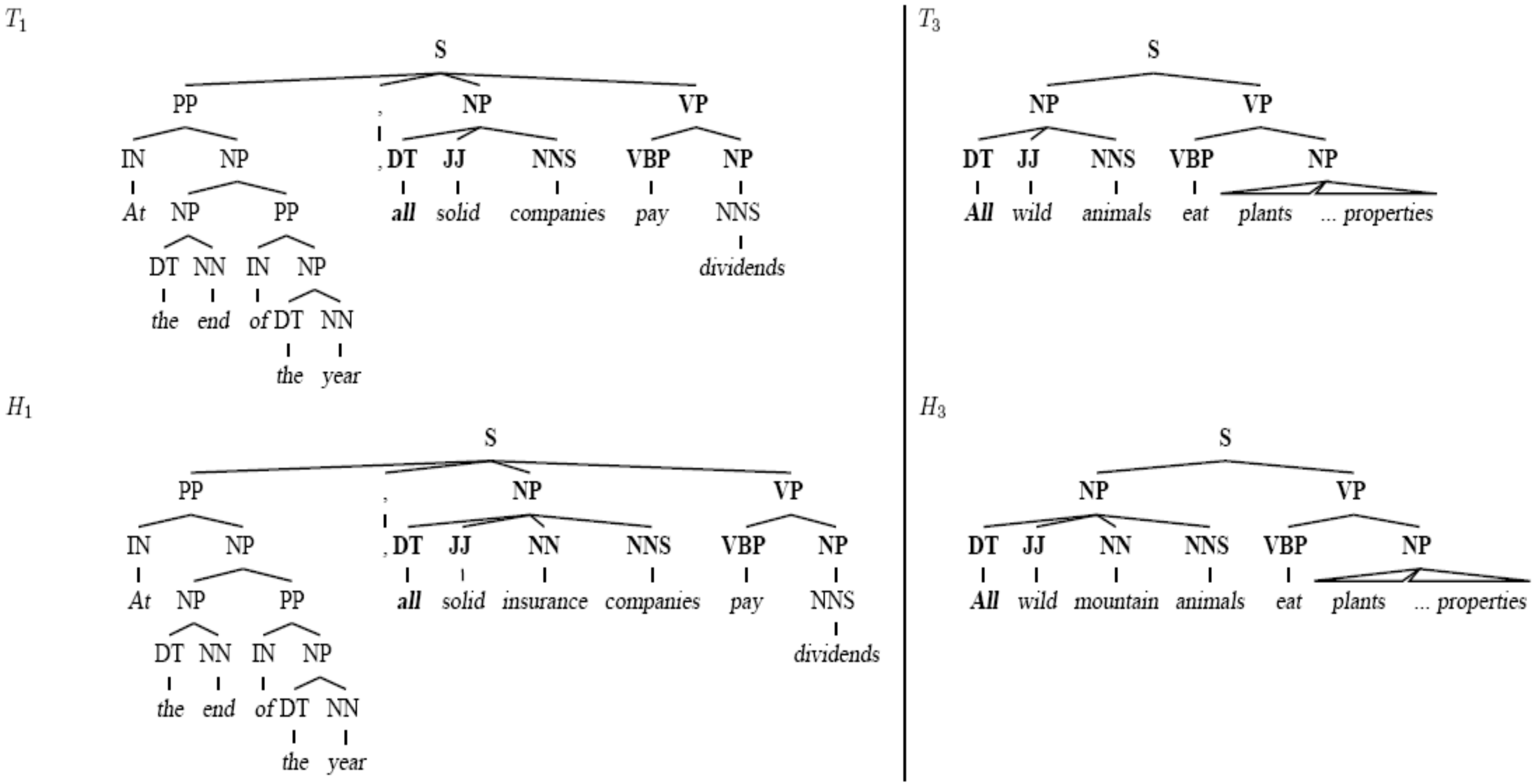
$T_1 \Rightarrow H_1$

T_1	*"At the end of the year, all solid companies pay dividends."*
H_1	*"At the end of the year, all solid insurance companies pay dividends."*

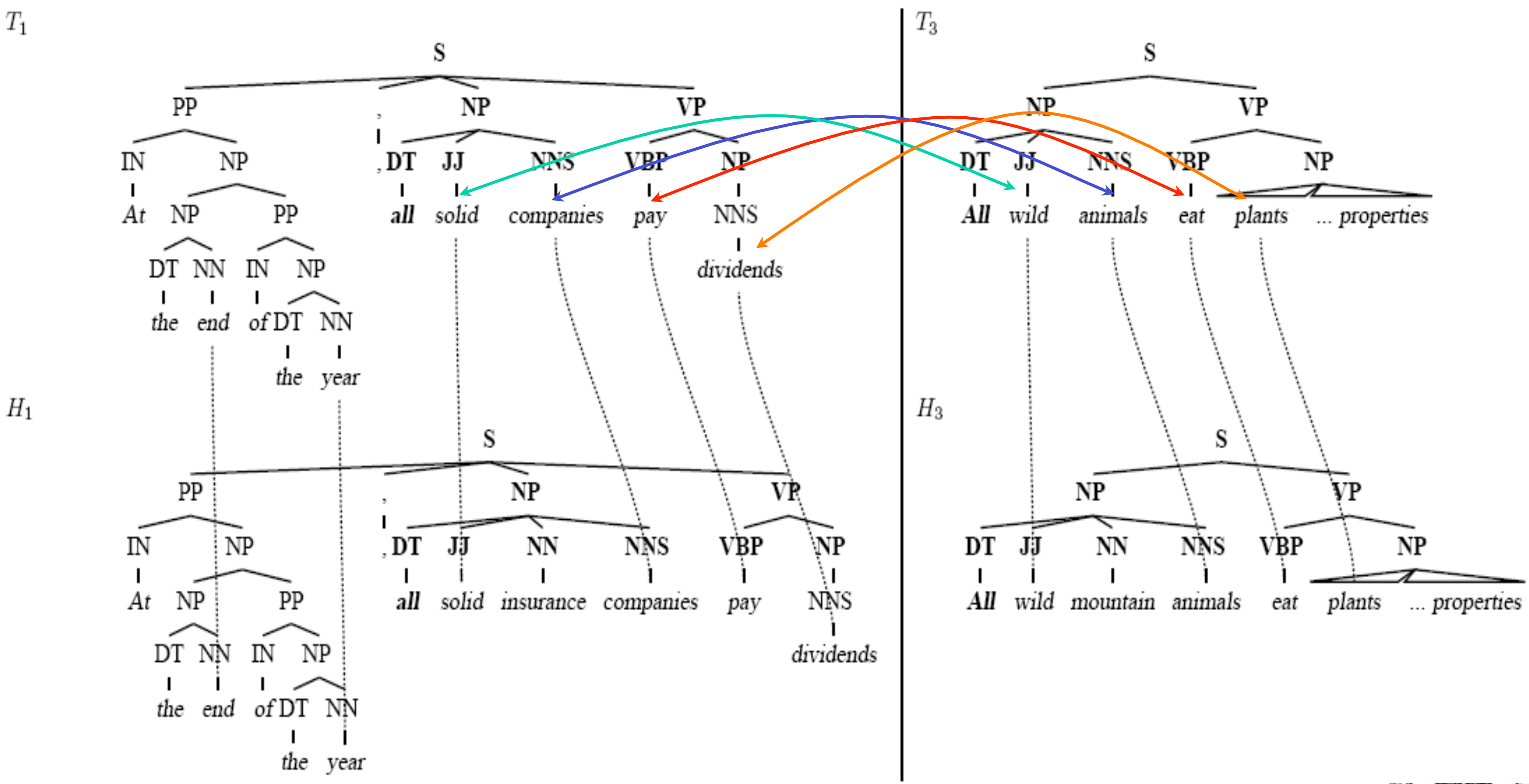
"Traditional" machine learning approaches:

similarity-based methods → distance in feature spaces

Determine Intra-pair links



Determine cross pair links



Our Model (Zanzotto and Moschitti, ACL2006)

Defining a similarity between pairs based on:

$$K_{ent}((T',H'),(T'',H''))=$$

$$K_I((T',H'),(T'',H''))+K_S((T',H'),(T'',H''))$$

- Intra-pair similarity

$$K_I((T',H'),(T'',H''))=TK(T',H')\times TK(T'',H'')$$

- Cross-pair similarity

$$K_S((T',H'),(T'',H''))\approx TK(T',T'')+TK(H',H'')$$

The final kernel

$$K_s((T', H'), (T'', H'')) =$$

$$\max_{c \in C} \Big(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i)) \Big)$$

where:

- ***c*** is an assignment of placeholders
- ***t*** transforms the trees according to the assigned placeholders

Experimental Results

- RTE1 (1st Recognising Textual Entailment Challenge) [Dagan et al., 2005]
  - 567 training and 800 test examples
- RTE2, [Bar Haim et al., 2006]
  - 800 training and 800 test examples

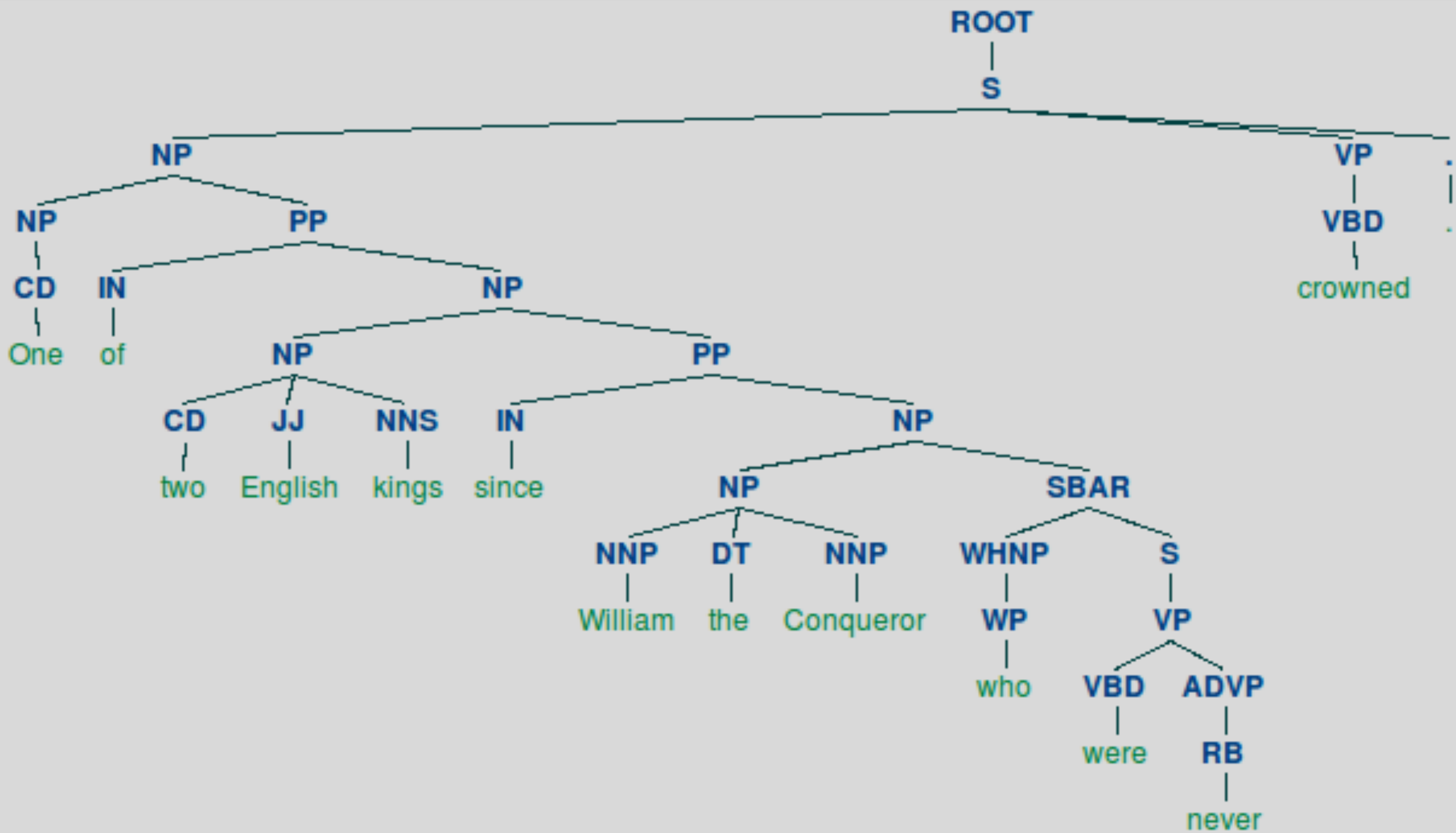
	BOW+LS	+ *TK*	+ K_{ent}	System Avg.
RTE1	0.5888	0.6213	0.6300	0.54
RTE2	0.6038	0.6238	0.6388	0.59

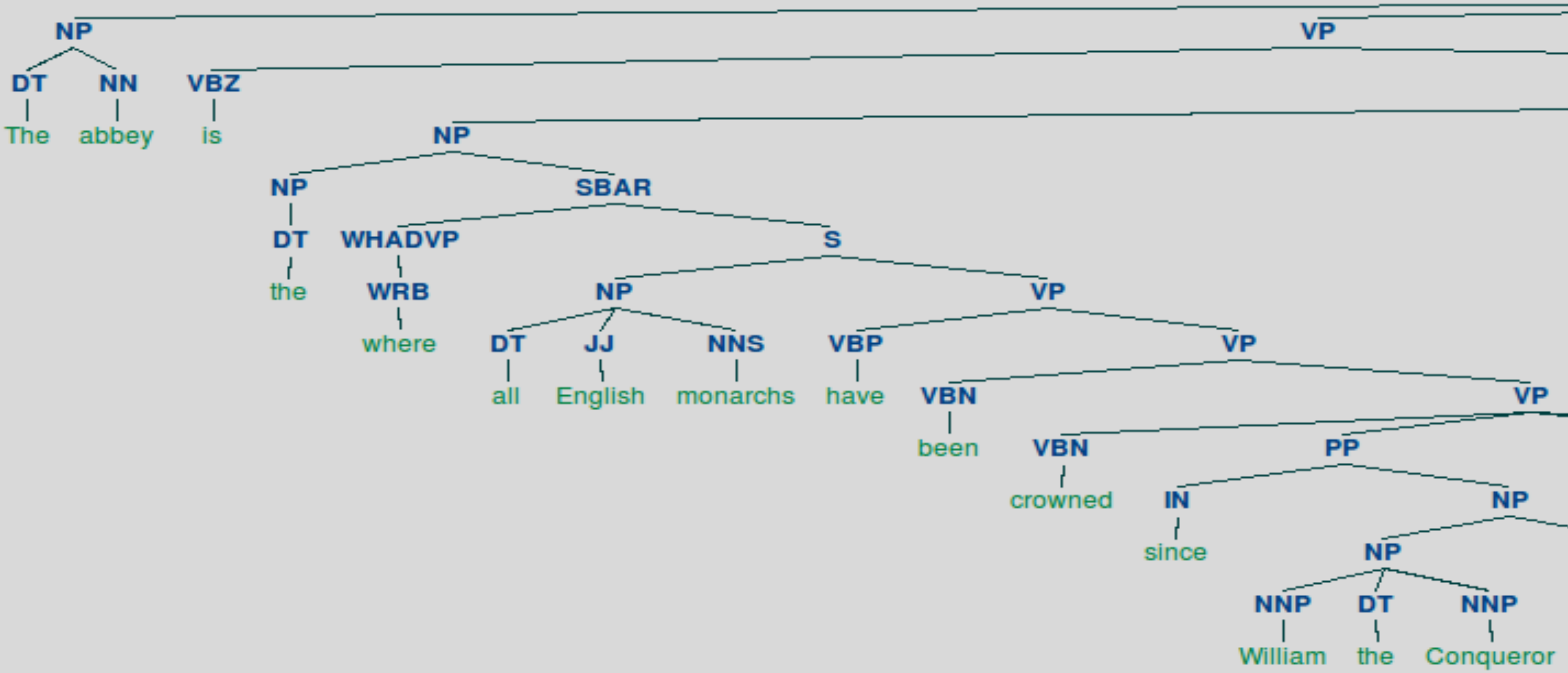
System	Strategy	Decision	An. Level	Knowledge Resources	Acc.
(Hickl et al., 2006)	lex,syn,trg	mlr	lxs,synt	WN,paraph,PropBank	0.7538
(Tatu and Moldovan, 2006)	lex	thr,inf	sur,sem	WN,SUMO,ExtWN,axioms	0.7375
(Zanzotto et al., 2006)	syn	mlr	lxs,syn	WN	0.6388
(Adams, 2006)	lex	mlr	sur,lxs	WN	0.6262
(Bos and Markert, 2006)	lex	mlr,inf	sur,lxs	WN,axioms	0.6160
(Kouylekov and Magnini, 2006)	synt	thr,mlr	lxs,syn	WN,DIRT	0.6050
(MacCartney et al., 2006)	synt	mlr	lxs,syn	WN	0.6050
(Snow et al., 2006)	trg,lex	rul,mlr	lxs,syn	WN,MindNet, thes	0.6025
(Herrera et al., 2006)	lex,syn	mlr	lex,syn	WN	0.5975
(Nielsen et al., 2006)	lex,syn	mlr	sur,syn		0.5960
(Marsi et al., 2006)	syn	thr	lxs,syn	WN	0.5960
(Katrenko and Adriaans, 2006)	lex,syn	mlr	syn		0.5900
(Burchardt and Frank, 2006)	syn	mlr	lxs,syn	WN,FrameNet,SUMO	0.5900
(Rus, 2006)	syn	thr	lxs,syn	WN	0.5900
(Litkowski, 2006)	lex	thr	sur		0.5810
(Inkpen et al., 2006)	trg,lex	mlr	lxs,syn	WN	0.5800
(Ferrndez et al., 2006)	syn	thr	lex,syn	WN	0.5563
(Schilder and McInnes, 2006)	lex,syn	mlr	lxs,syn	WN	0.5550

Relational Kernels for Answer Reranking


JEOPARDY!

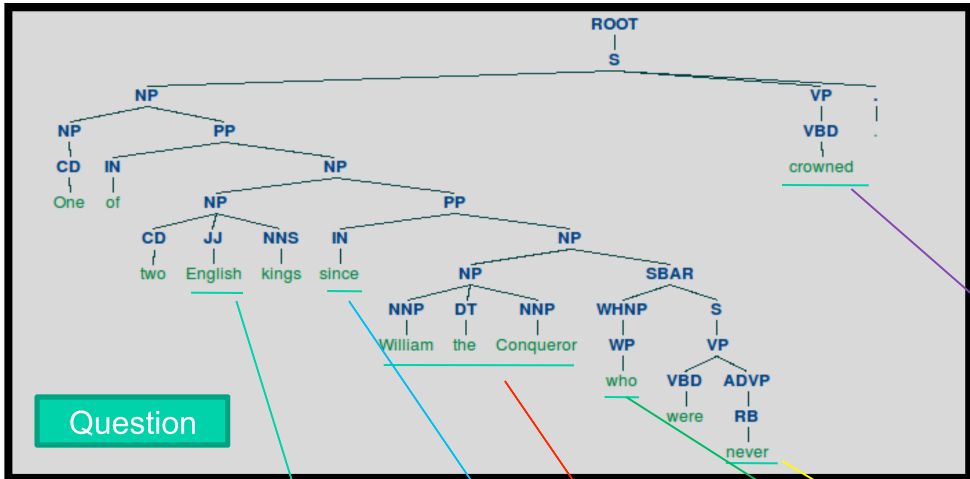
An example of Jeopardy! Question





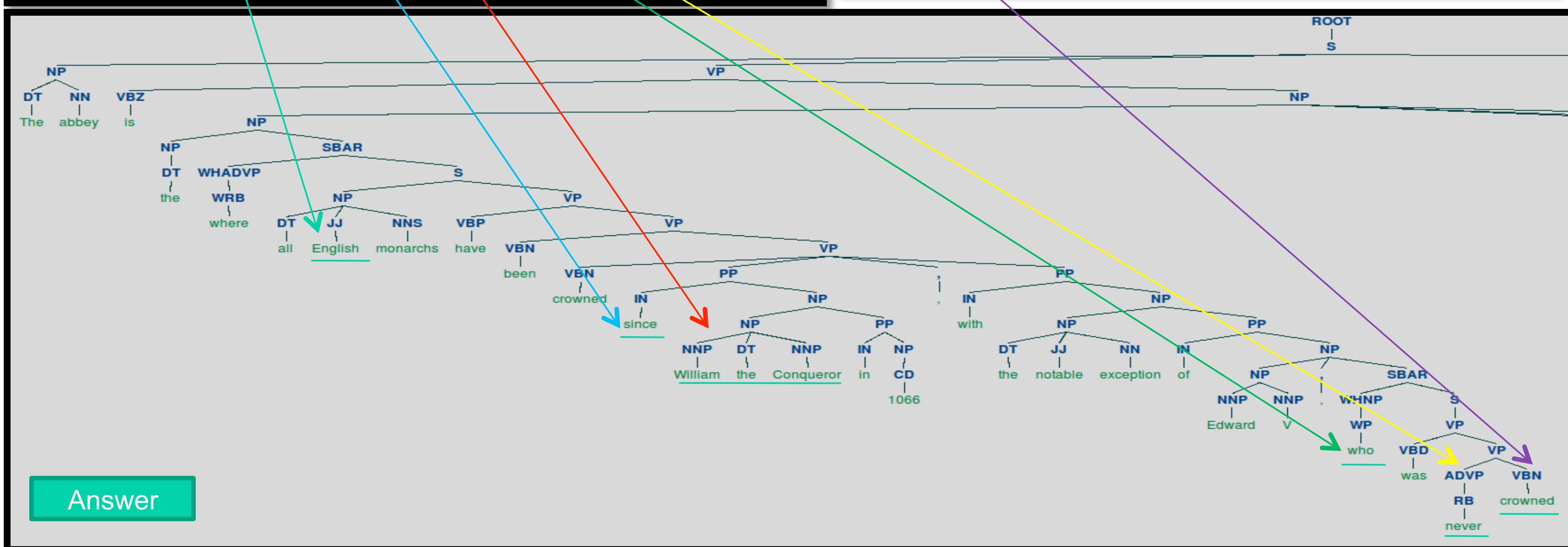
NP
VP
DT
NN
VBZ
The
abbey
is
NP
NP
SBAR
DT
WHADVP
S
the
WRB
NP
VP
where
DT
JJ
NNS
VBP
VP
all
English
monarchs
have
VBN
VP
been
VBN
PP
crowned
IN
NP
since
NP
NNP
DT
NNP
William
the
Conqueror

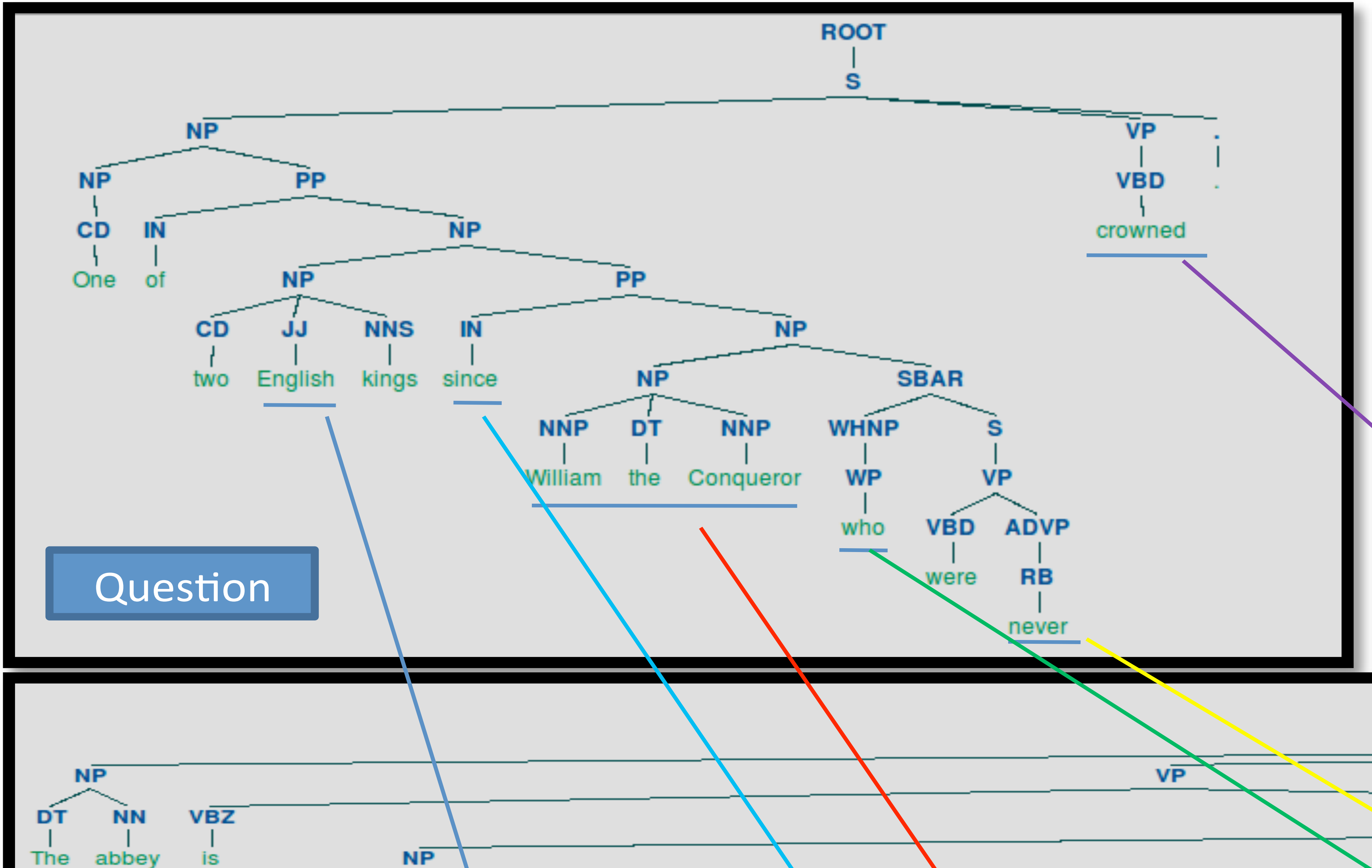
Baseline Model



Methodology:

1-Applying PTK without any extra annotation and evaluate the model as baseline.




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

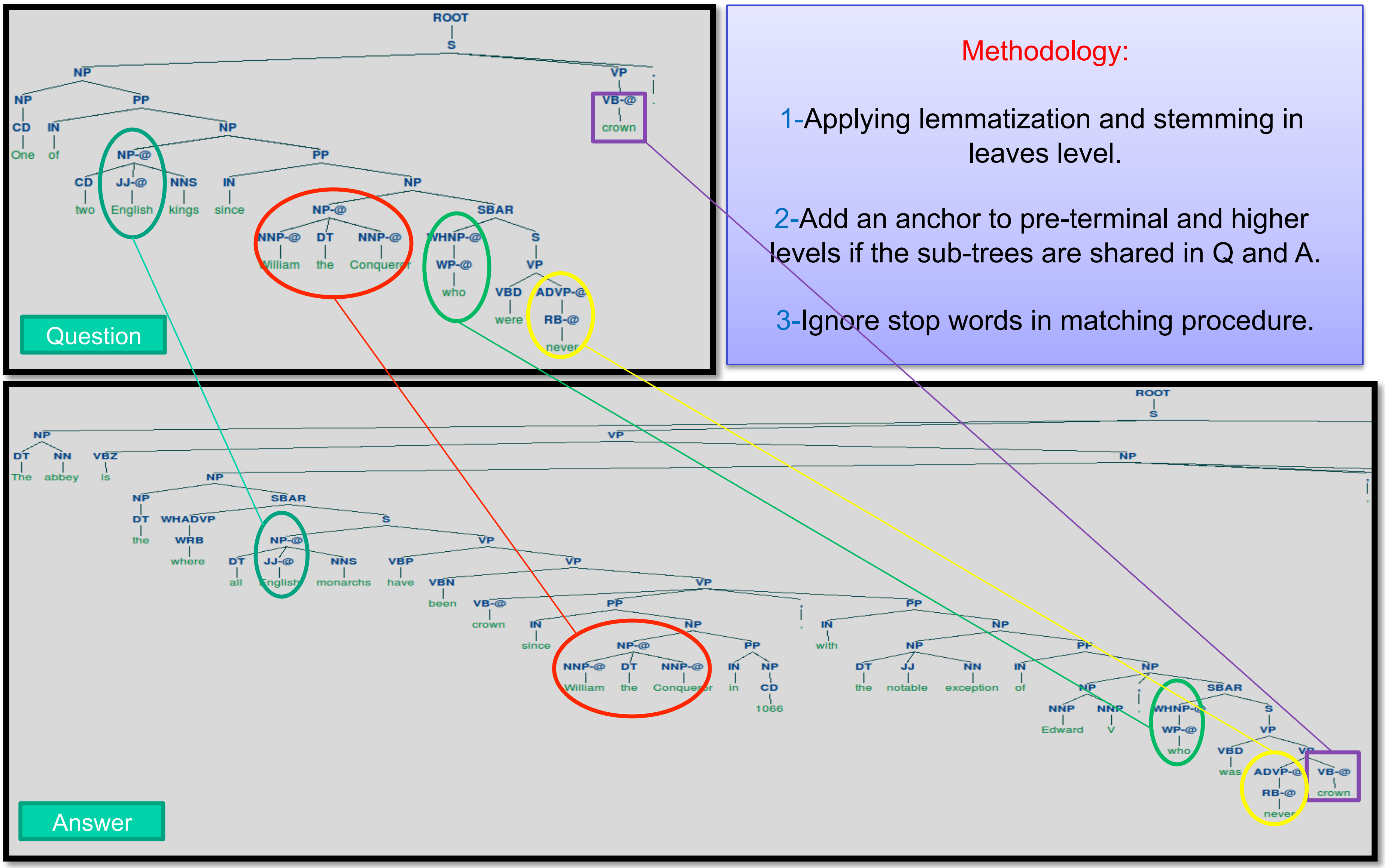
Best Model

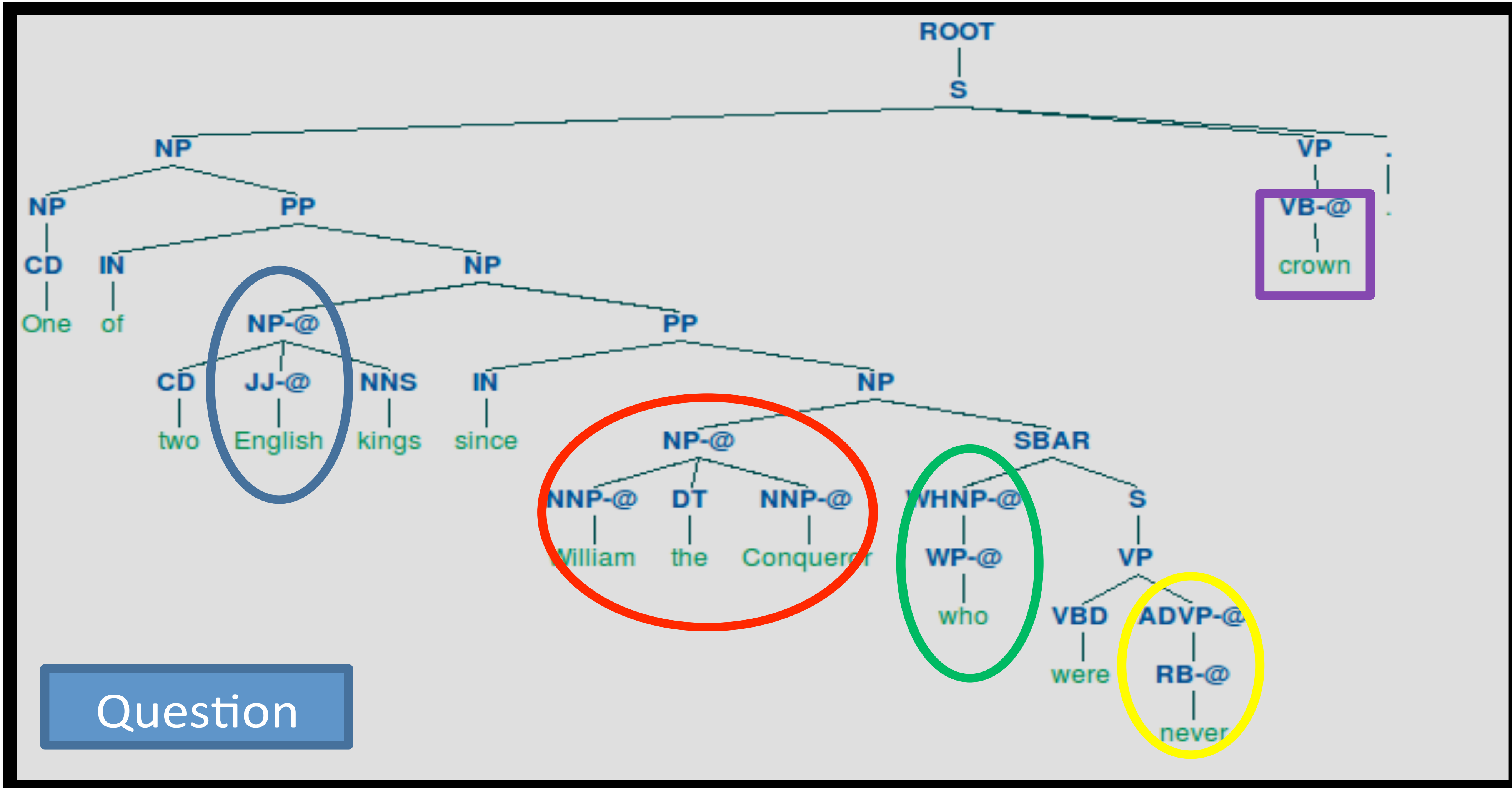
Methodology:

1-Applying lemmatization and stemming in leaves level.

2-Add an anchor to pre-terminal and higher levels if the sub-trees are shared in Q and A.

3-Ignore stop words in 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 similarity and the learning algorithm looses some of its power

Running example (randomly picked Q/A pair from Answerbag)

Question: Is movie theater popcorn vegan?

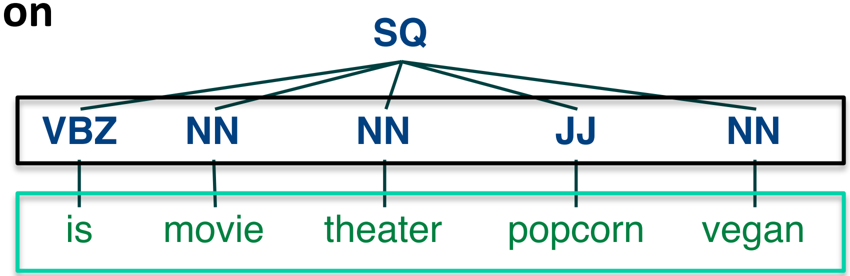
Answer:

(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, SIGIR2012]

Question

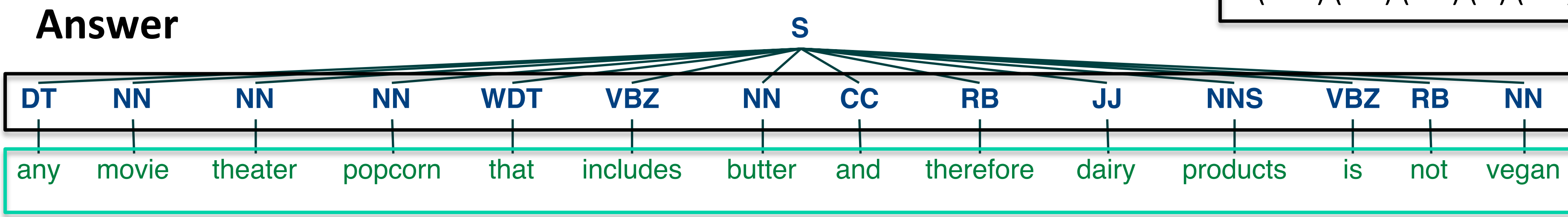


bag of pos tags
bag of words
and their combination

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

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

Answer

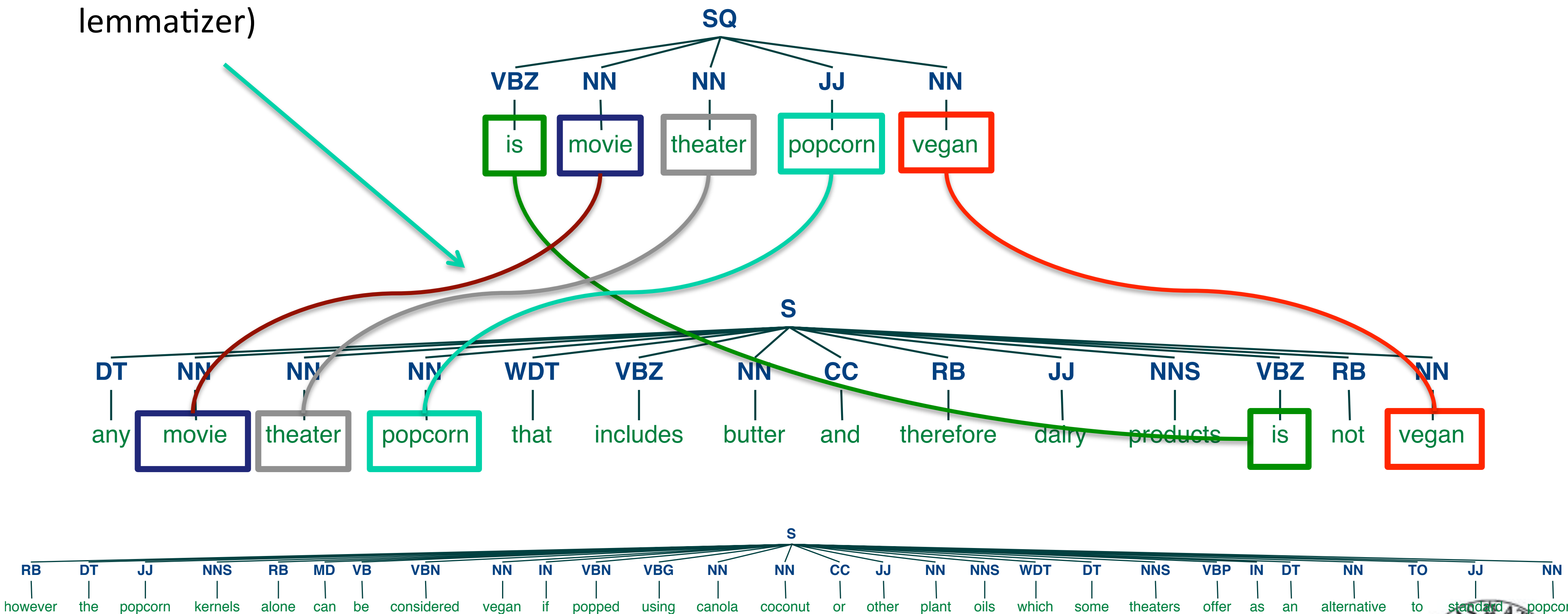


(any) (movie) (theater) (popcorn) (that) (includes) (butter) (and) (therefore) (dairy) (products) (is) (not) (vegan)

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

Linking question with the answer 01

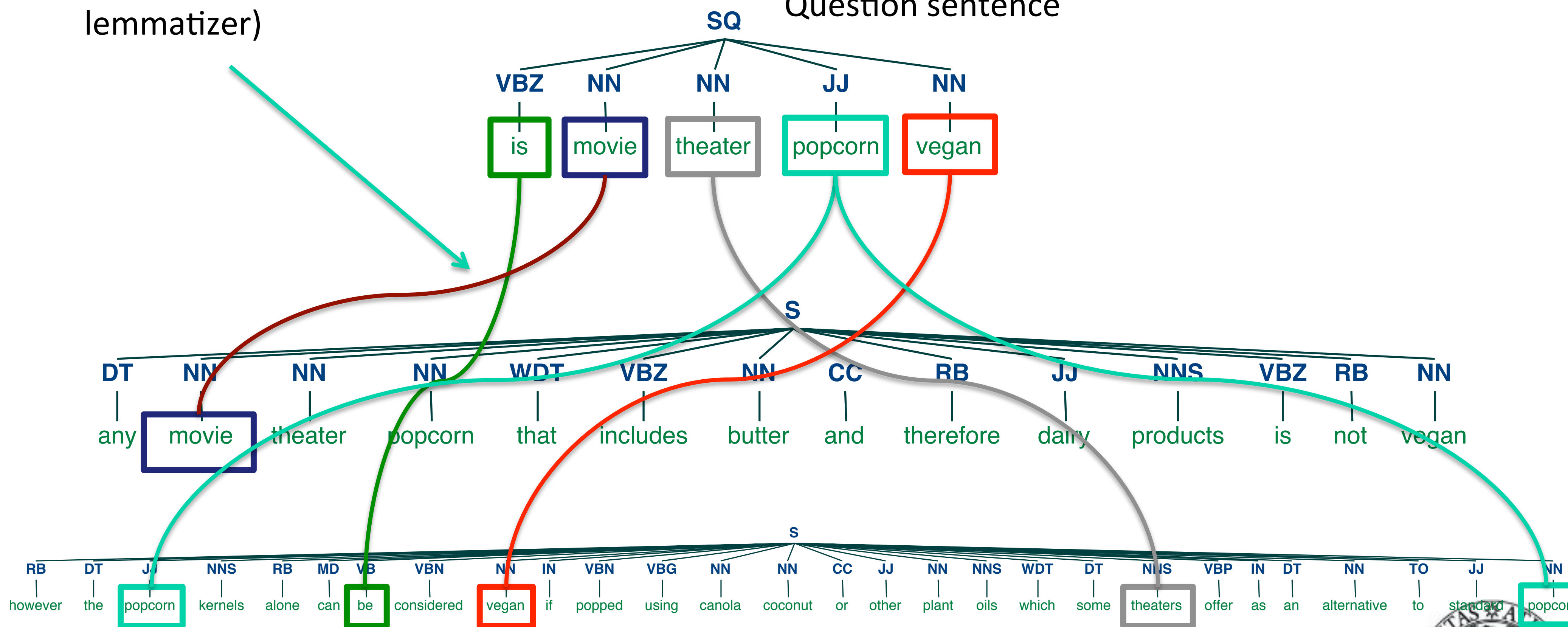
Lexical matching is on word lemmas (using WordNet lemmatizer)



Linking question with the answer 02

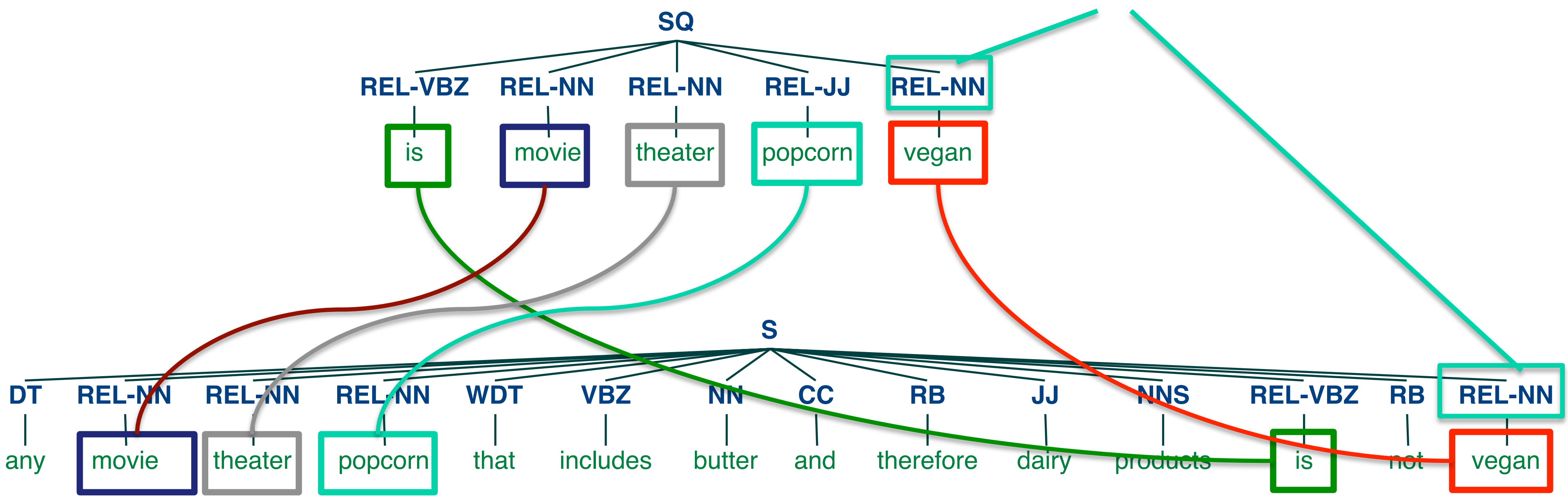
Lexical matching is on word lemmas (using WordNet lemmatizer)

Question sentence



Linking question with the answer: relational tag

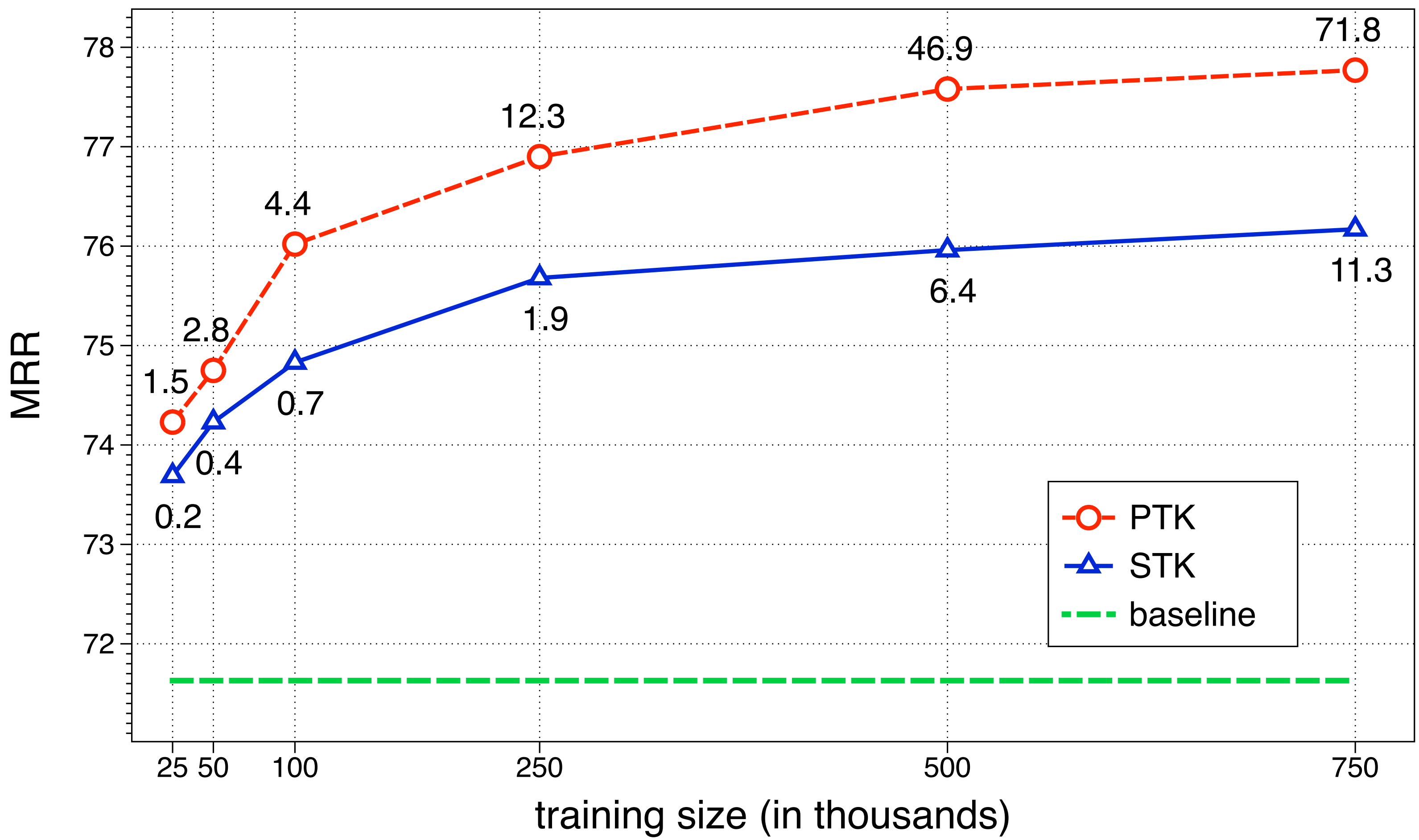
Marking pos tags of the aligned words by a relational tag: "REL"



Answerbag data

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

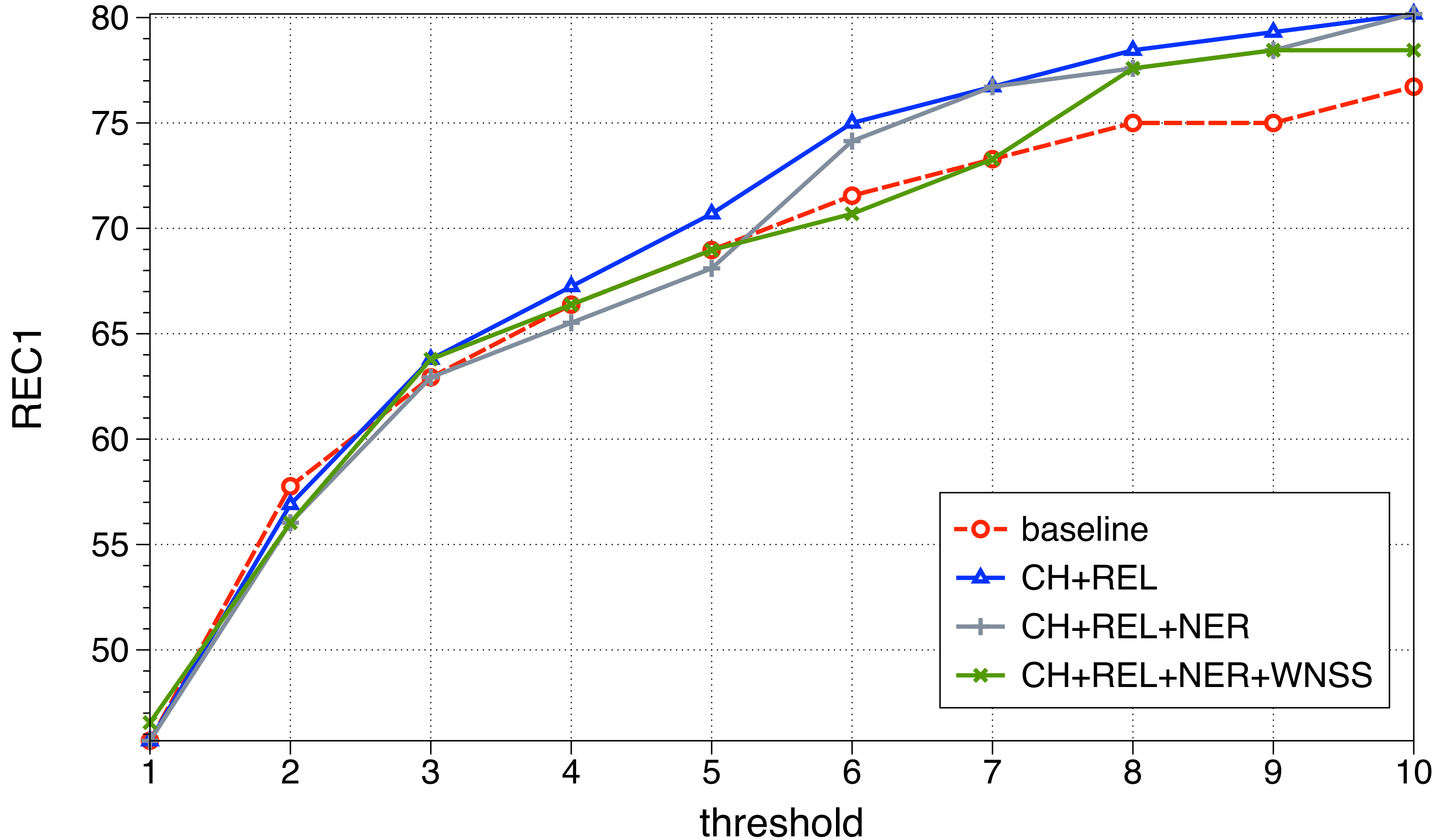
Learning Curve for Answerbag



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
- Each relevant passage is paired with each irrelevant
- Split the data:
  - train 70% (259 questions): 63361 examples for re-ranker
  - test 30% (116 question): 5706 examples for re-ranker

Jeopardy! data



Part II: Advanced Topics

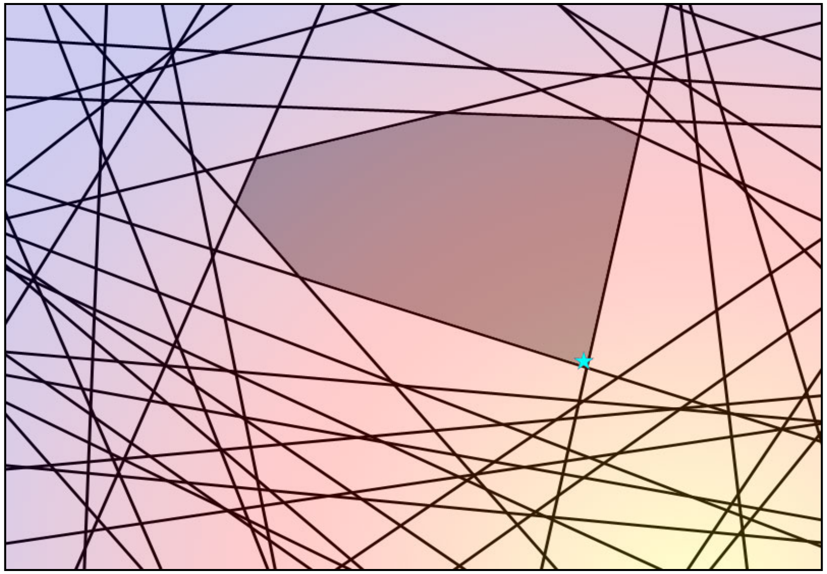
Efficiency Issue

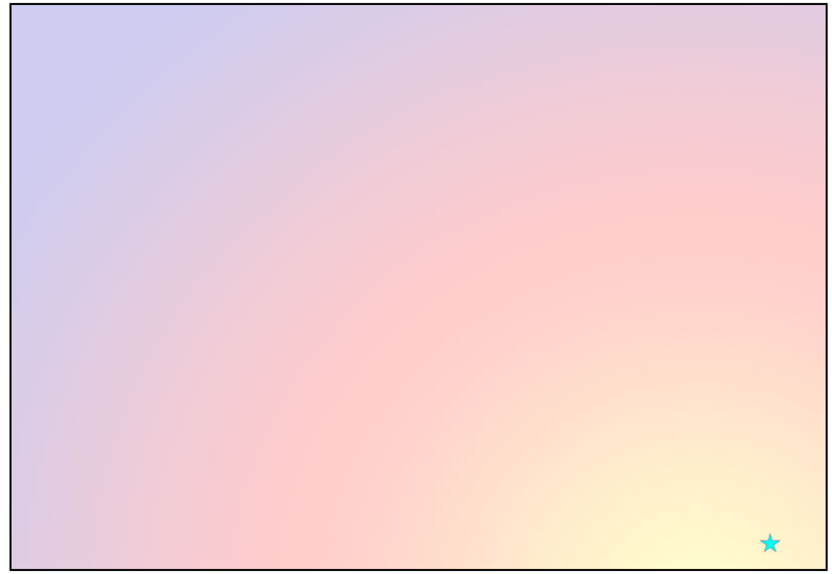
- Working in dual space with SVMs implies quadratic complexity
- Our solutions:
  - cutting-plane algorithm with sampling uSVMs

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

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

CPA in a nutshell





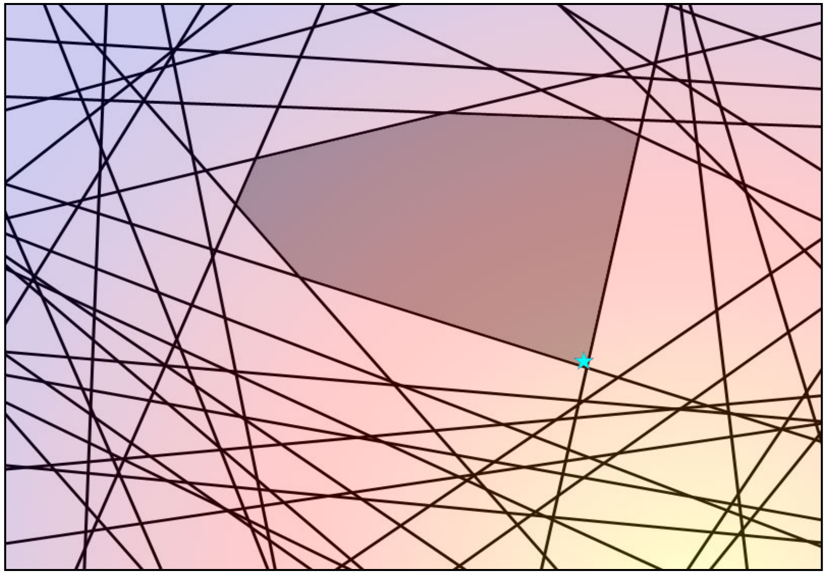
Original SVM Problem

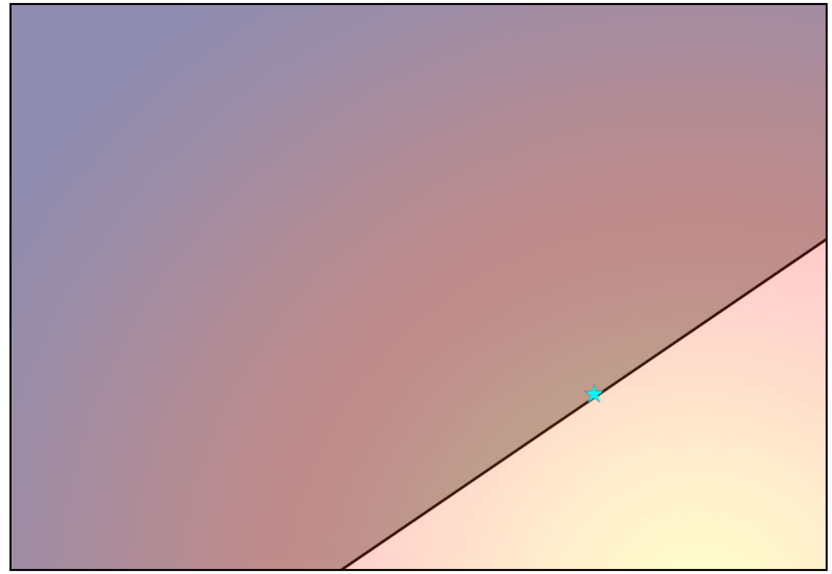
- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

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

CPA in a nutshell





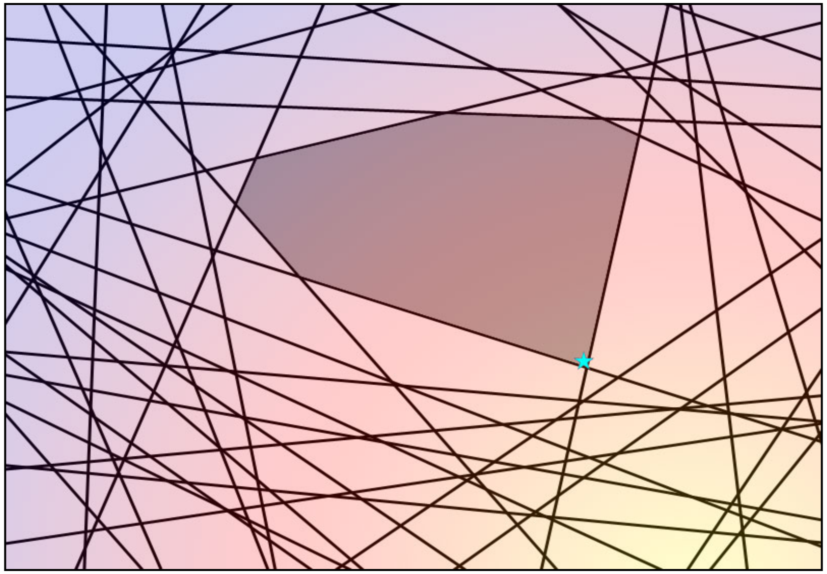
Original SVM Problem

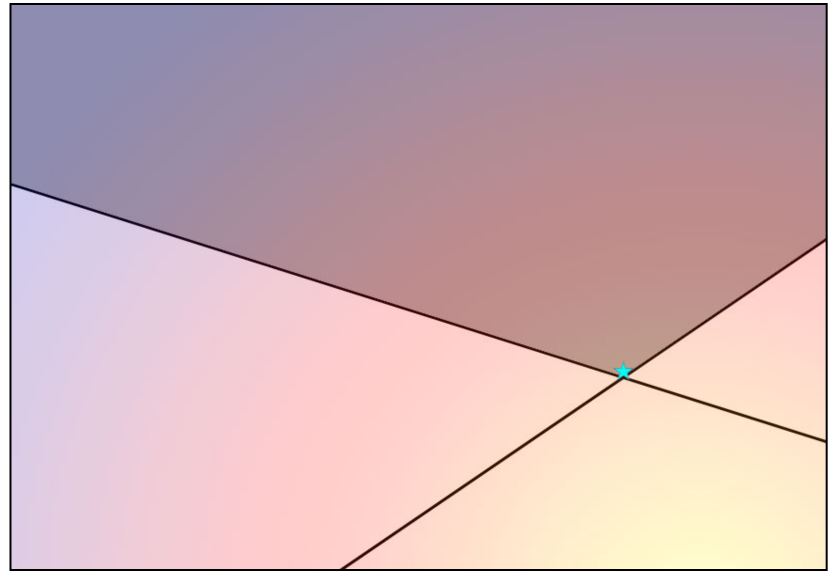
- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

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

CPA in a nutshell





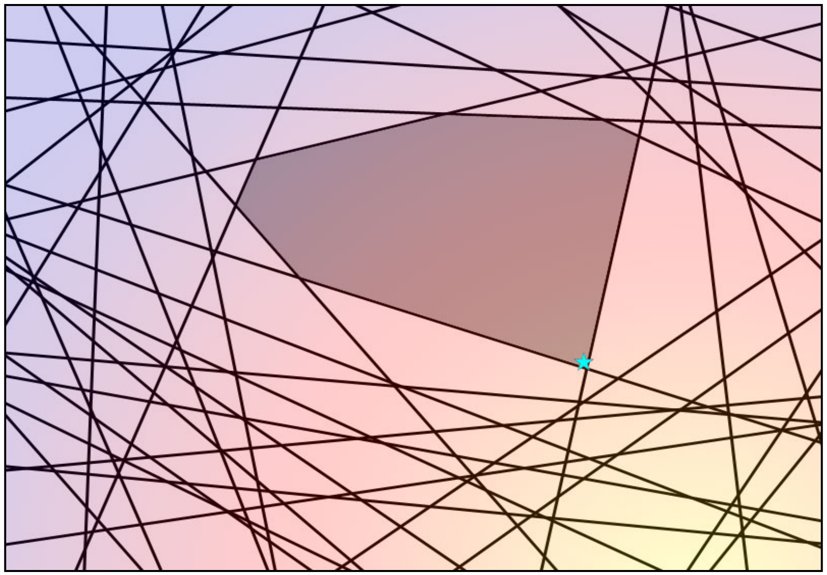
Original SVM Problem

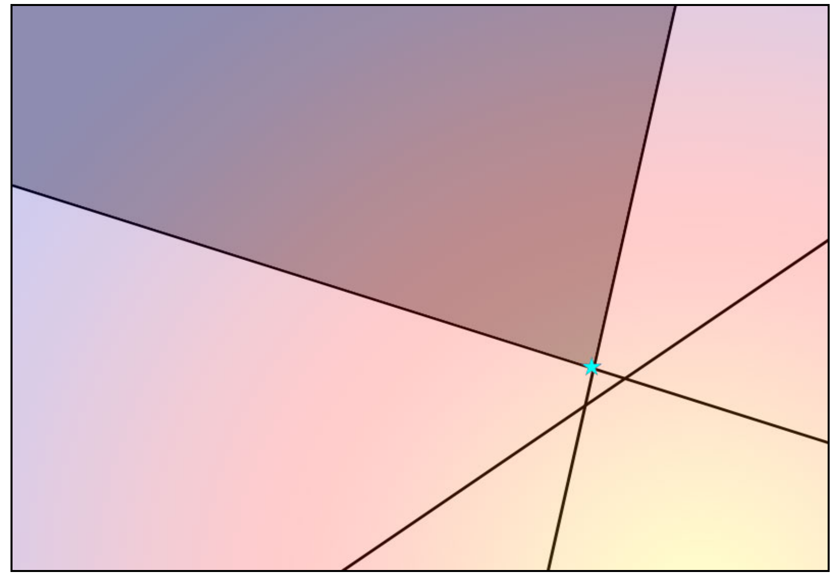
- Exponential constraints
- Most are dominated by a small set of "important" constraints

CPA SVM Approach

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

CPA in a nutshell





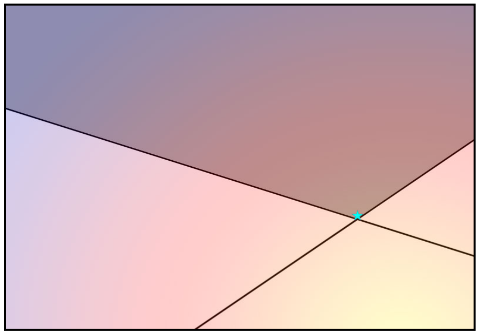
Original SVM Problem

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

CPA SVM Approach

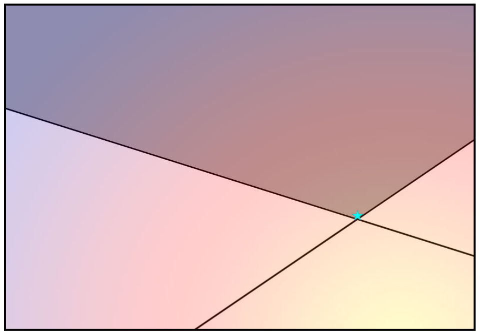
- Repeatedly finds the next most violated constraint…
- …until set of constraints is a good approximation.

Computing most violated constraint (MVC)



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

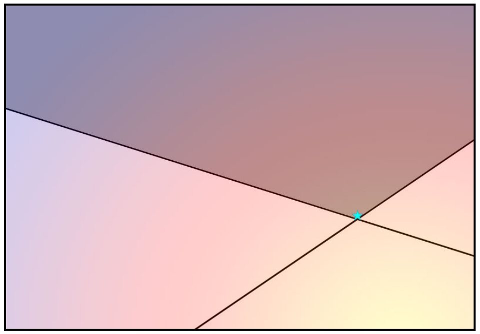
Computing most violated constraint (MVC)



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

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

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)

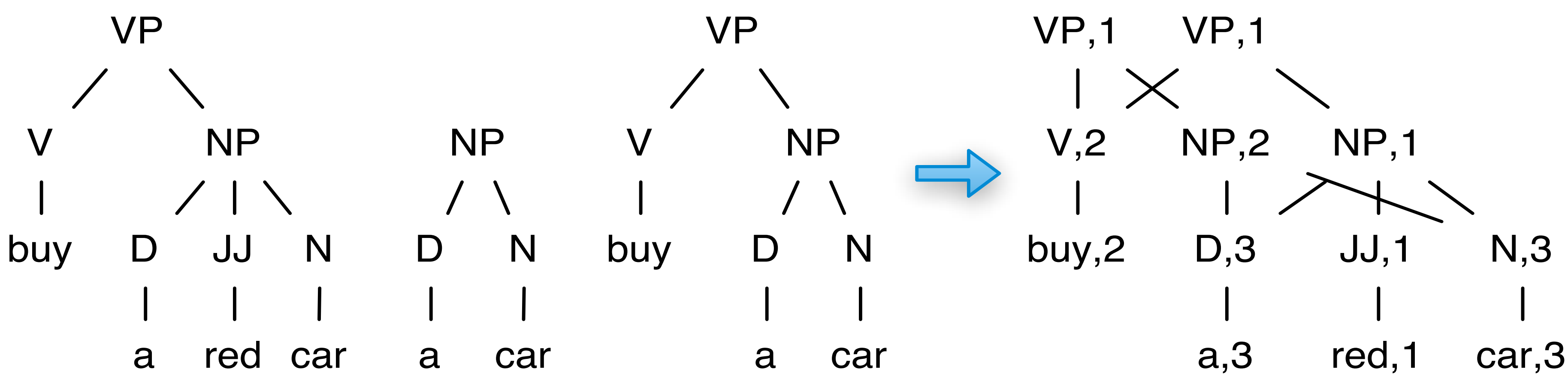
- Main bottleneck to apply kernels comes from the inner product:

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

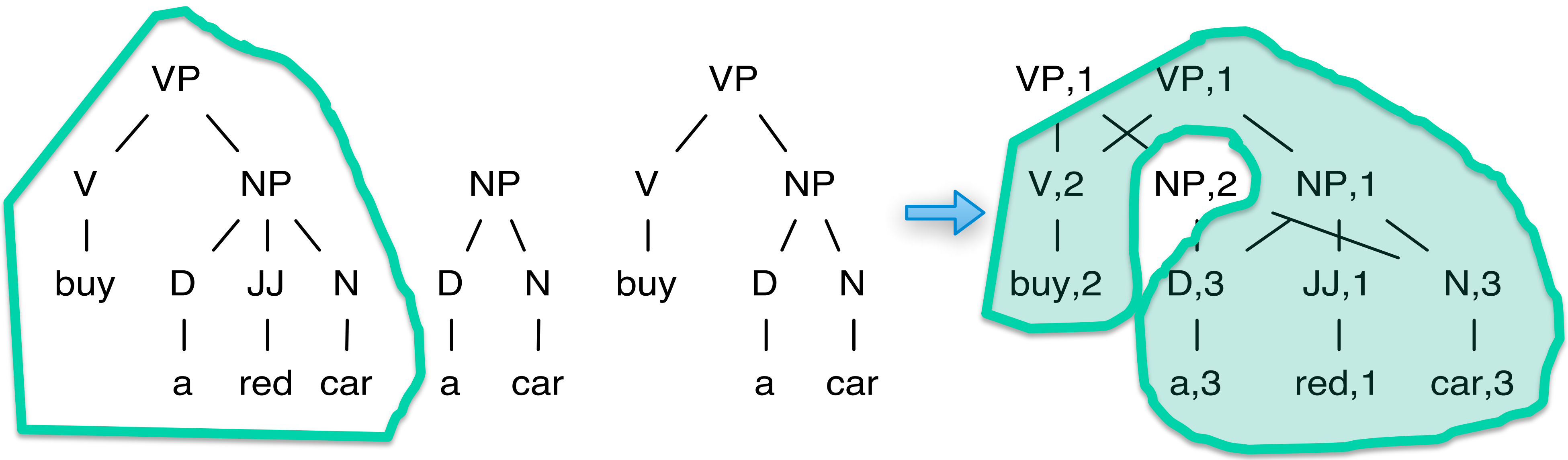
- Use sampling to approximate exact cutting plane models

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

Three syntactic trees and the resulting DAG



Three syntactic trees and the resulting DAG



SDAG

- Compacts each CPA model into a single DAG

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

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

SDAG+

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

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

$$\Downarrow$$

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

Reverse Kernel Engineering

- **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)

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

- 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):

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

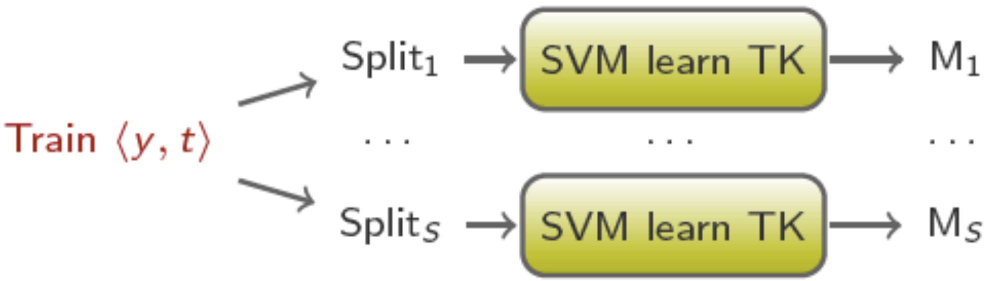
where:

- 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

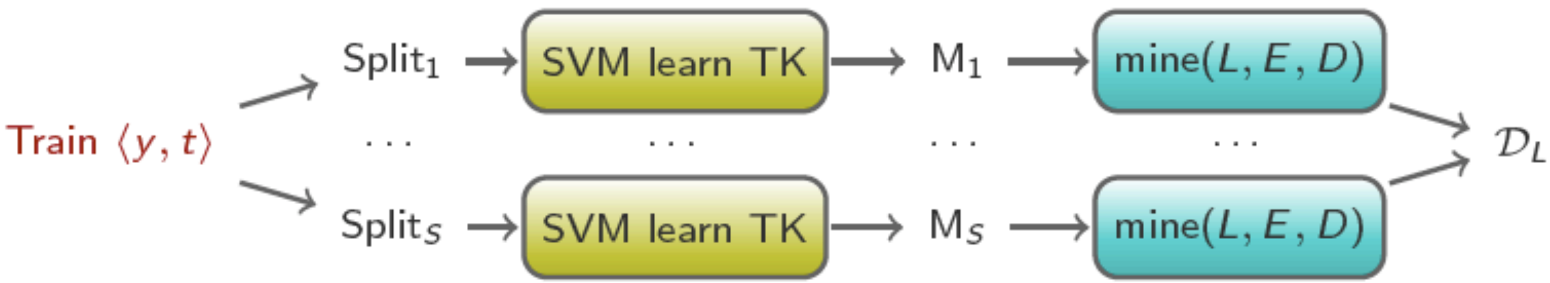
$$\text{Train } \langle y, t \rangle \quad \begin{array}{l} \nearrow \ \text{Split}_1 \\ \quad \cdots \\ \searrow \ \text{Split}_S \end{array}$$

Reverse Engineering Framework



FSL = Fragment Space Learning

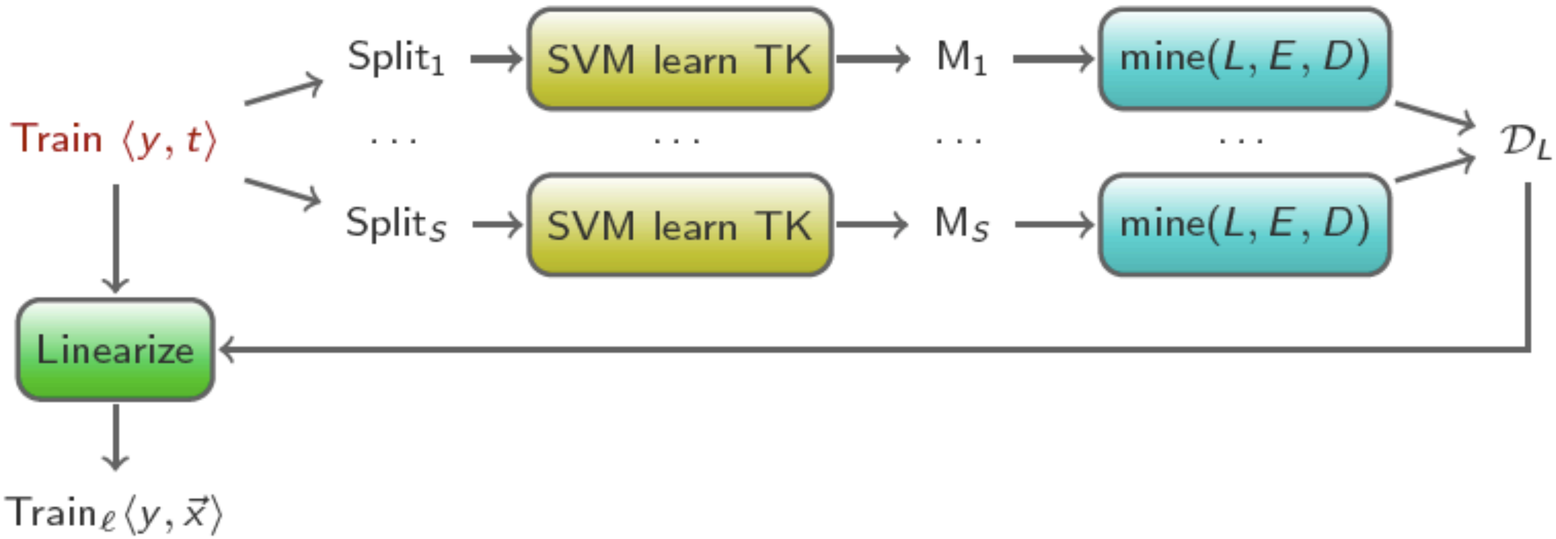
Reverse Engineering Framework

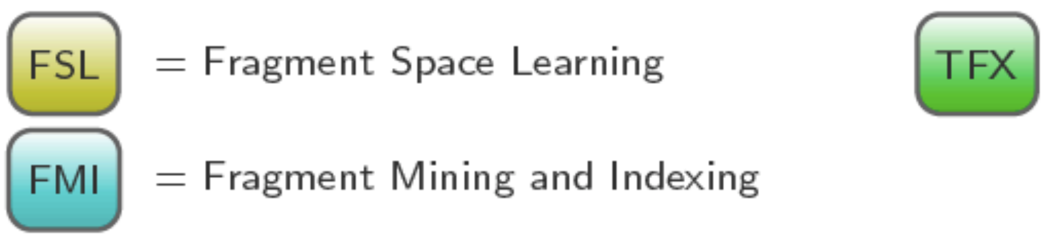


FSL = Fragment Space Learning

FMI = Fragment Mining and Indexing

Reverse Engineering Framework



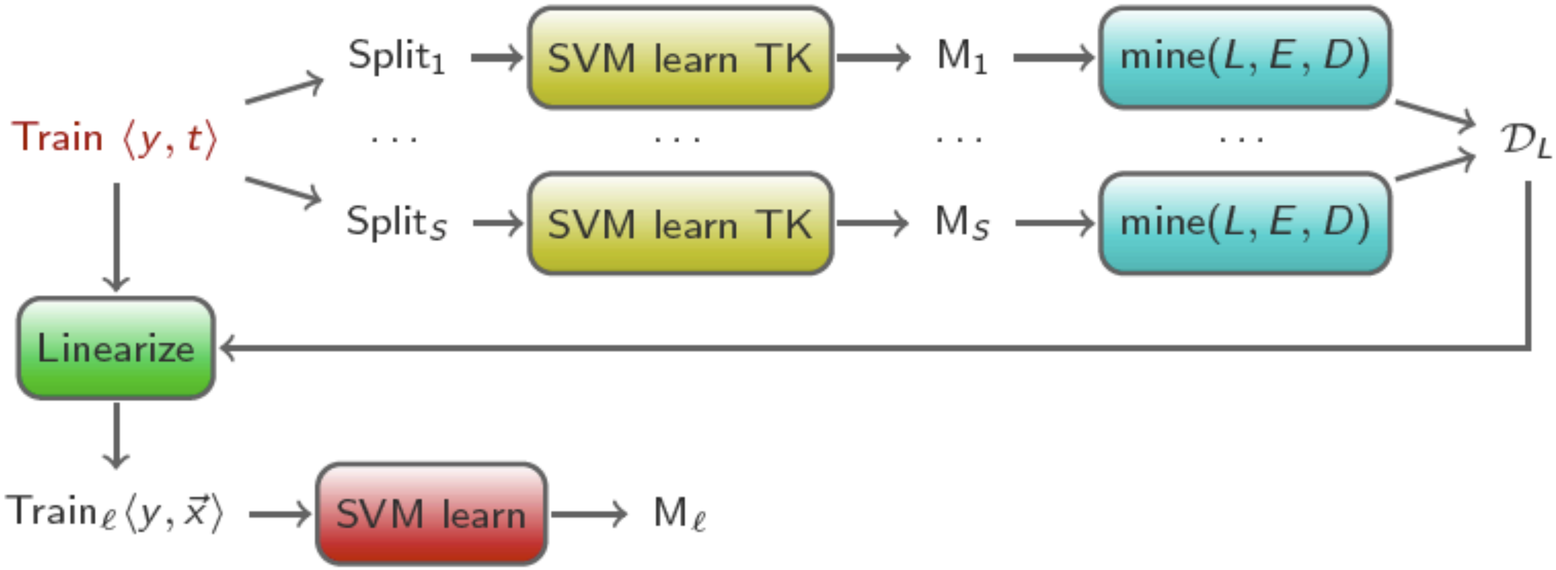


FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

FMI = Fragment Mining and Indexing

Reverse Engineering Framework



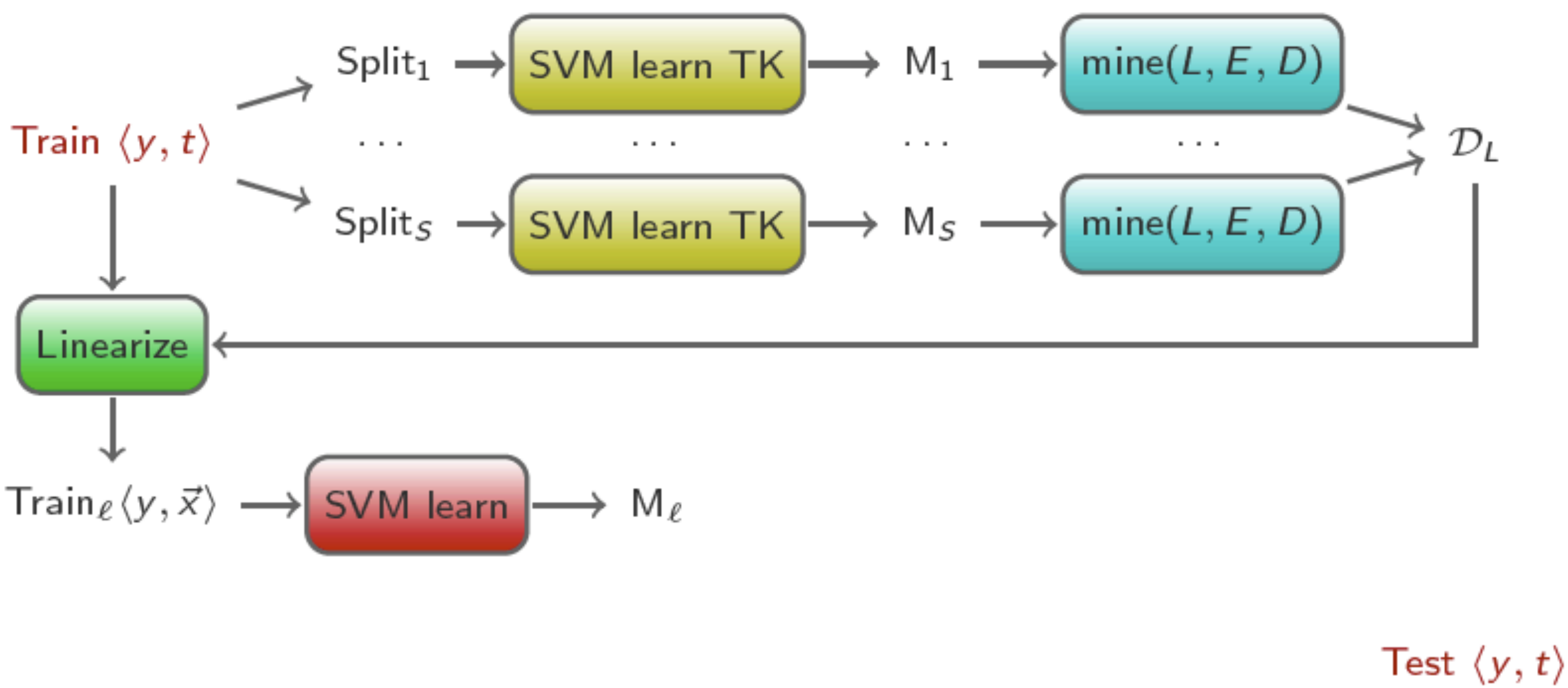
FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

FMI = Fragment Mining and Indexing

ESL = Explicit Space Learning

Reverse Engineering Framework



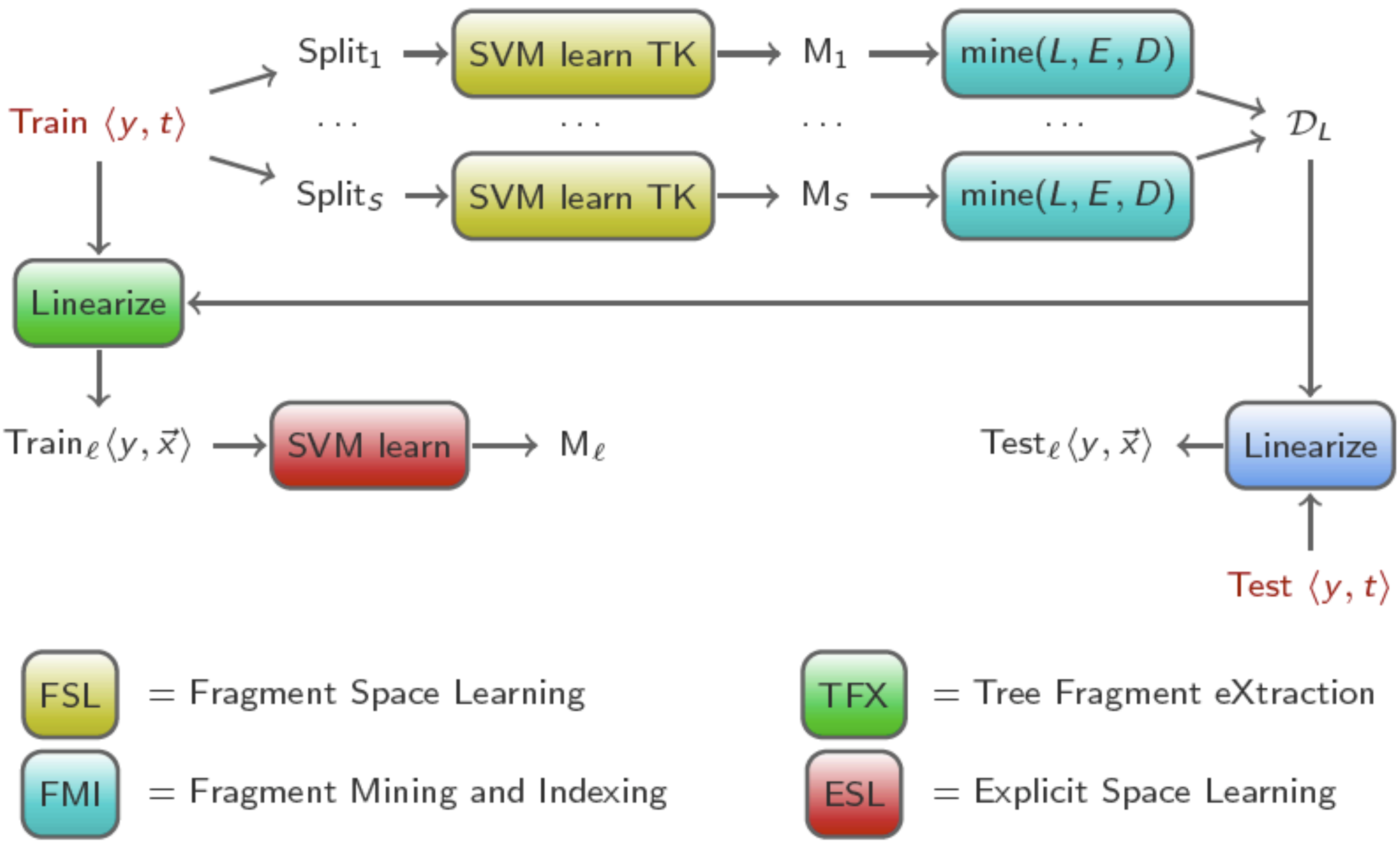
FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

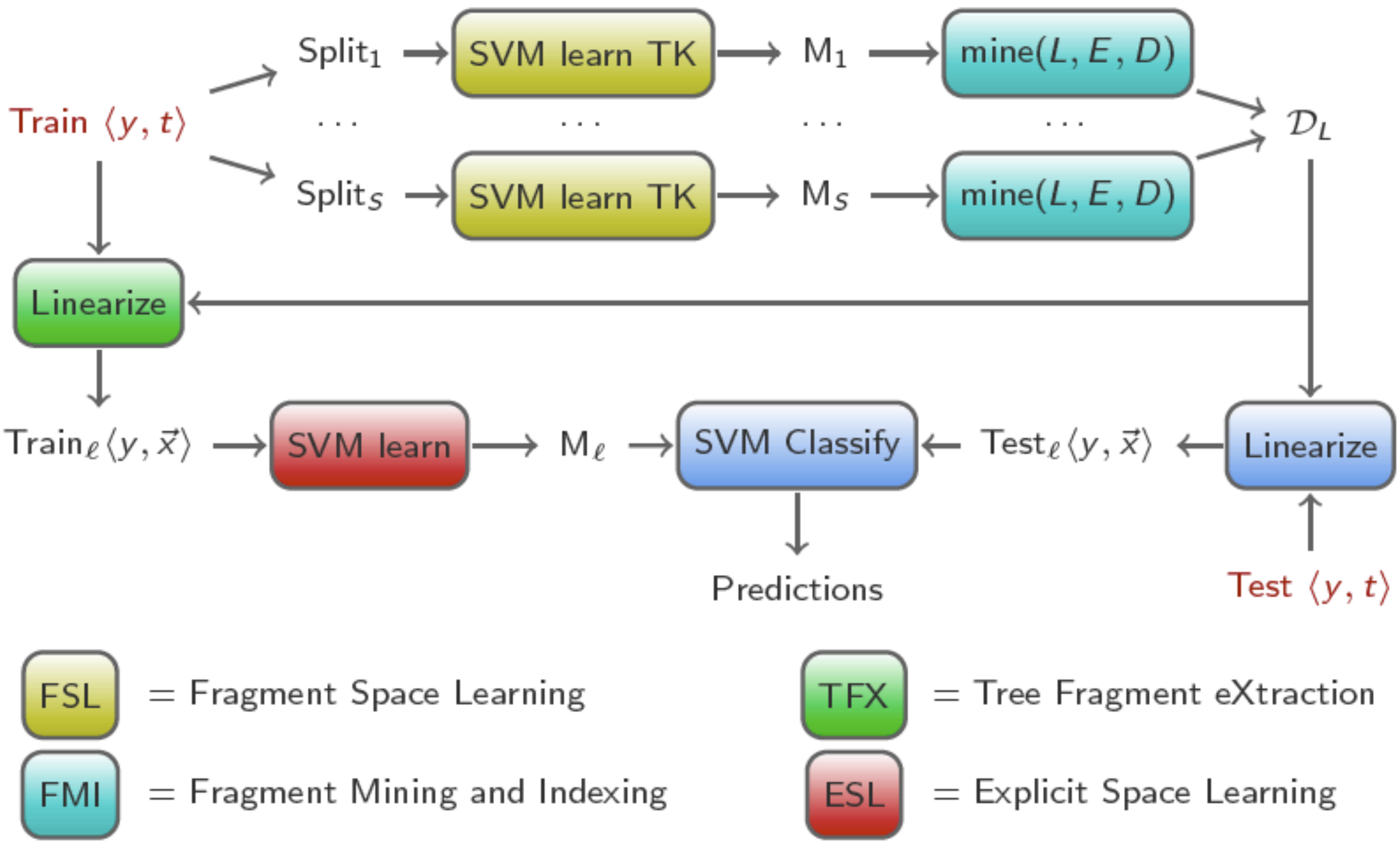
FMI = Fragment Mining and Indexing

ESL = Explicit Space Learning

Reverse Engineering Framework



Reverse Engineering Framework



Semantic Role Labeling

Setting

BC/BC_ℓ:

- training: 1 Mil AST_ms from PB secs 2-6
- test: 149,140 AST_ms from PB sec 24

RM/RM_ℓ:

- training: 179,091 core arg AST_ms (A0, A1, . . . A5) from PB secs 2-21
- test: 5,928 core arg AST_m from PB sec 24

SST_ℓ configuration:

FSL SVM-Light-TK, normalized SST, $\lambda = 0.4$ (default), $S = 50$

FMI $L = 50.000$ (threshold), $E = 1$ (maxexp)

ESL SVM-Light-TK, linear kernel

SST configuration: SVM-Light-TK, normalized SST, $\lambda = 0.4$ (default)

Results

About 10 time faster -Training (and testing)

Parallelizable!

Class	Data set Tr^+	Te^+	Accuracy SST	SST_ℓ
BC	61,062	8,515	81.8	81.3
A0	60,900	2,014	91.6	91.1
A1	90,636	3,041	89.0	89.4
A2	21,291	697	73.1	73.0
A3	3,481	105	56.8	53.0
A4	2,713	69	69.1	67.9
A5	69	2	66.7	0.0
RM			87.8	87.8

Table: Number of positive training (Tr^+) and test (Te^+) examples in the SRL dataset. Accuracy of the non-linearized (SST) and linearized (SST_ℓ) binary classifiers (i.e. BC, A0, ...A5) is F_1 measure. Accuracy of RM is the percentage of correct class assignments.

(ADJP(RB-B)(VBN-P))
(NP(VBN-P)(NNS-B))
(S(NP-B)(VP))
(VP(VBD-P(said))(SBAR))
(VP(VB-P)(NP-B))
(NP(VBG-P)(NNS-B))
(VP(VBD-P)(NP-B))
(VP(VBG-P)(NP-B))
(VP(VBZ-P)(NP-B))
(VP(VBN-P)(NP-B))
(VP(VBP-P)(NP-B))
(NP(NP-B)(VP))
(NP(VBG-P)(NN-B))
(S(S(VP(VBG-P)))(NP-B))
(NP(PRP-B))
(VP(AUX-P)(NP-B))
(VP(VBG-B(going))(S))
(VP(MD-B)(VP))
(PP(VBG-P)(NP-B))

Table: Best fragments for SRL BC.

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_ℓ
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

- We used powerful ML algorithms
  - e.g., Support Vector Machines
  - Robust to noise
- Abstract representations of examples
  - Similarity functions (Kernel Methods)
  - Structural syntactic/semantic similarity
- Modeling NLP tasks with: advanced syntactic and shallow semantic structures and relational marker
- Experiments demonstrate the benefit of such approach

Conclusions (cont'd)

- Kernel methods and SVMs are useful tools to design language applications
- Basic general kernel functions can be used to engineer new kernels
- Little effort in selecting and marking/tailoring/decorating/designing trees or designing sequences
- Easy modeling produces state-of-the-art accuracy in many tasks, SRL, RE, CR, QA, NER, SLU, RTE
- Fast prototyping and model adaptation

Future (on going work)

- Deeper modeling of paragraphs: *shallow semantics and discourse structures*
- The objective is to design more compact and accurate models applicable to whole paragraphs.
- Use of reverse kernel engineering to study linguistic phenomena:
  - [Pighin&Moschitti, CoNLL2009, EMNLP2009, CoNLL2010]
  - To mine the most relevant fragments according to SVMs gradient
  - To use the linear space
- Experimenting with combined uSVMs and linearized models: learning on large-scale data

Structural Kernels at ACL 2012

- **Session 2E: (July 9, 14:00 - 15:30) Lexical semantics (Chair: Lillian Lee):**

  Verb Classification using Distributional Similarity in Syntactic and Semantic Structures, Danilo Croce, Alessandro Moschitti, Roberto Basili and Martha Palmer

- **Session 6D: (July 10, 16:00 -17:30) Topics (Chair: Tadashi Nomoto):**

  Modeling Topic Dependencies in Hierarchical Text Categorization, Alessandro Moschitti, Qi Ju and Richard Johansson

Structural Kernels at ACL 2012

- *String Re-writing Kernel*, F. Bu, H. Li, and X. Zhu

(best paper award)

- Poster session:S-27
- *Identifying High-Impact Sub-Structures for Convolution Kernels in Document-level Sentiment Classification*, Zhaopeng Tu, Yifan He, Jennifer Foster, Josef van Genabith, Qun Liu and Shouxun Lin

Thank you

Acknowledgments

This research has been partially supported by the European project EternalS #247758:

Trustworthy Eternal Systems via Evolving Software, Data and Knowledge

Many Thanks to the IBM Watson team and all the other co-authors and contributors of the **iKernels** group

Acknowledgments

- I wish to thank Thorsten Joachims, Fabio Massimo Zanzotto, Daniele Pighin, Aliaksei Severyn for using some of their slides

References

- M. Dinarelli, A. Moschitti, and G. Riccardi. *Discriminative Reranking for Spoken Language Understanding*. IEEE Transaction on Audio, Speech and Language Processing, 2011.
- 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/~moschitt/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).

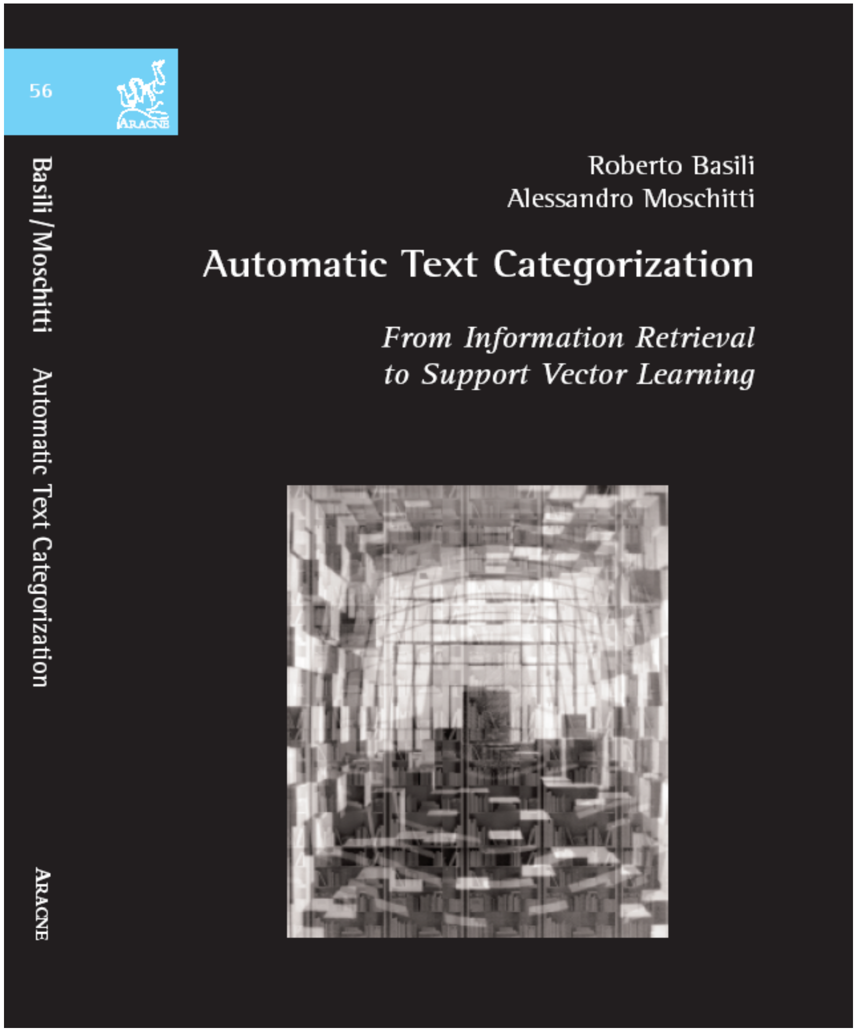
References

- 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.

An introductory book on SVMs, Kernel methods and Text Categorization



Non-exhaustive reference list from other authors

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

Non-exhaustive reference list from other authors

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

Non-exhaustive reference list from other authors

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

Non-exhaustive reference list from other authors

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

Non-exhaustive reference list from other authors

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

Non-exhaustive reference list from other authors

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