Tree Kernels

WM&IR a.a. 2013/2014

A. Moschitti, R. Basili
Revised (D. Croce)

Dipartimento di Informatica Sistemi e produzione
Università di Roma “Tor Vergata”
basili@info.uniroma2.it
Outline

• Motivations
• Kernels for syntagmatic structures
• Three different kernels
  – Expressivity
  – Complexity
• Two applications
  – Semantic role labeling
  – Question classification
Motivations

• Classification over highly structured data require complex similarity functions
• Modeling syntax in Natural Language learning tasks is crucial, e.g.
  – Syntactic disambiguation
  – Semantic role relations within predicate argument structures
  – 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?
Outline

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

• Complexity of the kernel computations
  – Efficient evaluation of PT kernel

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

• Comparative Evaluation of different Tree kernels
• Conclusions
The Collins and Duffy’s Tree Kernel 
(called SST in [Vishwanathan and Smola, 2002])

[Diagram of a tree kernel structure with labels: VP, V, NP, D, N, gives, a, talk]
The overall fragment set
Making Explicit the Feature Space

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

- \( \vec{x}_1 \cdot \vec{x}_2 \) counts the number of common substructures
Computing over the implicit representation

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

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

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

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

• Given the equation for the SST kernel (subset), i.e.

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

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

• …
Evaluation

- Given the equation for the ST kernel

\[ \Delta(n_1, n_2) = 0, \text{ if the productions are different} \]
\[ \Delta(n_1, n_2) = 1, \text{ if pre-terminal} \]
\[ \Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\Delta(ch(n_1, j), ch(n_2, j))) \]
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
  \[
  \Delta(n_1, n_2) = 1 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])
  \]

• By adding two decay factors we obtain:

\[
\mu \left( \lambda^2 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \lambda^{d(\vec{J}_1) + d(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}]) \right)
\]
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), \( \Delta_p(s_1 a, s_2 b) = \)

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

• We have different kernels that induce different feature spaces.
• How should such kernel functions be used?
• An answer can be given to the problem of encoding syntactic information.
• As example we study two different tasks requiring syntactic information.
Semantic Role Labeling

• Given an event:
  – Some words describe the relation among different participants
  – Such words can be considered predicates
  – The participants are their arguments.

• Example:
  Paul gives a lecture in Rome
Semantic Role Labeling

• Given an event:
  – Some words describe the relation among different participants
  – Such words can be considered predicates
  – The participants are their arguments.

• Example:

  \[ \text{Arg}_0 \text{ Paul} \text{ gives } \text{Arg}_1 \text{ a lecture} \text{ in Rome} \]
Semantic Role Labeling

• Given an event:
  – Some words describe the relation among different participants
  – Such words can be considered predicates
  – The participants are their arguments.

• Example:
  \[ [Arg_0 \text{ Paul}] [\text{ predicate} \text{ gives}] [Arg_1 \text{ a lecture}] [Arg_M \text{ in Rome}] \]

• PropBank and FrameNet propose two different theories and resources
Semantic/Syntactic structures

- Given a sentence with its semantic annotation:

\[
[ \text{Arg}_0 \text{ Paul} ] [ \text{predicate} \text{ gives} [ \text{Arg}_1 \text{ a lecture} ] [ \text{Arg}_M \text{ in Rome} ]
\]
A Tree Kernel for Semantic Role labeling

A Tree Kernel for Semantic Role Labeling
Gold Standard Tree Experiments

- 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 experiments (on the paper)
SVM-light-TK Software

- Encodes ST and SST in SVM-light [Joachims, 1999]
- Available at http://disi.unitn.it/moschitti/Tree-Kernel.htm
Running Time of Tree Kernel Functions

- FTK-SST
- QTK-SST
- FTK-PT
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?”
Exercise

• Learn a SVM model within a Bag-of-Word and Tree Representation
  – Only the $\text{Enty}$ class is considered

• Use the SVM-Light-TK-1.2 (http://disi.unitn.it/moschitti/TK1.2-software/download.html)
  – Linear kernel and Bow:
    ./svm_learn -t 0 bow/ENTY_train.dat linear_model
    ./svm_classify bow/ENTY_test.dat linear_model linear_res
  – Polynomial kernel (exponent d=2)
    ./svm_learn -t 1 -d 2 bow/ENTY_train.dat linear_poly2
    ./svm_classify bow/ENTY_test.dat linear_poly2 poly2_res
  – Subset tree kernel
    ./svm_learn -t 5 tree/ENTY_train.dat SSTK_model
    ./svm_classify tree/ENTY_test.dat SSTK_model SSTK_res
Homework

• Given the data (labeled with all the coarse classes) implement a multi-classifier
  – One-VS-ALL schema can be applied
    • The target classes can be derived by the original dataset
  – Parameterize the SVM learning algorithm and measure the Precision/Recall with respect to each class
Conclusions

• 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.
  – e.g., manual feature design to encode predicate argument relations is complex
• 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
• Experiments on SRL and QC show that
  – PT and SST are efficient and very fast
  – Higher accuracy when the opportune kernel is used for the target task
References

• Accessible from the network: