Support Vector Machines and Kernel Methods for Robust Semantic NL Processing

Roberto Basili\(^{(1)}\), Alessandro Moschitti\(^{(2)}\)

\(^{(1)}\) DISP, Università di Roma, Tor Vergata,
\(^{(2)}\) Università Trento
Overview

• Theory and practice of Support Vector Machines
• Kernel for HLTs
  – Structural Kernels
  – Lexical Semantic Kernels
• Semantic Role Labeling
  – Linear Features
  – The role of Syntax
• Classification for Question Answering
Predicate and Arguments

• The syntax-semantic mapping

```
        S
       / \
     N   VP
    /   / \ \
 Paul V   NP   PP
   / \     /  /  \
  Arg. 0 gives D N IN N
     \          /  /  \
      \        /   /   \
       \      /    /    \
        \    /     /     \
         \  /     /      \
           \ /     /       \
            \ /     /        \
             \ /     /         \
               \ /     /           \
                \ /     /             \
                 \ /     /               \
                  \ /     /                 \
                   \ /     /                   \
                    \ /     /                     \
                     \ /     /                       \
                      \ /     /                         \
                       \ /     /                           \
                        \ /     /                                 \
                         \ /     /                                   \
                          \ /     /                                     \
                           \ /     /                                       \
                            \ /     /                                         \
                             \ /     /                                           \
                              \ /     /                                             \
                               \ /     /                                               \
                                \ /     /                                                 \
                                 \ /     /                                                   \
                                  \ /     /                                                     \
                                   \ /     /                                                       \
                                    \ /     /                                                          \
                                     \ /     /                                                            \
                                      \ /     /                                                              \
                                       \ /     /                                                                \
                                        \ /     /                                                                  \
                                         \ /     /                                                                    \
                                          \ /     /                                                                      \
                                           \ /     /                                                                        \
                                            \ /     /                                                                              \
                                             \ /     /                                                                                      \
                                              \ /     /                                                                                          \
                                               \ /     /                                                                                                    \
                                                \ /     /                                                                                                           \
                                                 \ /     /                                                                                                                  \
                                                  \ /     /                                                                                                                            \
                                                    \ /     /                                                                                                                                  \
                                                     \ /     /                                                                                                                                            \
                                                      \ /     /                                                                                                                                                \
                                                        \ /     /                                                                                                                                                    \
                                                         \ /     /                                                                                                                                               \
                                                          \ /     /                                                                                                                                                 \
                                                           \ /     /                                                                                                                                                   \
                                                            \ /     /                                                                                                                                                     \
                                                             \ /     /                                                                                                                                                           \
                                                              \ /     /                                                                                                                                                                \
                                                               \ /     /                                                                                                                                                                           \
                                                                \ /     /                                                                                                                                                                          \
                                                                 \ /     /                                                                                                                                                                         \
                                                                  \ /     /                                                                                                                                                                         
                                                                                       `-- D gives a lecture in Rome
                                                                                        Arg. 1 Arg. M
                                                                                         
                                                                                         `-- Predicate
```

• Different semantic annotations (e.g. PropBank vs. FrameNet)
Linking syntax to semantics

- Police arrested the man for shoplifting
Semantics in NLP: Resources

• Lexicalized Models
  – Propbank
  – NomBank

• Framenet
  – Inspired by frame semantics
  – Frames are lexicalized prototyopes for real-world situations
  – Participants are called frame elements (roles)
Generative vs. Discriminative Learning in NLP

- **Generative models** (e.g. HMMs) require
  - The design of a model of *visible* and *hidden* variables
  - The definition of *laws of association* between hidden and visible variables
  - *Robust estimation methods* from the available samples

- **Limitations:**
  - Lack of precise generative models for language phenomena
  - Data sparseness: most of the language phenomena are simply too rare for robust estimation even in large samples
Generative vs. Discriminative Learning

- **Discriminative models** are not tight to any model (i.e. specific association among the problem variables).
- They learn to discriminate negative from positive evidence without building an explicit model of the target property.
- They derive useful evidence from training data only through observed individual features by optimizing some function of the recognition task (e.g. error).
- Examples of discriminative models are the perceptrons (i.e. linear classifiers).
An hyperplane has equation:

\[ f(\tilde{x}) = \tilde{x} \cdot \tilde{w} + b, \quad \tilde{x}, \tilde{w} \in \mathbb{R}^n, b \in \mathbb{R} \]

\( \tilde{x} \) is the vector of the instance to be classified
\( \tilde{w} \) is the hyperplane gradient

Classification function:

\[ h(x) = \text{sign}(f(x)) \]
Linear Classifiers (2)

- Computationally simple.
- Basic idea: select an hypothesis that makes no mistake over training-set.
- The separating function is equivalent to a neural net with just one neuron (perceptron)
Perceptron

\[ \varphi(\vec{x}) = \text{sgn}\left( \sum_{i=1}^{n} w_i \times x_i + b \right) \]
Geometric Margin
Geometric margin vs. data points in the training set

Geometrical margin

Training set margin
Maximal margin vs other margins
Perceptron: on-line algorithm

\[ \vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \leq i \leq l} \| \vec{x}_i \| \]

Repeat

for i = 1 to \( \ell \)

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

\[
\begin{align*}
\vec{w}_{k+1} &= \vec{w}_k + \eta y_i \vec{x}_i \\
b_{k+1} &= b_k + \eta y_i R^2
\end{align*}
\]

k = k + 1

endif

endfor

until no error is found

return k, (\( \vec{w}_k \), \( b_k \))
$* \ t = 1$
$0 \ t = -1$
Duality

- The decision function of linear classifiers can be written as follows:
  \[ h(x) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) = \text{sgn}\left( \sum_{j=1}^{\ell} \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x} + b \right) = \text{sgn}\left( \sum_{i=1}^{\ell} \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x} + b \right) \]

- as well the adjustment function
  \[
  \text{if } y_i \left( \sum_{j=1}^{\ell} \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b \right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta
  \]

- The learning rate \( \eta \) impacts only in the re-scaling of the hyperplanes, and does not influence the algorithm (\( \eta = 1 \)).

\[\rightarrow\] Training data only appear in the scalar products!!
Which hyperplane?
Maximum Margin Hyperplanes

IDEA: Select the hyperplane that maximizes the margin
How to get the maximum margin?

The geometric margin is:

\[
\frac{2|k|}{\|w\|}
\]

Optimization problem

\[
\text{MAX} \quad \frac{2|k|}{\|w\|}
\]

\[
\vec{w} \cdot \vec{x} + b \geq +k, \text{ if } \vec{x} \text{ is a positive ex.}
\]

\[
\vec{w} \cdot \vec{x} + b \leq -k, \text{ se } \vec{x} \text{ is a negative ex.}
\]
The optimization problem

• The optimal hyperplane satisfies:
  – Minimize $\tau(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2$
  – Under: $y_i ((\vec{w} \cdot \vec{x}_i) + b) \geq 1, i = 1, \ldots, l$

• The dual problem is simpler
Dual optimization problem

The Lagrangian dual problem of the above primal problem is

\[
\begin{align*}
\text{maximize} & \quad \theta(\tilde{\alpha}, \tilde{\beta}) \\
\text{subject to} & \quad \tilde{\alpha} \geq \tilde{0}
\end{align*}
\]

where \( \theta(\tilde{\alpha}, \tilde{\beta}) = \inf_{w \in W} L(\tilde{w}, \tilde{\alpha}, \tilde{\beta}) \)
Some consequences

• Lagrange constraints: $\sum_{i=1}^{l} a_i y_i = 0 \quad \tilde{w} = \sum_{i=1}^{l} \alpha_i y_i \tilde{x}_i$

• Karush-Kuhn-Tucker constraints

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

• The support vector are $\tilde{x}_i$ having not null $\alpha_i$, i.e. such that $y_i (\tilde{x}_i \cdot \tilde{w} + b) = -1$

They lie on the frontier

• $b$ is derived through the following formula

$$b^* = -\frac{\tilde{w}^* \cdot \tilde{x}^+ + \tilde{w}^* \cdot \tilde{x}^-}{2}$$
Non linearly separable training data

Slack variables \( \xi_i \) are introduced

Mistakes are allowed and optimization function is penalized
Soft Margin SVMs

New constraints:

\[ y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \quad \forall \vec{x}_i \]
\[ \xi_i \geq 0 \]

Objective function:

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

\( C \) is the trade-off between margin and errors
Dual optimization problem

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

\[
\alpha_i \geq 0, \quad \forall i = 1, \ldots, m
\]

\[
\sum_{i=1}^{m} y_i \alpha_i = 0
\]
### Soft Margin Support Vector Machines

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

$$y_i (\tilde{w} \cdot \tilde{x}_i + b) \geq 1 - \xi_i \quad \forall \tilde{x}_i$$

$$\xi_i \geq 0$$

- The algorithm tries to keep $\xi_i$ to 0 and maximize the margin.
- OBS: The algorithm does not minimize the number of errors (NP-complete problem) but just minimize the sum of the distances from the hyperplane.
- If $C \to \infty$, the solution is the one with the hard-margin.
- If $C = 0$ we get $\| \tilde{w} \| = 0$. In fact, it is always possible to satisfy $y_i b \geq 1 - \xi_i \quad \forall \tilde{x}_i$.
- When $C$ grows the number of errors is decreased with the error set to 0, when $C \to \infty$ (i.e. the hard-margin formulation).
Robustness: Soft vs Hard Margin SVMs

Soft Margin SVM

Hard Margin SVM

\[ \bar{w} \cdot \bar{x} + b = 0 \]
Soft vs Hard Margin SVMs

• A Soft-Margin SVM has always a solution
• A Soft-Margin SVM is more robust wrt odd training examples
  – Insufficient Vocabularies
  – High ambiguity of linguistic features
• An Hard-Margin SVM requires no parameter
References


• A tutorial on Support Vector Machines for Pattern Recognition (C.J.Burges )

• The Vapnik-Chervonenkis Dimension and the Learning Capability of Neural Nets (E.D: Sontag)

• Computational Learning Theory
  (Sally A Goldman Washington University St. Louis Missouri)
  – http://www.learningtheory.org/articles/COLTSurveyArticle.ps

• AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods), N. Cristianini and J. Shawe-Taylor Cambridge University Press.

• The Nature of Statistical Learning Theory, V. N. Vapnik - Springer Verlag (December, 1999)
An introductory book on SVMs, Kernel methods and Text Categorization

Roberto Basili
Alessandro Moschitti

Automatic Text Categorization
From Information Retrieval to Support Vector Learning
Semantic Role Labeling @ UTV

• An important application of SVM is Semantic Role labeling wrt Propbank or Framenet

• In the UTV system, a cascade of classification steps is applied:
  – Predicate detection
  – Boundary recognition
  – Argument categorization (Local models)
  – Reranking (Joint models)

• Input: a sentence and its parse trees
Linking syntax to semantics

- Police arrested the man for shoplifting
Motivations

• Modeling syntax in Natural Language learning task is complex, e.g.
  – Semantic role relations within predicate argument structures and
  – Question Classification

• Tree kernels are natural way to exploit syntactic information from sentence parse trees
  – useful to engineer novel and complex features.

• How do different tree kernels impact on different parsing paradigms and different tasks?

• Are they efficient in practical applications?
Tree kernels: Outline

• Tree kernel types
  – Subset (SST) Tree kernel
  – Subtree (ST) kernel
  – The Partial Tree kernel

• Fast kernel algorithms
  – Efficient evaluation of PT kernel

• Two NLP applications:
  – Semantic Role Labeling
  – Question Classification

• Tree kernel Evaluations
The Collins and Duffy’s Tree Kernel
(called SST in [Vishwanathan and Smola, 2002])
The overall fragment set
Explicit feature space

\[
\tilde{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0)
\]

• \(\tilde{x}_1 \cdot \tilde{x}_2\) counts the number of common substructures
Implicit Representation

\[ \bar{x}_1 \cdot \bar{x}_2 = \phi(T_1) \cdot \phi(T_2) = K(T_1, T_2) = \]
\[ = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2) \]
Implicit Representation

\[ \bar{x}_1 \cdot \bar{x}_2 = \phi(T_1) \cdot \phi(T_2) = K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2) \]

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

\[ \Delta(n_1, n_2) = \begin{cases} 0, & \text{if the productions are different} \\ 1, & \text{if pre-terminal} \\ \text{else} \end{cases} \]
\[ \Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j))) \]
Weighting

- Decay factor

\[
\Delta(n_1, n_2) = \lambda, \quad \text{if pre- terminals else}
\]

\[
\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))
\]

- Normalization

\[
K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) \times K(T_2, T_2)}}
\]
SubTree (ST) Kernel
[Vishwanathan and Smola, 2002]
Evaluation

- Given the equation for the SST kernel

\[
\Delta(n_1, n_2) = 0, \text{ if the productions are different else } \\
\Delta(n_1, n_2) = 1, \text{ if pre - terminals else } \\
\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))
\]
Evaluation

- Given the equation for the ST kernel

\[ \Delta(n_1, n_2) = 0, \text{ if the productions are different } \]
\[ \Delta(n_1, n_2) = 1, \text{ if pre-terminals else } \]

\[ \Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\Delta(ch(n_1, j), ch(n_2, j))) \]
Labeled Ordered Tree Kernel

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

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

- if the node labels of \( n_1 \) and \( n_2 \) are different then
  \[
  \Delta(n_1, n_2) = 0;
  \]
- else
  \[
  \Delta(n_1, n_2) = 1 + \sum_{\tilde{J}_1, \tilde{J}_2, l(\tilde{J}_1) = l(\tilde{J}_2)} \prod_{i=1}^{l(\tilde{J}_1)} \Delta(c_{n_1}[\tilde{J}_{1i}], c_{n_2}[\tilde{J}_{2i}])
  \]

• By adding two decay factors we obtain:

\[
\mu \left( \lambda^2 + \sum_{\tilde{J}_1, \tilde{J}_2, l(\tilde{J}_1) = l(\tilde{J}_2)} \lambda^{d(\tilde{J}_1) + d(\tilde{J}_2)} \prod_{i=1}^{l(\tilde{J}_1)} \Delta(c_{n_1}[\tilde{J}_{1i}], c_{n_2}[\tilde{J}_{2i}]) \right)
\]
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

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

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

\[
\Delta(a, b) \times \sum_{i=1}^{s_1} \sum_{r=1}^{s_2} \lambda^{s_1 - i + s_2 - r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r])
\]
Efficient Evaluation (2)

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

Note that \( D_p \) satisfies the recursive relation:

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

- 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 \( \rho \) is the maximum branching factor (\( p = \rho \))
SRL Demo

• Kernel-based system for SRL over raw texts
• based on the Charniak parser
• Adopts the Propbank standard but has also been applied to Framenet
Kernel-based Semantic Role Labeling

SRL User Interface

Enter a new sentence:

Select an example sentence:

Run System  Show Results

Couch-potato jocks watching ABC's "Monday Night Football" can now vote during halftime for the greatest play in 20 years from among four or five filmed replays.

During last summer, two thousand trees were burnt by criminals.

Mary would like to understand why John betrayed her.
Couch-potato lacks watching ABC’s “Monday Night Football” can now vote during halftime for the greatest play in 20 years from among four or five filmed replays.
Couch-potato jocks watching ABC's "Monday Night Football" can now vote during halftime for the greatest play in 20 years from among four or five filmed replays.
Capello will be officially unveiled on Monday and Leonardo believes that he is the right man to take England forward.
Capello will be officially unveiled on Monday and Leonardo believes that he is the right man to take England forward.
Capello will be officially unveiled on Monday and Leonardo believes that he is the right man to take England forward.
Automatic Predicate Argument Extraction

- Boundary Detection
  - One binary classifier

- Argument Type Classification
  - Multi-classification problem
  - $n$ binary classifiers (ONE-vs-ALL)
  - Select the argument with maximum score
Typical standard flat features
(Gildea & Jurafsky, 2002)

• Phrase Type of the argument
• Parse Tree Path, between the predicate and the argument
• Head word
• Predicate Word
• Position
• Voice
Features for the linear kernel in SRL

Table 1: Classifier features. The features are divided into constituent-based and the dependent systems are marked C and D, respectively.

<table>
<thead>
<tr>
<th>Features</th>
<th>Argument identification</th>
<th>Argument classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARGETLEMA</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>FES</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>TargetPOS</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>Voice</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>Position</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>ArgWord/POS</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>LeftWord/POS</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>RightWord/POS</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>ParentWord/POS</td>
<td>C, D</td>
<td></td>
</tr>
<tr>
<td>C-SUBCAT</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>C-PATH</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>PhrasalType</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>GovCat</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D-SUBCAT</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>D-PATH</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ChildDepSet</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ParentHasObj</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>RELToParent</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

C-SUBCAT. Subcategorization frame: corresponds to the phrase-structure rule used to expand the phrase around the target. For *give* in the example, this feature is VP→VB NP NP.

C-PATH. A string representation of the path through the constituent tree from the target word to the argument constituent. For instance, the path from *gave* to *she* is ↑VP-↑S-↓NP.

PhrasalType. Phrase type of the argument constituent, e.g., NP for *she*.

GovCat. Governing category: this feature is either S or VP, and is found by starting at the argument constituent and moving upwards until either a VP or a sentence node (S, SINV, or SQ) is found. For instance, for *she*, this feature is S, while for *the horse*, it is VP. This can be thought of as a very primitive way of distinguishing subjects and objects.
An example

Phrase
Type
Predicate
Word
Head Word
Parse Tree
Position
Right
Voice
Active

Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active

Phrase
Type
Predicate
Word
Head Word

Parse Tree
Position
Right
Voice
Active
Flat features (Linear Kernel)

- To each example is associated a vector of 6 feature types

\[
\tilde{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 1, 1)
\]

\[
\begin{align*}
\text{PT} & \quad \text{PTP} & \quad \text{HW} & \quad \text{PW} & \quad \text{P} & \quad \text{V}
\end{align*}
\]

- The dot product counts the number of features in common

\[
\tilde{x} \cdot \tilde{z}
\]
Automatic Predicate Argument Extraction

Deriving Positive/Negative example

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
Argument Classification Accuracy

% Training Data

Accuracy

0.75
0.80
0.83
0.85
0.88

ST
SST
Linear
PT
# SRL in FrameNet: Results

<table>
<thead>
<tr>
<th>Eval Setting</th>
<th>Tree Kernels</th>
<th>Tree Kernels + PK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P$</td>
<td>$R$</td>
</tr>
<tr>
<td>BD</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BD Proj.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BD+RC</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BD+RC Proj.</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PK alone [TK]</th>
<th>TK + PK [TKL + PK]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BD</td>
<td>.949 .652 .773</td>
<td>.915 .698 .792</td>
</tr>
<tr>
<td>BD Proj.</td>
<td>.919 .631 .748</td>
<td>.875 .668 .758</td>
</tr>
<tr>
<td>BD+RC</td>
<td>.697 .479 .568</td>
<td>.680 .519 .588</td>
</tr>
<tr>
<td>BD+RC Proj.</td>
<td>.672 .462 .548</td>
<td>.648 .495 .561</td>
</tr>
</tbody>
</table>

Table 4.1: Results on FrameNet dataset. The table shows Precision, Recall, and F-measure achieved by the Polynomial Kernel (PK) and two different Tree Kernels (TK and TKL). Also, results for their combinations are shown. All experiments exploit 2% training data for Boundary Detection, and 90% for Role Classification.
Framenet SRL: best results

- **Best system** [Erk & Pado, 2006]
  - 0.855 Precision, 0.669 Recall
  - 0.751 F1

- Trento (+RTV) system (Coppola, PhD2009)

<table>
<thead>
<tr>
<th>Eval Setting</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BD (nodes)</td>
<td>1.0</td>
<td>.732</td>
<td>.847</td>
</tr>
<tr>
<td>BD (words)</td>
<td>.963</td>
<td>.702</td>
<td>.813</td>
</tr>
<tr>
<td>BD+RC (nodes)</td>
<td>.784</td>
<td>.571</td>
<td>.661</td>
</tr>
<tr>
<td>BD+RC (words)</td>
<td>.747</td>
<td>.545</td>
<td>.630</td>
</tr>
</tbody>
</table>

Table 4.2: Results on the FrameNet dataset. Best configuration from Table 4.1, raised to 90% of training data for BD and RC.
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

- 5500 training and 500 test questions [Li and Roth, 2004]
- Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Using the whole question parse trees
  - Two parsing paradigms: Constituent and Dependency
  - Example
    
    “What is an offer of direct stock purchase plan?”
Answer Type and Focus

• **Focus** is the word that expresses the relevant entity in the question
  - Used to select a set of relevant documents
  - ES: Where was **Mozart** born?

• **Answer Type** is the category of the entity to be searched as answer
  - PERSON, MEASURE, TIME
    PERIOD, DATE, ORGANIZATION, DEFINITION
  - ES: Where was **Mozart** born?
    • LOCATION
What famous communist leader died in Mexico City?

Answer type: PERSON
Focus: leader

This rule matches any question starting with what, whose first noun, if any, is a person (i.e. satisfies the person-p predicate)
Answer Type Taxonomies

Figure 1: Answer Type Taxonomy

(M. Pasca and S. Harabagiu, SIGIR 2001)
QA: an overall view

Figure 3: Retrieval Feedbacks in a Q/A System

(M. Pasca and S. Harabagiu, SIGIR 2001)
The dependency Tree

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

  - is
  - What
  - offer
  - an
  - plan
  - direct
  - stock
  - purchase

- PTs can be very effective, e.g.

  - [Plan [direct][purchase]]
  - [Plan [stock][purchase]]
<table>
<thead>
<tr>
<th>Parsers</th>
<th>Constituent</th>
<th>Dependency</th>
<th>BOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernels</td>
<td>SST PT</td>
<td>SST PT</td>
<td>Linear</td>
</tr>
<tr>
<td>Acc.</td>
<td>88.2</td>
<td>87.2</td>
<td>82.1 90.4</td>
</tr>
</tbody>
</table>
Conclusions

• Kernel–based learning is very useful in NLP as for the possibility of embedding similarity measures for highly structured data
  – Sequence
  – Trees

• Tree kernels are a natural way to introduce syntactic information in natural language learning.
  – Very useful when few knowledge is available about the proposed problem.
  – Alleviate manual feature engineering (predicate knowledge)

• Different forms of syntactic information require different tree kernels.
  – Collins and Duffy’s kernel (SST) useful for constituent parsing
  – The new Partial Tree kernel useful for dependency parsing
Conclusions (2)

• Experiments on SRL and QC show that
  – PT and SST are efficient and very fast
  – Higher accuracy when the proper kernel is used for the target task

• Open research issue are
  – Proper kernel design issues for the different tasks
  – Combination of syntagmatic kernels with semantic ones
    • An example is the Wordnet-based kernel in (Basili et al CoNLL 05)
Tree-kernel: References

• Available over the Web: