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

Alessandro Moschitti
Dept. of Computer Science and Engineering
University of Trento
moschitti@disi.unitn.it

ACL 2012
The 50th Annual Meeting of the Association for Computational Linguistics
July 8 (Sun) – July 14 (Sat), 2012 ICC JEJU, Jeju Island, Korea
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

\[
f : T \rightarrow \{C, \overline{C}\}
\]

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

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

buy acquisition stocks sell market

\[ \Phi(d_z) = \tilde{z} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0) \]

buy company stocks sell

- The dot product \( \tilde{x} \cdot \tilde{z} \) counts the number of features in common

- This provides a sort of \textit{similarity}
Linear Classifier

The equation of a hyperplane is

\[ f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x}, \vec{w} \in \mathbb{R}^n, b \in \mathbb{R} \]

- \( \vec{x} \) 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, \( \tilde{x} \rightarrow \phi(\tilde{x}) \)
A kernel-based Machine: Perceptron training

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

do

for i = 1 to \( \ell \)

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

\[ \tilde{w}_{k+1} = \tilde{w}_k + \eta y_i \tilde{x}_i \]

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

k = k + 1

endif

endfor

while an error is found

return k, (\( \tilde{w}_k \), \( b_k \))
Since the sign of the contribution \( x_i \) is given by \( y_i \), \( \alpha_i \) is positive and is proportional (through the \( \eta \) factor) to the number of times that \( x_i \) is incorrectly classified.

Difficult points that cause many mistakes will be associated with large \( \alpha_i \).

It is interesting to note that, if we fix the training set \( S \), we can use \( \alpha_i \) as alternative coordinates of a dual space to represent the target hypothesis associated with \( w \).

The resulting decision function is the following:

\[
h(x) = \text{sgn}(w \cdot x + b) = \text{sgn}(m \sum_{i=1}^{m} \alpha_i y_i x_i \cdot x + b) = \text{sgn}(\sum_{i=1}^{m} \alpha_i y_i (x_i \cdot x) + b)
\]

(11)

Given the dual representation, we can adopt a learning algorithm that works in the dual space described in Table 3. Note that as the Novikoff's theorem states that the learning rate \( \eta \) only changes the scaling of the hyperplanes, it does not affect the algorithm thus we can set \( \eta = 1 \).

On the contrary, if the perceptron algorithm starts with a different initialization, it will find a different separating hyperplane. The reader may wonder if such hyperplanes are all equivalent in terms of the classification accuracy of the test set; the answer is no: different hyperplanes may lead to different error probabilities. In particular, the next...
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:

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

- Note that data only appears in the scalar product
Dual Representation for Learning

- as well as the updating function

\[
\text{if } y_i (\sum_{j=1}^{\ell} \alpha_j y_j \bar{x}_j \cdot \bar{x}_i + b) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta
\]

- The learning rate \(\eta\) only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix \(\eta = 1\).
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 \( \frac{2|k|}{\|w\|} \)
Support Vector Machines

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

We need to solve

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

$w \cdot x + b \geq 1$, if $x$ is positive

$w \cdot x + b \leq -1$, if $x$ is negative
Optimization Problem

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

- To solve the dual problem we need to evaluate:
  \[ \theta(\alpha, \beta) = \inf_{\vec{w} \in W} L(\vec{w}, \alpha, \beta) \]

- Let us impose the derivatives to 0, with respect to \( \vec{w} \)
  \[ \frac{\partial L(\vec{w}, b, \alpha)}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{m} y_i \alpha_i \vec{x}_i \]
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

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

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

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j$$
The Final Dual Optimization Problem

\[
\text{maximize} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \\
\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m \\
\sum_{i=1}^{m} y_i \alpha_i = 0
\]
Soft Margin optimization problem

\[
\begin{align*}
\text{maximize} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij}) \\
\text{subject to} \quad & \alpha_i \geq 0, \quad \forall i = 1, \ldots, m \\
& \sum_{i=1}^{m} y_i \alpha_i = 0
\end{align*}
\]
Kernels in Support Vector Machines

- In Soft Margin SVMs we maximize:

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

- By using kernel functions we rewrite the problem as:

\[
\begin{aligned}
\text{maximize} & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (k(o_i, o_j) + \frac{1}{C} \delta_{ij}) \\
\alpha_i & \geq 0, \quad \forall i = 1, \ldots, m \\
\sum_{i=1}^{m} y_i \alpha_i & = 0
\end{aligned}
\]
Soft Margin Support Vector Machines

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

- The algorithm tries to keep $\xi_i$ 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

\[ \text{Var}_1 = \sum_i \xi_i \]

\[ \text{Var}_2 = \sum_i \xi_i \]

\[ \sum_i \xi_i \leq C \]

Soft Margin SVM

\[ \tilde{\mathbf{w}} \cdot \mathbf{x} + b = 0 \]

Hard Margin SVM

\[ \tilde{\mathbf{w}} \cdot \mathbf{x} + b = 0 \]
Parameters

\[
\min \frac{1}{2} \| \tilde{w} \|_2^2 + C \sum_i \xi_i = \min \frac{1}{2} \| \tilde{w} \|_2^2 + C^+ \sum_i \xi_i^+ + C^- \sum_i \xi_i^-
\]

\[
= \min \frac{1}{2} \| \tilde{w} \|_2^2 + C \left( J \sum_i \xi_i^+ + \sum_i \xi_i^- \right)
\]

- C: trade-off parameter
- J: cost factor
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 \( \phi \) is a mapping from \( X \) to an (inner product) feature space.

- Kernels are the product of mapping functions such as

\[
    \vec{x} \in \mathbb{R}^n, \quad \phi(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_m(\vec{x})) \in \mathbb{R}^m
\]
The Kernel Gram Matrix

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

\[
K_{training} = \begin{bmatrix}
  k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_m) \\
  k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_m) \\
  \vdots & \vdots & \ddots & \vdots \\
  k(x_m, x_1) & k(x_m, x_2) & \cdots & k(x_m, x_m)
\end{bmatrix}
\]

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

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

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

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

Def. B.13 Positive (Semi-) definite Matrix
A square matrix $A \in \mathbb{R}^{n \times n}$ 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 \( K(x, z) \) be a symmetric function on \( X \). Then \( K(x, z) \) is a kernel function if and only if the matrix

\[
k(x, z) = \phi(x) \cdot \phi(z)
\]

is positive semi-definite (has non-negative eigenvalues).

- If the matrix is positive semi-definite then we can find a mapping \( \phi \) implementing the kernel function
Mercer’s Theorem (finite space)

Let us consider \( K = \left( K(\vec{x}_i, \vec{x}_j) \right)_{i,j=1}^n \)

\( K \) symmetric \( \Rightarrow \exists V: K = V \Lambda V' \) for Takagi factorization of a complex-symmetric matrix, where:

- \( \Lambda \) is the diagonal matrix of the eigenvalues \( \lambda_t \) of \( K \)
- \( \vec{v}_t = (v_{ti})_{i=1}^n \) are the eigenvectors, i.e. the columns of \( V \)
- Let us assume lambda values non-negative

\[ \phi: \vec{x}_i \rightarrow \left( \sqrt{\lambda_t} v_{ti} \right)_{t=1}^n \in \mathbb{R}^n, \ i = 1, \ldots, n \]
Mercer’s Theorem
(sufficient conditions)

- Therefore

\[ \Phi(\bar{x}_i) \cdot \Phi(\bar{x}_j) = \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} = (V\Lambda V')_{ij} = K_{ij} = K(\bar{x}_i, \bar{x}_j) \]

- which implies that K is a kernel function
Mercer’s Theorem
(necessary conditions)

- Suppose we have negative eigenvalues $\lambda_s$ and eigenvectors $\tilde{v}_s$ the following point

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

- has the following norm:

$$\left\| \tilde{z} \right\|^2 = \tilde{z} \cdot \tilde{z} = \sqrt{\Lambda} V' \tilde{v}_s \sqrt{\Lambda} V' \tilde{v}_s = \tilde{v}_s' V \sqrt{\Lambda} \sqrt{\Lambda} V' \tilde{v}_s =$$

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

this contradicts the geometry of the space.
Is it a valid kernel?

- It may not be a kernel so we can use $M' \cdot M$

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

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

From the previous proposition it follows that: If we find a decomposition $A$ in $M' M$, then $A$ is semi-definite positive matrix as

$$\bar{x}' A \bar{x} = \bar{x}' M' M \bar{x} = (M \bar{x})' (M \bar{x}) = M \bar{x} \cdot M \bar{x} = || M \bar{x} ||^2 \geq 0.$$
Valid Kernel operations

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

\[
\Phi(d_x) = \bar{x} = (0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1)
\]

\[
\text{buy acquisition stocks sell market}
\]

\[
\Phi(d_z) = \bar{z} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 0)
\]

\[
\text{buy company stocks sell}
\]

- The dot product \( \bar{x} \cdot \bar{z} \) counts the number of features in common

- This provides a sort of similarity
Feature Conjunction (polynomial Kernel)

- The initial vectors are mapped in a higher space

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

- More expressive, as \((x_1x_2)\) encodes

Stock+Market vs. Downtown+Market features

- We can smartly compute the scalar product as

\[
\Phi(\vec{x}) \cdot \Phi(\vec{z}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) = \\
= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 = \\
= (x_1z_1 + x_2z_2 + 1)^2 = (\vec{x} \cdot \vec{z} + 1)^2 = K_{poly}(\vec{x}, \vec{z})
\]
Sub-hierarchies in WordNet

- **artefact**
  - **motor vehicle**
    - **motorcar**
    - **go-kart**
    - **truck**
    - **hatch-back**
    - **compact**
    - **gas guzzler**

  - **{living thing, organism}**
    - **{plant, flora}**
    - **{animal, fauna}**
  - **{non-living thing, object}**
    - **{natural object}**
    - **{substance}**
    - **{artifact}**
    - **{food}**

- **{thing, entity}**
  - **{person, human being}**
Similarity based on WordNet

Inverted Path Length:

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

Wu & Palmer:

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

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

Doc 1

- industry
- telephone
- market

Doc 2

- company
- product
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

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

\[ \text{bank} \quad \text{ank} \quad \text{bnk} \quad \text{bk} \quad \text{b} \]

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

\[ \text{rank} \quad \text{ank} \quad \text{rnk} \quad \text{rk} \quad \text{r} \]

\[ \vec{x} \cdot \vec{z} \text{ counts the number of common substrings} \]

\[ \vec{x} \cdot \vec{z} = \phi("bank") \cdot \phi("rank") = k("bank","rank") \]
Formal Definition

\[ s = s_1, \ldots, s_{|s|}, \quad \vec{I} = (i_1, \ldots, i_{|u|}) \]

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

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

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

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

- $\phi_a(\text{Bank}) = \phi_a(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(2-2+1)} = \lambda,$

- $\phi_n(\text{Bank}) = \phi_n(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(3-3+1)} = \lambda,$

- $\phi_k(\text{Bank}) = \phi_k(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(4-4+1)} = \lambda,$

- $\phi_{an}(\text{Bank}) = \phi_{an}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(3-2+1)} = \lambda^2,$

- $\phi_{ank}(\text{Bank}) = \phi_{ank}(\text{Rank}) = \lambda^{(i_3-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3,$

- $\phi_{nk}(\text{Bank}) = \phi_{nk}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-3+1)} = \lambda^2$

- $\phi_{ak}(\text{Bank}) = \phi_{ak}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3$

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

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

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

$D_p$ satisfies the recursive relation:

$$D_p(k,l) = SK_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k,l-1) + \lambda D_p(k-1,l) - \lambda^2 D_p(k-1,l-1)$$
Evaluating DP2

- Evaluate the weight of the string of size $p$ in case a character will be matched.
- This is done by multiplying the double summation by the number of substrings of size $p-1$.

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

```
NP  D  N
  VP
    V NP
      delivers D N
        a talk
```
The overall fragment set

Children are not divided
Explicit kernel space

\[ \phi(T_x) = \bar{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0) \]

\[ \phi(T_z) = \bar{z} = (1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 0) \]

\[ \bar{x} \cdot \bar{z} \] 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

\[ \hat{x} \cdot \hat{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) = \]
\[ = \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z) \]

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

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

\[ \Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} \left(1 + \Delta(ch(n_x, j), ch(n_z, j))\right) \]
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
  $\Delta(n_1, n_2) = 1 + \sum_{\bar{J}_1, \bar{J}_2, l(\bar{J}_1) = l(\bar{J}_2)}^{l(\bar{J}_1)} \prod_{i=1}^{l(\bar{J}_1)} \Delta(c_{n_1}[\bar{J}_{1i}], c_{n_2}[\bar{J}_{2i}])$

- By adding two decay factors we obtain:

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

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

Given the two child sequences \( s_1a = c_{n_1} \) and \( s_2b = c_{n_2} \) (\( a \) and \( b \) are the last children), \( \Delta_p(s_1a, s_2b) = D_p \)

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

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

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

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

0. if \( n_1 \) and \( n_2 \) are pre-terminals and \( \text{label}(n_1) = \text{label}(n_2) \) then \( \Delta(n_1, n_2) = \lambda \kappa S(ch^1_{n_1}, ch^1_{n_2}) \),

1. if the productions at \( n_1 \) and \( n_2 \) are different then \( \Delta(n_1, n_2) = 0 \);

2. \( \Delta(n_1, n_2) = \lambda \),

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

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

else

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

Lexical Similarity

PTK
Different versions of Computational Dependency Trees for PTK/SPTK

LOCT

LPST

TOP

be::v
what::w
width::n
the::d
of::i
field::n
a::d football::n

be::v
what::w
width::n
SBJ WP
the::d
of::i
PRD NN P.
NMOD DT
field::n
NMOD IN
a::d
football::n
PMOD NN
NMOD DT
NMOD NN

be::v
what::w
the::d
width::n
of::i
a::d
football::n
field::n

one-vs-all
Tree Kernel Efficiency

\[ y = 0.068x^{1.213} \]

\[ y = 0.081x^{1.705} \]

\[ y = 0.0513x^{2.005} \]
Simple Kernel Applications
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)

```
  SBARQ
   ▼
  /  \
WHNP  SQ
  ▼  ▼
WP    VP
  ▼  ▼
Who  AUX  VP
      ▼  ▼
did  V    NP
      ▼  ▼
deliver  D  N
            ▼  ▼
a      talk
```
Explicit kernel space

\[ \phi(T_x) = \vec{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0) \]

\[ \phi(T_z) = \vec{z} = (1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0) \]

\[ \vec{x} \cdot \vec{z} \] counts the number of common substructures
## Question Classification with SSTK
[Blohedorn&Moschitti, CIKM2007]

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>λ parameter</strong></td>
<td>0.4</td>
</tr>
<tr>
<td>linear (bow)</td>
<td></td>
</tr>
<tr>
<td>string matching</td>
<td>0.890</td>
</tr>
<tr>
<td>full</td>
<td>0.904</td>
</tr>
<tr>
<td>full-ic</td>
<td>0.908</td>
</tr>
<tr>
<td>path-1</td>
<td>0.906</td>
</tr>
<tr>
<td>path-2</td>
<td>0.896</td>
</tr>
<tr>
<td>lin</td>
<td>0.908</td>
</tr>
<tr>
<td>wup</td>
<td>0.908</td>
</tr>
</tbody>
</table>
Same Task with PTK, SPTK and Dependency Trees

The aim of the experiments is to analyze the role of lexical similarity embedded in syntactic structures. For this purpose, we present several syntactic structures on QC. The first row lists the tree kernel models. The last column shows the different structures described in [2].

We implemented multitask learning our model, associated with the maximum SVM margin. The -x of SVMs using ST's current state is state-of-the-art, but it is improved slightly lower value on ontologically significant difference.6

The first column shows the different structures described in [2]. For more details, we applied ST to several structures for QC is reported in Table 3. We also extended the SVM to soft

Wak [x] is which is a large scale document collection made by [xy] and describes in [xfis xfl] which includes structural kernels such as ST and PT and SPT. Applied to the several structures for QC, the ST defined in [2]

Our referring corpus is the U[x; dataset [xz] and is composed of a billion tokens see [2] for more details. We implemented multitask learning our model, associated with the maximum SVM margin. The -x of SVMs using ST's current state is state-of-the-art, but it is improved slightly lower value on ontologically significant difference.

Note that higher accuracy values for smoothed ST are shown in Table 3. The aim of the experiments is to analyze the role of lexical similarity in syntactic structures. For this purpose, we present our results.

The first column shows the different structures described in [2]. For more details, we applied ST to several structures for QC is reported in Table 3. We also extended the SVM to soft

Wak [x] is which is a large scale document collection made by [xy] and describes in [xfis xfl] which includes structural kernels such as ST and PT and SPT. Applied to the several structures for QC, the ST defined in [2]

Our referring corpus is the U[x; dataset [xz] and is composed of a billion tokens see [2] for more details. We implemented multitask learning our model, associated with the maximum SVM margin. The -x of SVMs using ST's current state is state-of-the-art, but it is improved slightly lower value on ontologically significant difference.
State-of-the-art Results
[Croce et al., EMNLP 2011]

<table>
<thead>
<tr>
<th></th>
<th>STK</th>
<th>PTK</th>
<th>SPTK(LSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT</td>
<td>91.20%</td>
<td>90.80%</td>
<td>91.00%</td>
</tr>
<tr>
<td>LOCT</td>
<td>-</td>
<td>89.20%</td>
<td>93.20%</td>
</tr>
<tr>
<td>LCT</td>
<td>-</td>
<td>90.80%</td>
<td><strong>94.80%</strong></td>
</tr>
<tr>
<td>LPST</td>
<td>-</td>
<td>89.40%</td>
<td>89.60%</td>
</tr>
<tr>
<td>BOW</td>
<td></td>
<td><strong>88.80%</strong></td>
<td></td>
</tr>
</tbody>
</table>
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 $q$:
  - $K_1(\langle q_1, q_2 \rangle) + K_2(\langle q_1, q_2 \rangle) + ... + K_n(\langle q_1, q_2 \rangle)$
- Tree kernels measure similarity between trees
Syntactic Tree Kernel (STK)
(Collins and Duffy 2002)

Syntactic Tree:

```
VP
  V
    hit
  NP
    D
    a
    N
    phosphor
```
Syntactic Tree Kernel (STK) (Collins and Duffy 2002)
The resulting explicit kernel space

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

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

$\vec{x} \cdot \vec{z}$ 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
When hit by electrons, a phosphor gives off electromagnetic energy in this form.
The given sentence is: "When a phosphor hit by electrons gives off electromag. energy, a form of this is AM-TMP hit."
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

<table>
<thead>
<tr>
<th>Kernel Space</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBC</td>
<td>28.27</td>
<td>70.59</td>
<td>40.38</td>
</tr>
<tr>
<td>BOW</td>
<td>47.67</td>
<td>46.73</td>
<td>47.20</td>
</tr>
<tr>
<td>WSK</td>
<td>47.11</td>
<td>50.65</td>
<td>48.82</td>
</tr>
<tr>
<td>STK-CT</td>
<td>50.51</td>
<td>32.35</td>
<td>39.44</td>
</tr>
<tr>
<td>PTK-CT</td>
<td>47.84</td>
<td>57.84</td>
<td>52.37</td>
</tr>
<tr>
<td>PTK-DT</td>
<td>44.81</td>
<td>57.84</td>
<td>50.50</td>
</tr>
<tr>
<td>PASS</td>
<td>33.50</td>
<td>21.90</td>
<td>26.49</td>
</tr>
<tr>
<td>PSK</td>
<td>39.88</td>
<td>45.10</td>
<td>42.33</td>
</tr>
<tr>
<td>CSK</td>
<td>39.07</td>
<td>77.12</td>
<td>51.86</td>
</tr>
</tbody>
</table>
## Model Combinations

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

<table>
<thead>
<tr>
<th></th>
<th>NoDef</th>
<th>StatDef</th>
<th>NoDef</th>
<th>RuleDef</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># Questions</strong></td>
<td>1606</td>
<td>1606</td>
<td>1875</td>
<td>1875</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>63.76%</td>
<td>65.57%</td>
<td>56.64%</td>
<td>57.51%</td>
</tr>
<tr>
<td><strong>P@70</strong></td>
<td>82.22%</td>
<td>84.53%</td>
<td>72.73%</td>
<td>74.87%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># Def Q’s</th>
<th>Accuracy</th>
<th>P@70</th>
<th>Earnings</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoDef</td>
<td>0</td>
<td>69.71%</td>
<td>86.79%</td>
<td>$24,818</td>
</tr>
<tr>
<td>RuleDef</td>
<td>480</td>
<td>69.23%</td>
<td>86.31%</td>
<td>$24,397</td>
</tr>
<tr>
<td>StatDef</td>
<td>131</td>
<td>69.85%</td>
<td>87.19%</td>
<td>$25,109</td>
</tr>
</tbody>
</table>
Error Analysis

Test Example
• PTK ok
• STK not ok

Training Example

PTK similarity

STK similarity
Question and Answer Classification
Answer/Passage Reranking

Question Analysis

Question

Hypothesis Generation

Answer/Passage Reranking

Answer/Passage

Hypothesis and Evidence Scoring

Candidate Rankin

Hypothesis Sources

Candidate Answer Generation

Supporting Evidence Retrieval

Deep Evidence Scoring

Answer Sources

Evidence Sources

Trained Models

Answer and Confidence
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:

```
BOX
  /\     /
 /  \   /  
What is an offer of ...
```

```
BOX
  /\     /
 /  \   /  
WHNP VBZ DT NN IN ...
```

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 ($\text{PAS}_\text{PTK}$)

- [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}

\[ \Rightarrow \text{BOW+POS, BOW+PT, PT+POS, ...} \]
Results on TREC Data
(5 folds cross validation)

F1-measure

Kernel Type

BOW  POS  POS_SK  WSK  PT  PAS_SSTK  PAS_PTK  BOW+POS  BOW+PT  POS_SK+PT  WSK+PT  POS_SK+PT+PAS_SSTK  POS_SK+PT+PAS_PTK
Results on TREC Data (5 folds cross validation)
Results on TREC Data
(5 folds cross validation)
Results on TREC Data
(5 folds cross validation)

F1-measure

Kernel Type

- BOW
- POS
- POS_SK
- WSK
- PT
- PAS_SSTK
- PAS_PTK
- BOW+POS
- BOW+PT
- POS_SK+PT
- WSK+PT
- POS_SK+PT+PAS_SSTK
- POS_SK+PT+PAS_PTK
Results on TREC Data
(5 folds cross validation)
Results on TREC Data
(5 folds cross validation)

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>22</td>
</tr>
<tr>
<td>POS</td>
<td>34</td>
</tr>
<tr>
<td>POS_SK</td>
<td>32</td>
</tr>
<tr>
<td>WSK</td>
<td>26</td>
</tr>
<tr>
<td>PT</td>
<td>40</td>
</tr>
<tr>
<td>PAS_SSTK</td>
<td>40</td>
</tr>
<tr>
<td>PAS_PTK</td>
<td>36</td>
</tr>
<tr>
<td>BOW+POS</td>
<td>32</td>
</tr>
<tr>
<td>BOW+PT</td>
<td>38</td>
</tr>
<tr>
<td>POS_SK+PT</td>
<td>38</td>
</tr>
<tr>
<td>WSK+PT</td>
<td>36</td>
</tr>
<tr>
<td>POS_SK+PT+PAS_SSTK</td>
<td>34</td>
</tr>
<tr>
<td>POS_SK+PT+PAS_PTK</td>
<td>36</td>
</tr>
</tbody>
</table>
Results on TREC Data
(5 folds cross validation)

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>≈24</td>
</tr>
<tr>
<td>POSSK+STK+PAS_PTK</td>
<td>≈39</td>
</tr>
</tbody>
</table>

⇒ 62% of improvement
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:
“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))))
Kernel Combinations an example

\( K_p^3 \) polynomial kernel of flat features

\( K_{\text{Tree}} \) Tree kernel

Kernel Combinations:

\[
K_{\text{Tree}+P} = \gamma \times K_{\text{Tree}} + K_p^3 ,
\]

\[
K_{\text{Tree} \times P} = K_{\text{Tree}} \times K_p^3
\]

\[
K_{\text{Tree} \times P} = \frac{K_{\text{Tree} \times K_p^3}}{K_{\text{Tree}} \times K_p}
\]

\[
K_{\text{Tree} \times P} = \frac{K_{\text{Tree} \times K_p^3}}{K_{\text{Tree}} \times K_p}
\]
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:
  
  \[
  \text{[Arg}_0 \text{ Paul]} \quad [\text{predicate gives }] \quad [\text{Arg}_1 \text{ a talk}] \quad [\text{Arg}_M \text{ in Rome}]
  \]
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

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

Predicate
S
VP
NP
D
N
PP
IN
N

Arg. 1
Paul delivers a talk in Rome

Voice Active
Given the sentence:

\[
\begin{align*}
&[\text{Arg0 } \text{Paul}] \quad [\text{predicate delivers}] \quad [\text{Arg1 a talk}] \quad [\text{ArgM in formal Style}] \\
\end{align*}
\]

These are Semantic Structures
In other words we consider…
Sub-Categorization Kernel (SCF) [Moschitti, ACL 2004]

Paul delivers a talk in a formal style.
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

Accuracy

FrameNet
PropBank

1 2 3 4 5

d
## PropBank Results

<table>
<thead>
<tr>
<th>Args</th>
<th>P3</th>
<th>PAT</th>
<th>PAT+P</th>
<th>PAT×P</th>
<th>SCF+P</th>
<th>SCF×P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arg0</td>
<td>90.8</td>
<td>88.3</td>
<td>92.6</td>
<td>90.5</td>
<td>94.6</td>
<td>94.7</td>
</tr>
<tr>
<td>Arg1</td>
<td>91.1</td>
<td>87.4</td>
<td>91.9</td>
<td>91.2</td>
<td>92.9</td>
<td>94.1</td>
</tr>
<tr>
<td>Arg2</td>
<td>80.0</td>
<td>68.5</td>
<td>77.5</td>
<td>74.7</td>
<td>77.4</td>
<td>82.0</td>
</tr>
<tr>
<td>Arg3</td>
<td>57.9</td>
<td>56.5</td>
<td>55.6</td>
<td>49.7</td>
<td>56.2</td>
<td>56.4</td>
</tr>
<tr>
<td>Arg4</td>
<td>70.5</td>
<td>68.7</td>
<td>71.2</td>
<td>62.7</td>
<td>69.6</td>
<td>71.1</td>
</tr>
<tr>
<td>ArgM</td>
<td>95.4</td>
<td>94.1</td>
<td>96.2</td>
<td>96.2</td>
<td>96.1</td>
<td>96.3</td>
</tr>
<tr>
<td>Global Accuracy</td>
<td>90.5</td>
<td>88.7</td>
<td>91.3</td>
<td>90.4</td>
<td>92.4</td>
<td>93.2</td>
</tr>
</tbody>
</table>
Argument Classification on PAT using different Tree Fragment Extractor

![Graph showing the accuracy of argument classification on PAT using different tree fragment extractors as a function of the percentage of training data. The graph plots accuracy (%) against the percentage of training data (%). There are four lines representing different extractors: ST, SST, Linear, and PT. The accuracy increases as the percentage of training data increases.]
Boundary Detection

S

N
Paul

V
delivers

D

N
a
talk

IN
in

jj

IN

NP

N

Arg. 1

formal

style
Improvement by Marking Boundary nodes

A) PAF+
   VP
   V
   D
   N
   delivers a talk

   PAF-
   VP
   V
   NP
   N
   delivers talk

B) MPAF+
   VP
   V
   NP
   NP-B
   delivers a talk

   MPAF-
   VP
   V
   NP
   N-B
   delivers talk
Node Marking Effect

C) \[ \text{VP} \]
\[ \text{V} \quad \text{NP} \quad \text{VP} \quad \text{V} \quad \text{N} \]
\[ \text{delivers} \quad \text{V} \quad \text{NP} \quad \text{delivers} \quad \text{talk} \]

D) \[ \text{common MPAF features} \]
\[ \text{V} \]
\[ \text{delivers} \]
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

<table>
<thead>
<tr>
<th></th>
<th>Section 2</th>
<th></th>
<th>Section 24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>pos</td>
<td>neg</td>
<td>tot</td>
</tr>
<tr>
<td>Internal</td>
<td>11,847</td>
<td>71,126</td>
<td>82,973</td>
</tr>
<tr>
<td>Pre-terminal</td>
<td>894</td>
<td>114,052</td>
<td>114,946</td>
</tr>
<tr>
<td>Both</td>
<td>12,741</td>
<td>185,178</td>
<td>197,919</td>
</tr>
</tbody>
</table>
Predicate Argument Feature (PAF) vs. Marked PAF (MPAF) [Moschitti et al, CLJ 2008]

<table>
<thead>
<tr>
<th>Tagging strategy</th>
<th>CPU$_{time}$</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAF</td>
<td>5,179.18</td>
<td>75.24</td>
</tr>
<tr>
<td>MPAF</td>
<td>3,131.56</td>
<td>82.07</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>Eval Setting</th>
<th>$P$</th>
<th>$R$</th>
<th>$F_1$</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>
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%

<table>
<thead>
<tr>
<th>Evaluation Stage</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Detection</td>
<td>0.905</td>
<td>0.873</td>
<td>0.889</td>
</tr>
<tr>
<td>Boundary Detection + Role Classification</td>
<td>0.774</td>
<td>0.747</td>
<td>0.760</td>
</tr>
</tbody>
</table>

- Automatic SRL viable for Spoken Dialog Data
The Relation Extraction Problem

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

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
Relation Extraction: The task

- Task definition: to label the semantic relation between pairs of entities in a sentence
  - The **governor** from **Connecticut**
  - Is there a relation between M1 and M2? If, so what kind of relation?

M1 type: PER  |  M2 type: LOC

M := Entity Mention
Relation Extraction defined in ACE

- Major relation types (from ACE 2004)

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

- Entity types: PER, ORG, LOC, GPE, FAC, VEH, WEA
System Description (Nguyen et al, 2009)

ACE documents → Raw texts → Stanford Parser → Parse Trees with Entities → Tree Kernel-based SVMs → Multi-class Classification → Entities and Relations → RELATIONS
The Path-enclosed tree captures the “PHYSICAL.LOCATED” relation between “corporation” and “Iowa”
## Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Data</th>
<th>P (%)</th>
<th>R (%)</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al. (2006) Composite Kernel (linear) with Context-Free Parse Tree</td>
<td>ACE 2004</td>
<td>73.5</td>
<td>67.0</td>
<td>70.1</td>
</tr>
<tr>
<td>Ours Composite Kernel (linear) with Context-Free Parse Tree</td>
<td>ACE 2004</td>
<td>69.6</td>
<td>68.2</td>
<td>69.2</td>
</tr>
</tbody>
</table>

Both use the Path-Enclosed Tree for Relation Representation
Several Combination Kernels

[Vien et al, EMNLP 2009]

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

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

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

\[ CK_4 = K_{PTK-DW} + K_{PTK-GR} \]

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

\[ SSK = \sum_{i=1,\ldots,6} SK_i \]

\[ CSK = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{SST} + SSK) \]
# Results on ACE 2004

<table>
<thead>
<tr>
<th>Kernel</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK₁</td>
<td>69.5</td>
<td>68.3</td>
<td>68.9</td>
</tr>
<tr>
<td>SK₁</td>
<td>72.0</td>
<td>52.8</td>
<td>61.0</td>
</tr>
<tr>
<td>SK₂</td>
<td>61.7</td>
<td>60.0</td>
<td>60.8</td>
</tr>
<tr>
<td>SK₃</td>
<td>62.6</td>
<td>60.7</td>
<td>61.6</td>
</tr>
<tr>
<td>SK₄</td>
<td>73.1</td>
<td>50.3</td>
<td>59.7</td>
</tr>
<tr>
<td>SK₅</td>
<td>59.0</td>
<td>60.7</td>
<td>59.8</td>
</tr>
<tr>
<td>SK₆</td>
<td>57.7</td>
<td>61.8</td>
<td>59.7</td>
</tr>
<tr>
<td>SK₃ + SK₄</td>
<td>75.0</td>
<td>63.4</td>
<td>68.8</td>
</tr>
<tr>
<td>SK₃ + SK₆</td>
<td>66.8</td>
<td>65.1</td>
<td>65.9</td>
</tr>
<tr>
<td>SSK = Σᵢ SKᵢ</td>
<td>73.8</td>
<td>66.2</td>
<td>69.8</td>
</tr>
<tr>
<td>SST Kernel + SSK</td>
<td>75.6</td>
<td>66.6</td>
<td>70.8</td>
</tr>
<tr>
<td>CK₁ + SSK</td>
<td>76.6</td>
<td>67.0</td>
<td>71.5</td>
</tr>
<tr>
<td>(Zhou et al., 2007)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CK₁ with Heuristics</td>
<td>82.2</td>
<td>70.2</td>
<td>75.8</td>
</tr>
</tbody>
</table>
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]

<table>
<thead>
<tr>
<th></th>
<th>MUC-6</th>
<th></th>
<th></th>
<th>ACE-02-BNews</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>P</td>
<td>F</td>
<td>R</td>
<td>P</td>
<td>F</td>
</tr>
<tr>
<td>All attribute value features</td>
<td>64.3</td>
<td>63.1</td>
<td>63.7</td>
<td>58.9</td>
<td>68.1</td>
<td>63.1</td>
</tr>
<tr>
<td>+ Syntactic Tree</td>
<td>65.2</td>
<td>80.1</td>
<td>71.9</td>
<td>65.6</td>
<td>69.7</td>
<td>67.6</td>
</tr>
<tr>
<td>+ Word Sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Results on the overall Coreference Resolution using SVMs

<table>
<thead>
<tr>
<th></th>
<th>Basic Features SVMs</th>
<th>Basic Features + Syntax Tree</th>
<th>Basic Features + SyntaxTree + Word Sequences</th>
<th>All Sources of Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MUC-6</td>
<td>ACE02-BNews</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>P</td>
<td>F</td>
<td>R</td>
</tr>
<tr>
<td>Basic Features</td>
<td>61.5</td>
<td>67.2</td>
<td>64.2</td>
<td>54.8</td>
</tr>
<tr>
<td>SVMs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Features +</td>
<td>63.4</td>
<td>67.5</td>
<td>65.4</td>
<td>56.6</td>
</tr>
<tr>
<td>Syntax Tree</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Features +</td>
<td>64.4</td>
<td>67.8</td>
<td>66.0</td>
<td>57.1</td>
</tr>
<tr>
<td>SyntaxTree + Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequences</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All Sources of</td>
<td>60.1</td>
<td>76.2</td>
<td>67.2</td>
<td>60.0</td>
</tr>
<tr>
<td>Knowledge</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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:

\[ K(x_1, y_1) = \text{PTK}(q_{x_1}, q_{y_1}) + \text{PTK}(a_{x_1}, a_{y_1}) \]
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*

  \[
  \text{list} \quad \text{TWA} \quad \text{flights} \quad \text{from} \quad \text{Boston} \quad \text{to} \quad \text{Philadelphia}
  \]

  \[
  \text{null} \quad \text{airline\_code} \quad \text{null} \quad \text{null} \quad \text{fromloc\_city} \quad \text{null} \quad \text{toloc\_city}
  \]

- The concepts are used to build rules for the dialog manager (e.g. actions for using the DB)
  - from location
  - to location
  - airline code
    
    \[
    \begin{array}{l}
    \text{list flights from boston to Philadelphia} \\
    \text{FRAME: FLIGHT} \\
    \text{FROMLOC.CITY = boston} \\
    \text{TOLOC.CITY = Philadelphia}
    \end{array}
    \]
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

Input Utterance

Hypotheses
H1
H2
H3
...
Hn

Pairs
<H1,H2>
<H1,H3>
...
<Hn,H1>
<Hn,H2>

Re-ranker

Hypotheses
H4
H3
...
H1
Hn
Reranking concept labeling

- I have a problem with my monitor

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

$H^i$: I NULL have NULL a NULL problem $\text{HW-B}$ with NULL my NULL monitor
## Luna Corpus

- **Wizard of OZ, helpdesk scenario**

<table>
<thead>
<tr>
<th>Corpus LUNA</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>words</td>
<td>concepts</td>
</tr>
<tr>
<td>Dialogs</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>Turns</td>
<td>1,019</td>
<td></td>
</tr>
<tr>
<td>Tokens</td>
<td>8,512</td>
<td>2,887</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>1,172</td>
<td>34</td>
</tr>
<tr>
<td>OOV rate</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
# Media Corpus

<table>
<thead>
<tr>
<th></th>
<th>training</th>
<th>development</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td># sentences</td>
<td>12,908</td>
<td>1,259</td>
<td>3,005</td>
</tr>
<tr>
<td># tokens</td>
<td>94,466</td>
<td>10,849</td>
<td>25,606</td>
</tr>
<tr>
<td># vocabulary</td>
<td>2,210</td>
<td>838</td>
<td>1,276</td>
</tr>
<tr>
<td># OOV rate [%]</td>
<td>–</td>
<td>1.33</td>
<td>1.39</td>
</tr>
<tr>
<td></td>
<td>43,078</td>
<td>4,705</td>
<td>11,383</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>66</td>
<td>78</td>
</tr>
</tbody>
</table>
I have a problem with my monitor.
Cross-language approach: Italian version

```
ROOT

NULL  PROBLEM-B  PROBLEM-I  HW-B  HW-I

Ho   un    problema  col  monitor
```
Multilevel Tree

ROOT

NULL

PROBLEM

Ho

PROBLEM-B

un

PROBLEM-I

problema

HW

HW-B

col

HW-I

monitor
Enriched Multilevel Tree

ROOT

PROBLEM

NULL

PROBLEM-B

F0:Ho  F1:Ho  F0:un  F1:ART

PROBLEM-I

F0:problema  F1:problema

HW

HW-B

F0:col  F1:SPRE

HW-I

F0:monitor  F1:monitor
# Results on LUNA

<table>
<thead>
<tr>
<th>Model</th>
<th>Text Input (CER)</th>
<th>Speech Input (CER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FST</td>
<td>24.4%</td>
<td>27.4%</td>
</tr>
<tr>
<td>SVM</td>
<td>25.3%</td>
<td>27.1%</td>
</tr>
<tr>
<td>CRF</td>
<td>21.3%</td>
<td>23.5%</td>
</tr>
<tr>
<td>FST-RR</td>
<td>20.7%</td>
<td>22.8%</td>
</tr>
<tr>
<td>CRF-RR</td>
<td>19.9%</td>
<td>21.9%</td>
</tr>
<tr>
<td>$FST + RR_S$</td>
<td>19.2%</td>
<td>21.5%</td>
</tr>
<tr>
<td>$CRF + RR_S$</td>
<td>19.0%</td>
<td>21.1%</td>
</tr>
</tbody>
</table>
## Results on Media

<table>
<thead>
<tr>
<th>Model</th>
<th>Text Input (CER)</th>
<th>Speech Input (CER)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FST</strong></td>
<td>14.2%</td>
<td>17.0%</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>13.4%</td>
<td>15.9%</td>
</tr>
<tr>
<td><strong>CRF</strong></td>
<td>11.7%</td>
<td>14.2%</td>
</tr>
<tr>
<td><strong>FST-RR</strong></td>
<td>11.9%</td>
<td>14.6%</td>
</tr>
<tr>
<td><strong>CRF-RR</strong></td>
<td>11.5%</td>
<td>14.1%</td>
</tr>
<tr>
<td>(FST + RR_S)</td>
<td>11.3%</td>
<td>13.8%</td>
</tr>
<tr>
<td>(CRF + RR_S)</td>
<td>11.1%</td>
<td>13.1%</td>
</tr>
</tbody>
</table>
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
Today, a car was pushed into a ravine.

SVMs F1 from 75.89 to 77.25
Relational Kernels
Recognizing Textual Entailment

... the *textual entailment recognition* task:

determine whether or not a text $T$ implies a hypothesis $H$

$T_1 \Rightarrow H_1$

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>“At the end of the year, all solid companies pay dividends.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>“At the end of the year, all solid insurance companies pay dividends.”</td>
</tr>
</tbody>
</table>

“Traditional” machine learning approaches:

similarity-based methods $\rightarrow$ distance in feature spaces
Determine Intra-pair links

\[ T_1 \]

\[ H_1 \]

\[ T_3 \]

\[ H_3 \]
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

\[ K_s((T', H'), (T'', H'')) = \]

\[ \max_{c \in C} \left( K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i)) \right) \]

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

<table>
<thead>
<tr>
<th></th>
<th>BOW+LS</th>
<th>+ TK</th>
<th>+ $K_{ent}$</th>
<th>System Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTE1</td>
<td>0.5888</td>
<td>0.6213</td>
<td>0.6300</td>
<td>0.54</td>
</tr>
<tr>
<td>RTE2</td>
<td>0.6038</td>
<td>0.6238</td>
<td>0.6388</td>
<td>0.59</td>
</tr>
<tr>
<td>System</td>
<td>Strategy</td>
<td>Decision</td>
<td>An. Level</td>
<td>Knowledge Resources</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>(Hickl et al., 2006)</td>
<td>lex,syn, trg</td>
<td>mlr</td>
<td>lxs,synt</td>
<td>WN, paraph, PropBank</td>
</tr>
<tr>
<td>(Tatu and Moldovan, 2006)</td>
<td>lex</td>
<td>thr,inf</td>
<td>sur,syn</td>
<td>WN, SUMO, ExtWN, axioms</td>
</tr>
<tr>
<td>(Zanzotto et al., 2006)</td>
<td>syn</td>
<td>mlr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Adams, 2006)</td>
<td>lex</td>
<td>mlr</td>
<td>sur,lxs</td>
<td>WN</td>
</tr>
<tr>
<td>(Bos and Markert, 2006)</td>
<td>lex</td>
<td>mlr,inf</td>
<td>sur,lxs</td>
<td>WN, axioms</td>
</tr>
<tr>
<td>(Kouylekov and Magnini, 2006)</td>
<td>synt</td>
<td>thr,mlr</td>
<td>lxs,syn</td>
<td>WN, DIRT</td>
</tr>
<tr>
<td>(MacCartney et al., 2006)</td>
<td>synt</td>
<td>mlr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Snow et al., 2006)</td>
<td>trg, lex</td>
<td>rul,mlr</td>
<td>lxs,syn</td>
<td>WN, MindNet, thes</td>
</tr>
<tr>
<td>(Herrera et al., 2006)</td>
<td>lex,syn</td>
<td>mlr</td>
<td>lex,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Nielsen et al., 2006)</td>
<td>lex,syn</td>
<td>mlr</td>
<td>sur,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Marsi et al., 2006)</td>
<td>syn</td>
<td>thr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Katrenko and Adriaans, 2006)</td>
<td>lex,syn</td>
<td>mlr</td>
<td>syn</td>
<td>WN, FrameNet, SUMO</td>
</tr>
<tr>
<td>(Burchardt and Frank, 2006)</td>
<td>syn</td>
<td>mlr</td>
<td>lxs,syn</td>
<td>WN, FrameNet, SUMO</td>
</tr>
<tr>
<td>(Rus, 2006)</td>
<td>syn</td>
<td>thr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Litkowski, 2006)</td>
<td>lex</td>
<td>thr</td>
<td>sur</td>
<td></td>
</tr>
<tr>
<td>(Inkpen et al., 2006)</td>
<td>trg, lex</td>
<td>mlr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Ferrendez et al., 2006)</td>
<td>syn</td>
<td>thr</td>
<td>lex,syn</td>
<td>WN</td>
</tr>
<tr>
<td>(Schilder and McInnes, 2006)</td>
<td>lex,syn</td>
<td>mlr</td>
<td>lxs,syn</td>
<td>WN</td>
</tr>
</tbody>
</table>
Relational Kernels for Answer Reranking
An example of Jeopardy! Question
The abbey is where all English monarchs have been crowned since William the Conqueror.
Baseline Model

Methodology:

1-Applying PTK without any extra annotation and evaluate the model as baseline.
One of the English kings since William the Conqueror were never crowned.
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.
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: [Sveryn&Moschitti, SIGIR2012]

Question

is movie theater popcorn vegan

bag of pos tags

bag of words

and their combination

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

Answer

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

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

(DT) (NN) (NN) (NN) (WDT) (VBZ) (NN) (CC) (RB) (JJ) (NNS) (VBZ) (RB) (NN)
Lexical matching is on word lemmas (using WordNet lemmatizer)
However, the popcorn kernels alone can be considered vegan if popped using canola or coconut oil, which some theaters offer as an alternative to standard popcorn.

Question sentence:

Is any movie theater popcorn that includes butter and therefore dairy products vegan?

Lexical matching is on word lemmas (using WordNet lemmatizer).
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

![Graph showing learning curves for different models with MRR (Mean Reciprocal Rank) on the y-axis and training size (in thousands) on the x-axis. The graph compares models labeled PTK, STK, and baseline.](image)
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
6. CONCLUSIONS

The key aspect in learning to rank answer passages for question answering is the use of relationships between the question and answer passage. We experimented with Support Vector Machines (SVMs) to learn these relationships and improve the performance of the question answering system. Our experiments with SVMs showed that they can be used to improve the accuracy of question answering systems, particularly when combined with other features such as shallow syntactic tree structures. However, additional pre-processing is required for some models, and we found that a simpler model outperforms semantic models when training a model on larger datasets.

We also studied the learning curves for these models on different training sets. The plots demonstrate nice scaling behavior when training the reranker model on larger data. The plots show that the learning curves are steepest for some models, while maintaining the optimal accuracy. Therefore, we build learning curves for the reranker model using different tag sets, which require additional pre-processing and form POST tagging and chunking over more refined models. We prefer a simpler model which only requires to perform shallow syntactic tree structures as NtR or WNSS.

On the other hand, we exploit the power of supervised methods to learn the properties above. Supervised methods can generalize the properties found in large datasets and enable the learning algorithm to learn them automatically. We used different question-answer pairs and used them to evaluate the performance of our models. We found that the performance of our models improved up to 10 points for an error reduction of 5 to 10 points in most cases.

We also studied the performance of our models on different question datasets. Our models allow for efficient and automatically derived encoding pair properties. Additionally, large scale experiments show that significant improvement can be obtained by combining different models and using different tag sets.

Therefore, we propose robust and simple models to learn the properties above. We show that these models can be used to improve the performance of question answering systems and that they can be combined with other features to achieve better accuracy.
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)

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

\[
g^{(j)} = \frac{1}{n} \sum_{k=1}^{n} c_{k}^{(j)} y_k \phi(\vec{x}_k)
\]
Computing most violated constraint (MVC)

\[
\mathbf{w} \cdot \phi(\mathbf{x}_i) = \sum_{j=1}^{t} \alpha_j \mathbf{g}^{(j)} \cdot \phi(\mathbf{x}_i)
\]

\[
\mathbf{g}^{(j)} = \frac{1}{n} \sum_{k=1}^{n} c_k^{(j)} y_k \phi(\mathbf{x}_k)
\]

\[
\mathbf{w} \cdot \phi(\mathbf{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{n} \left( \frac{1}{n} c_k^{(j)} y_k \right) K(\mathbf{x}_i, \mathbf{x}_k)
\]
Approximate CPA (Yu & Joachims, 2009)

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

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

- Use sampling to approximate exact cutting plane models

\[
\tilde{w} \cdot \phi(\tilde{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{r} \left( \frac{1}{r} c_k^{(j)} y_k \right) K(\tilde{x}_i, \tilde{x}_k)
\]
Three syntactic trees and the resulting DAG

VP,1

VP

VP,1

NP,2

V,2

NP,1

D,3

NP,1

NP

V

NP

D

N

buy

red

car

a

car

buy

a

car

buy,2

D,3

JJ,1

N,3

a,3

red,1

car,3
Three syntactic trees and the resulting DAG

VP
  \ / 
V NP
  |   |   |   
buy D JJ N
  |   |   |   
a red car

NP V NP
  |   |   
D N buy D N
  |   |   
a car a car

VP
  VP,1
    \ / 
  VP,1 | VP,1
    |   |   
  buy,2 | NP,2
    |   |   
  D,3 | JJ,1 | N,3
    |   |   
  a,3 red,1 car,3
SDAG

- Compacts each CPA model into a single DAG

\[
\mathbf{w} \cdot \phi(\mathbf{x}_i) = \sum_{j=1}^{t} \alpha_j \sum_{k=1}^{r} \left( \frac{1}{r} c_{j}^{(k)} y_k \right) K(\mathbf{x}_i, \mathbf{x}_k)
\]

\[
\mathbf{w} \cdot \phi(\mathbf{x}_i) = \sum_{j=1}^{t} \alpha_j K_{dag}(\text{dag}_{(j)}, \mathbf{x}_i)
\]
SDAG+

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

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

\[
\hat{\omega} \cdot \phi(\vec{x}_i) = K_{dag}(\vec{dag}(t), \vec{x}_i)
\]
Reverse Kernel Engineering

- **Input**: an SVM model, i.e., $\hat{\mathcal{W}}$
- **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: \texttt{MINE\_MODEL}(M, L, E, \lambda)

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

- 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:
  - \texttt{FRAG}(n)
  - \texttt{EXPAND}(f, E)
- Parameters:
  - $\text{maxexp} \; (E)$
  - 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 → fragments):

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

where:
- \( t_i \) is the \( i \)-th tree in the model
- \( \alpha_i \) 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 \( t_i \)
- \( \lambda \) is the kernel decay factor
- \( \ell(f_j) \) is the depth (number of levels) of the fragment
Reverse Engineering Framework

Train $\langle y, t \rangle$

Split$_1$

\ldots

Split$_S$
Reverse Engineering Framework

Train \( \langle y, t \rangle \)

Split \( s \) \( \rightarrow \) SVM learn TK \( \rightarrow \) \( M_s \)

\( \cdots \) \( \rightarrow \) \( \cdots \) \( \rightarrow \) \( \cdots \)

Split \( 1 \) \( \rightarrow \) SVM learn TK \( \rightarrow \) \( M_1 \)

FSL = Fragment Space Learning
Reverse Engineering Framework

Train $\langle y, t \rangle$

$\text{Split}_1 \rightarrow \text{SVM learn TK} \rightarrow M_1 \rightarrow \text{mine}(L, E, D)$

$\cdots \cdots \cdots \cdots \cdots \rightarrow \mathcal{D}_L$

$\text{Split}_S \rightarrow \text{SVM learn TK} \rightarrow M_S \rightarrow \text{mine}(L, E, D)$

$\text{FSL} = \text{Fragment Space Learning}$

$\text{FMI} = \text{Fragment Mining and Indexing}$
Reverse Engineering Framework

Train $\langle y, t \rangle$

Linearize

Train $\ell \langle y, x \rangle$

Split$_1$ $\rightarrow$ SVM learn TK $\rightarrow$ M$_1$ $\rightarrow$ mine($L, E, D$)

... $\rightarrow$ ... $\rightarrow$ ... $\rightarrow$ ...

Split$_S$ $\rightarrow$ SVM learn TK $\rightarrow$ M$_S$ $\rightarrow$ mine($L, E, D$)

$D_L$

FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

FMI = Fragment Mining and Indexing
Reverse Engineering Framework

Train \( \langle y, t \rangle \) → Split_1 → SVM learn TK → M_1 → mine(L, E, D)

Train_\ell \langle y, x' \rangle → Split_\ell → SVM learn TK → M_\ell

\( D_L \) → ...

FSL = Fragment Space Learning
FMI = Fragment Mining and Indexing
TFX = Tree Fragment eXtraction
ESL = Explicit Space Learning
Reverse Engineering Framework

Train \( \langle y, t \rangle \)

Train\( _\ell \langle y, x \rangle \)

Test \( \langle y, t \rangle \)

FSL = Fragment Space Learning

TFX = Tree Fragment eXtraction

FMI = Fragment Mining and Indexing

ESL = Explicit Space Learning
Reverse Engineering Framework

Train \( \langle y, t \rangle \) → Split_1 → SVM learn TK → M_1 → mine(\( L, E, D \))

... ... ... ... ...

Train_\ell \langle y, \bar{x} \rangle → Split_S → SVM learn TK → M_S → mine(\( L, E, D \))

\( \mathcal{D}_L \)

Train_\ell \langle y, \bar{x} \rangle → SVM learn → M_\ell

Test_\ell \langle y, \bar{x} \rangle ← Linearize

Test \langle y, t \rangle

FSL = Fragment Space Learning

FMI = Fragment Mining and Indexing

TFX = Tree Fragment eXtraction

ESL = Explicit Space Learning
Reverse Engineering Framework

Train $\langle y, t \rangle$

Linearize

Train $\ell \langle y, \bar{x} \rangle$

Predictions

Test $\langle y, t \rangle$

FSL = Fragment Space Learning

FMI = Fragment Mining and Indexing

TFX = Tree Fragment eXtraction

ESL = Explicit Space Learning
Semantic Role Labeling
Setting

BC/BC$_{\ell}$:
- training: 1 Mil AST$_{ms}$ from PB secs 2-6
- test: 149,140 AST$_{ms}$ from PB sec 24

RM/RM$_{\ell}$:
- 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$_{\ell}$ 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)
About 10 time faster - Training (and testing)
Parallelizable!

<table>
<thead>
<tr>
<th>Class</th>
<th>Data set</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tr⁺</td>
<td>Te⁺</td>
</tr>
<tr>
<td>BC</td>
<td>61,062</td>
<td>8,515</td>
</tr>
<tr>
<td>A0</td>
<td>60,900</td>
<td>2,014</td>
</tr>
<tr>
<td>A1</td>
<td>90,636</td>
<td>3,041</td>
</tr>
<tr>
<td>A2</td>
<td>21,291</td>
<td>697</td>
</tr>
<tr>
<td>A3</td>
<td>3,481</td>
<td>105</td>
</tr>
<tr>
<td>A4</td>
<td>2,713</td>
<td>69</td>
</tr>
<tr>
<td>A5</td>
<td>69</td>
<td>2</td>
</tr>
<tr>
<td>RM</td>
<td>87.8</td>
<td></td>
</tr>
</tbody>
</table>

Table: Number of positive training (Tr⁺) and test (Te⁺) examples in the SRL dataset. Accuracy of the non-linearized (SST) and linearized (SSTₖ) binary classifiers (i.e. BC, A0, ... A5) is F₁ measure. Accuracy of RM is the percentage of correct class assignments.

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
- **SST\(\ell\):** linearized tree kernel

<table>
<thead>
<tr>
<th>Class</th>
<th>Data set</th>
<th>Accuracy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tr(^+)</td>
<td>Te(^+)</td>
<td>SST</td>
<td>SST(\ell)</td>
</tr>
<tr>
<td>ABBR</td>
<td>89</td>
<td>9</td>
<td>80.0</td>
<td>87.5</td>
</tr>
<tr>
<td>DESC</td>
<td>1,164</td>
<td>138</td>
<td>96.0</td>
<td>94.5</td>
</tr>
<tr>
<td>ENTY</td>
<td>1,269</td>
<td>94</td>
<td>63.9</td>
<td>63.5</td>
</tr>
<tr>
<td>HUM</td>
<td>1,231</td>
<td>65</td>
<td>88.1</td>
<td>87.2</td>
</tr>
<tr>
<td>LOC</td>
<td>834</td>
<td>81</td>
<td>77.6</td>
<td>77.9</td>
</tr>
<tr>
<td>NUM</td>
<td>896</td>
<td>113</td>
<td>80.4</td>
<td>80.8</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td>86.2</td>
<td>86.6</td>
</tr>
</tbody>
</table>
Interpretation (Abbreviation Class)

(\text{NN(abbreviation)})
(\text{NP(DT)(NN(abbreviation))})
(\text{NP(DT(the))(NN(abbreviation))})
(\text{IN(for)})
(\text{VB(stand)})
(\text{VBZ(does)})
(\text{PP(IN)})
(\text{VP(VB(stand))(PP)})
(\text{NP(NP(DT)(NN(abbreviation))))(PP)})
(\text{SQ(VBZ)(NP)(VP(VB(stand))(PP))})
(\text{SBARQ(WHNP)(SQ(VBZ)(NP)(VP(VB(stand))(PP)))(.)})
(\text{SQ(VBZ(does))(NP)(VP(VB(stand))(PP))})
(\text{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:
  - 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


References


References

- Alessandro Moschitti’ handouts http://disi.unitn.eu/~moschitt/teaching.html
References


References


References


References


References


References


References


References


An introductory book on SVMs, Kernel methods and Text Categorization

Roberto Basili
Alessandro Moschitti

Automatic Text Categorization

From Information Retrieval to Support Vector Learning
Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors


Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors

Non-exhaustive reference list from other authors